Notice

The information contained in this document is subject to change without notice.

UNLESS EXPRESSLY SET FORTH IN A WRITTEN AGREEMENT SIGNED BY AN AUTHORIZED REPRESENTATIVE OF STRATUS TECHNOLOGIES, STRATUS MAKES NO WARRANTY OR REPRESENTATION OF ANY KIND WITH RESPECT TO THE INFORMATION CONTAINED HEREIN, INCLUDING WARRANTY OF MERCHANTABILITY AND FITNESS FOR A PURPOSE. Stratus Technologies assumes no responsibility or obligation of any kind for any errors contained herein or in connection with the furnishing, performance, or use of this document.

Software described in Stratus documents (a) is the property of Stratus Technologies Bermuda, Ltd. or the third party, (b) is furnished only under license, and (c) may be copied or used only as expressly permitted under the terms of the license.

Stratus documentation describes all supported features of the user interfaces and the application programming interfaces (API) developed by Stratus. Any undocumented features of these interfaces are intended solely for use by Stratus personnel and are subject to change without warning.

This document is protected by copyright. All rights are reserved. Stratus Technologies grants you limited permission to download and print a reasonable number of copies of this document (or any portions thereof), without change, for your internal use only, provided you retain all copyright notices and other restrictive legends and/or notices appearing in the copied document.

Stratus, the Stratus logo, ftServer, the ftServer logo, Continuum, StrataLINK, and StrataNET are registered trademarks of Stratus Technologies Bermuda, Ltd.

The Stratus Technologies logo, the Continuum logo, the Stratus 24 x 7 logo, ActiveService, Automated Uptime, ftScalable, and ftMessaging are trademarks of Stratus Technologies Bermuda, Ltd.

RSN is a trademark of Lucent Technologies, Inc.
All other trademarks are the property of their respective owners.


Part Number: R098
Revision Number: 19
OpenVOS Release Number: 19.1.0
Publication Date: November 2017

Stratus Technologies, Inc.
5 Mill and Main Place, Suite 500
Maynard, Massachusetts 01754-2660

© 2017 Stratus Technologies Bermuda, Ltd. All rights reserved.
Preface

The OpenVOS Commands Reference Manual (R098) documents the OpenVOS operating system commands and command functions for the OpenVOS release shown on the Notice page.

This manual is intended for users who have access to the OpenVOS operating system, applications and systems programmers, and system administrators.

Manual Version

This manual is a revision. Change bars, which appear in the margin, note the specific changes to text since the previous publication of this manual.

This revision incorporates the following new commands or command functions.

\[
\begin{array}{|c|c|}
\hline
\text{(date)} & \text{(iso\_date)} \\
\text{(date\_time)} & \text{(iso\_date\_time)} \\
\text{(time)} & \\
\hline
\end{array}
\]

This revision incorporates changes to the following commands or command functions.

\textit{list\_batch\_requests}

The term OpenVOS applies to operating system releases from Release 17.0.0 and later. The term VOS applies to operating system releases prior to OpenVOS Release 17.0.0.

Manual Organization

This manual has two chapters and three appendixes.

Chapter 1 describes the OpenVOS command functions.

Chapter 2 describes the OpenVOS commands.

Appendix A describes tape parameter values.

Appendix B describes the general software limits for OpenVOS.

Appendix C describes a way to reduce program module size when using shared virtual memory databases.
Preface

Related Manuals

See the following Stratus manuals for related documentation.

- VOS Reference Manual (R002)
- OpenVOS Commands User’s Guide (R089)
- Using OpenVOS Extended Names (R631)

See also the OpenVOS System Administration manuals.

- OpenVOS System Administration: Administering and Customizing a System (R281)
- OpenVOS System Administration: Starting Up and Shutting Down a Module or System (R282)
- OpenVOS System Administration: Registration and Security (R283)
- OpenVOS System Administration: Disk and Tape Administration (R284)
- OpenVOS System Administration: Backing Up and Restoring Data (R285)
- OpenVOS System Administration: Configuring a System (R287)

Notation Conventions

This manual uses the following notation conventions.

- Italics introduces or defines new terms. For example:
  
  The *master disk* is the name of the member disk from which the module was booted.

- Boldface emphasizes words in text. For example:

  Every module **must** have a copy of the *module_start_up.cm* file.

- Monospace represents text that would appear on your terminal’s screen (such as commands, subroutines, code fragments, and names of files and directories). For example:

  ```
  change_current_dir (master_disk)>system>doc
  ```

- Monospace italic represents terms that are to be replaced by literal values. In the following example, the user must replace the monospace-italic term with a literal value.

  ```
  list_users -module module_name
  ```

- Monospace bold represents user input in examples and figures that contain both user input and system output (which appears in monospace). For example:

  ```
  display_access_list system_default
  %dev#m1>system>acl>system_default
  w  *.*
  ```
Format for Commands and Requests

Stratus manuals use the following format conventions for documenting commands and requests. (A request is typically a command used within a subsystem, such as analyze_system.) Note that the command and request descriptions do not necessarily include each of the following sections.

name
The name of the command or request is at the top of the first page of the description.

Privileged
This notation appears after the name of a command or request that can be issued only from a privileged process.

Purpose
Explains briefly what the command or request does.

Display Form
Shows the form that is displayed when you type the command or request name followed by -form or when you press the key that performs the DISPLAY FORM function. Each field in the form represents a command or request argument. If an argument has a default value, that value is displayed in the form.

The following table explains the notation used in display forms.

The Notation Used in Display Forms

<table>
<thead>
<tr>
<th>Notation</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Required field with no default value.</td>
<td></td>
</tr>
<tr>
<td>The cursor, which indicates the current position on the screen. For example, the cursor may be positioned on the first character of a value, as in all.</td>
<td></td>
</tr>
<tr>
<td>current_user</td>
<td>The default value is the current user, module, system, or disk. The actual name is displayed in the display form of the command or request.</td>
</tr>
<tr>
<td>current_module</td>
<td></td>
</tr>
<tr>
<td>current_system</td>
<td></td>
</tr>
<tr>
<td>current_disk</td>
<td></td>
</tr>
</tbody>
</table>

Command-Line Form
Shows the syntax of the command or request with its arguments. You can display an online version of the command-line form of a command or request by typing the command or request name followed by -usage.
The following table explains the notation used in command-line forms. In the table, the term *multiple values* refers to explicitly stated separate values, such as two or more object names. Specifying multiple values is *not* the same as specifying a star name. When you specify multiple values, you must separate each value with a space.

**The Notation Used in Command-Line Forms**

<table>
<thead>
<tr>
<th>Notation</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>argument_1</td>
<td>Required argument.</td>
</tr>
<tr>
<td>argument_1...</td>
<td>Required argument for which you can specify multiple values.</td>
</tr>
<tr>
<td>{element_1} {element_2}</td>
<td>Set of arguments that are mutually exclusive; you must specify one of these arguments.</td>
</tr>
<tr>
<td>[argument_1]</td>
<td>Optional argument.</td>
</tr>
<tr>
<td>[argument_1]...</td>
<td>Optional argument for which you can specify multiple values.</td>
</tr>
<tr>
<td>[argument_1] [argument_2]</td>
<td>Set of optional arguments that are mutually exclusive; you can specify only one of these arguments.</td>
</tr>
</tbody>
</table>

**Note:** Dots, brackets, and braces are not literal characters; you should *not* type them. Any list or set of arguments can contain more than two elements. Brackets and braces are sometimes nested.

**Arguments**

Describes the command or request arguments. The following table explains the notation used in argument descriptions.

**The Notation Used in Argument Descriptions**

<table>
<thead>
<tr>
<th>Notation</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>(CYCLE)</td>
<td>This argument has predefined values. In the display form, you display these values in sequence by pressing the key that performs the CYCLE function.</td>
</tr>
<tr>
<td>Required</td>
<td>You cannot issue the command or request without specifying a value for this argument.</td>
</tr>
</tbody>
</table>

If an argument is required but has a default value, it is not labeled *Required* since you do not need to specify it in the command-line form. However, in the display form, a required field must have a value—either the displayed default value or a value that you specify.
Explanation
Explains how to use the command or request and provides supplementary information.

Error Messages
Lists common error messages with a short explanation.

Examples
Illustrates uses of the command or request.

Related Information
Refers you to related information (in this manual or other manuals), including descriptions of commands, subroutines, and requests that you can use with or in place of this command or request.

Getting Help
If you have a technical question about ftServer system hardware or software, try these online resources first:

- **Online documentation at the StrataDOC Web site.** Stratus provides complimentary access to StrataDOC, an online-documentation service that enables you to view, search, download, and print customer documentation. You can access StrataDOC at the following Web site:
  
  [http://stratadoc.stratus.com](http://stratadoc.stratus.com)

- **Online support from Stratus Customer Service.** You can find the latest technical information about an ftServer system in the Stratus Customer Service Portal at the following Web site:
  
  [http://www.stratus.com/go/support](http://www.stratus.com/go/support)

  The Service Portal provides access to Knowledge Base articles for all Stratus product lines. You can locate articles by performing a simple or advanced keyword search, viewing recent articles or top FAQs, or browsing a product and category.

  To log in to the Service Portal, enter your employee user name and password or, if you have not been provided with a login account, click **Register Account.** When registering a new account, ensure that you specify an email address from a company that has a service agreement with Stratus.

  If you cannot resolve your questions with these online self-help resources, and the ftServer system is covered by a service agreement, contact the Customer Assistance Center. To contact the CAC, use the Service Portal to log a support request. Click **Customer Support** and **Add Issue,** and then complete the **Create Issue** form. A member of our Customer Service team will be glad to assist you.
Preface

Commenting on This Manual

You can comment on this manual using one of the following methods. When you submit a comment, be sure to provide the manual’s name and part number, a description of the problem, and the location in the manual where the affected text appears.

- From StrataDOC, click the site feedback link at the bottom of any page. In the pop-up window, answer the questions and click Submit.
- From any email client, send email to comments@stratus.com.
- From the Stratus Customer Service Portal, log on to your account and create a new issue.

Stratus welcomes any corrections and suggestions for improving this manual.
## Contents

1. OpenVOS Command Functions
   - (abs) .......................................................... 1-4
   - (access) ...................................................... 1-5
   - (after) ....................................................... 1-7
   - (ask) .......................................................... 1-8
   - (before) ..................................................... 1-10
   - (break) ....................................................... 1-11
   - (byte) ........................................................ 1-12
   - (calc) ......................................................... 1-13
   - (ceil) ....................................................... 1-16
   - (concat) .................................................... 1-18
   - (command_status) ........................................ 1-19
   - (copy) ....................................................... 1-20
   - (count) ..................................................... 1-21
   - (current_dir) ............................................. 1-22
   - (current_module) ......................................... 1-23
   - (date) ....................................................... 1-24
   - (date_time) ............................................... 1-27
   - (decimal) .................................................. 1-29
   - (directory_name) ......................................... 1-30
   - (end_of_file) ............................................. 1-31
   - (exists) .................................................... 1-32
   - (extended_names) ....................................... 1-34
   - (extended_names_version) ............................... 1-35
   - (file_info) ............................................... 1-37
   - (floor) ..................................................... 1-39
   - (given) ..................................................... 1-40
   - (group_name) ............................................. 1-41
   - (has_access) .............................................. 1-42
   - (hexadecimal) .......................................... 1-44
   - (home_dir) ................................................ 1-45
   - (index) ..................................................... 1-46
   - (iso_date) ............................................... 1-47
   - (iso_date_time) ......................................... 1-50
   - (language_name) ....................................... 1-53
   - (length) ................................................... 1-54
   - (lock_type) ............................................... 1-55
   - (locked) ................................................... 1-56
   - (ltrim) ..................................................... 1-57
   - (master_disk) ............................................ 1-58
   - (max) ...................................................... 1-59
## Contents

(message). ................................................................. 1-60
(min) ................................................................. 1-60
(mod) ................................................................. 1-60
(module_info) ...................................................... 1-60
(module_name) ...................................................... 1-60
(name_string) ...................................................... 1-60
(object_name) ...................................................... 1-60
(online) ............................................................... 1-69
(path_name) ......................................................... 1-69
(person_name) ..................................................... 1-69
(posix_path) ........................................................ 1-69
(process_dir) ....................................................... 1-69
(process_info) ..................................................... 1-69
(process_type) ..................................................... 1-69
(quote) ............................................................... 1-79
(rank) ................................................................. 1-80
(referencing_dir) ................................................... 1-80
(reverse) ............................................................ 1-80
(rtrim) ................................................................. 1-80
(search) ............................................................... 1-80
(software_purchased) ............................................. 1-80
(string) ............................................................... 1-80
(substitute) ......................................................... 1-80
(substr) .............................................................. 1-80
(system_info) ....................................................... 1-80
(system_name) ...................................................... 1-80
(terminal_info) .................................................... 1-80
(terminal_name) ................................................... 1-80
(time) ................................................................. 1-80
(translate) .......................................................... 1-80
(trunc) ............................................................... 1-80
(unique_string) .................................................... 1-80
(unquote) ............................................................ 1-80
(user_name) ........................................................ 1-80
(verify) ............................................................. 1-80
(vos_path) .......................................................... 1-80
(where_path) ....................................................... 1-80
Date and Time Keywords ............................................... 1-110

### 2. OpenVOS User and Programming Commands

- add_entry_names ............................................. 1-1
- add_library_path ........................................... 1-2
- add_profile .................................................... 1-2
- analyze_pc_samples ....................................... 1-2
- attach_default_output .................................. 1-2
- attach_port .................................................... 1-2
- batch ............................................................ 1-2
- bind ............................................................. 1-2
- break_process ............................................... 1-2
- bundle .......................................................... 1-2
- c ................................................................. 1-3
- c_preprocess ................................................ 1-3
- call_thru ....................................................... 1-3
cancel_batch_requests ........................................... 1-106
cancel_device_reservation .................................... 1-108
cancel_print_requests ......................................... 1-109
cc ........................................................................ 1-111
change_current_dir .............................................. 1-132
change_password ................................................. 1-134
check_posix ........................................................ 1-137
close_dir .......................................................... 1-139
close_file .......................................................... 1-142
cobol ................................................................. 1-144
compare_dirs ...................................................... 1-157
compare_files ..................................................... 1-160
consolidate_dir ................................................... 1-169
cvt_fixed_to_stream ............................................ 1-200
cvt_stream_to_fixed ............................................ 1-207
debug ............................................................... 1-241
decrypt ............................................................ 1-272
decrypt ............................................................ 1-274
delete_dir .......................................................... 1-276
delete_file ........................................................ 1-279
delete_index ...................................................... 1-282
delete_library_path ............................................. 1-284
detach_default_output .......................................... 1-287
detach_port ....................................................... 1-288
dismount_tape .................................................... 1-291
display .............................................................. 1-296
display_access .................................................... 1-299
display_access_list ............................................. 1-301
display_batch_status .......................................... 1-303
display_current_dir ............................................ 1-304
display_current_module ....................................... 1-305
display_date_time .............................................. 1-307
display_default_access_list .................................... 1-309
display_default_open_options ................................ 1-311
display_device_info ........................................... 1-319
display_dir_status .............................................. 1-321
display_disk_info .............................................. 1-327
display_disk_usage ............................................ 1-329
display_error ..................................................... 1-331
display_file ..................................................... 1-337
display_file_status ............................................ 1-337

OpenVOS Commands Reference Manual (R098)  xi
Contents

display_line .................................................. 1-349
display_notices ................................................ 1-351
display_object_module_info .............................. 1-352
display_open_options ......................................... 1-358
display_print_defaults ..................................... 1-361
display_print_status ....................................... 1-363
display_program_module .................................. 1-366
display_system_usage ..................................... 1-382
display_tape_params ....................................... 1-388
display_terminal_parameters ....................... 1-392
display_usb_info ........................................... 1-393
dump_file ..................................................... 1-394
dump_record .................................................. 1-396
dump_tape ....................................................... 1-397
edit ............................................................... 1-400
edit_form ....................................................... 1-405
earns ............................................................ 1-411
encode_vos_file ............................................ 1-417
encrypt ........................................................ 1-420
enforce_region_locks .................................... 1-423
fortran .......................................................... 1-425
get_external_variable .................................... 1-437
give_access .................................................. 1-439
give_default_access ...................................... 1-444
handle_sig_dfl ............................................. 1-446
harvest_pc_samples ...................................... 1-448
help ............................................................ 1-456
kill .............................................................. 1-459
ldd .............................................................. 1-461
line_edit ..................................................... 1-462
link ............................................................ 1-465
link_dirs ...................................................... 1-468
list ............................................................ 1-471
list_batch_requests ..................................... 1-477
list_devices ............................................... 1-481
list_dynamic_dependencies .......................... 1-483
list_gateways .............................................. 1-484
list_library_paths ..................................... 1-485
list_modules ................................................ 1-487
list_port_attachments ................................. 1-489
list_print_requests ...................................... 1-494
list_process_cmd_limits ............................. 1-497
list_save_tape ............................................ 1-500
list_systems ................................................. 1-502
list_tape ..................................................... 1-504
list_terminal_types ..................................... 1-506
list_users ................................................... 1-508
locate_expandable_dirs ............................... 1-517
locate_files ................................................ 1-519
locate_indexed_files .................................. 1-521
locate_large_dirs ....................................... 1-526
locate_large_files ..................................... 1-528
locate_stream_files ............................... 1-531
login .......................................................... 1-534
<table>
<thead>
<tr>
<th>Command</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>logout</td>
<td>1-538</td>
</tr>
<tr>
<td>mount_tape</td>
<td>1-540</td>
</tr>
<tr>
<td>move_device_reservation</td>
<td>1-550</td>
</tr>
<tr>
<td>move_dir</td>
<td>1-552</td>
</tr>
<tr>
<td>move_file</td>
<td>1-558</td>
</tr>
<tr>
<td>mp_debug</td>
<td>1-564</td>
</tr>
<tr>
<td>nls_edit_form</td>
<td>1-569</td>
</tr>
<tr>
<td>pascal</td>
<td>1-576</td>
</tr>
<tr>
<td>pl1</td>
<td>1-588</td>
</tr>
<tr>
<td>position_tape</td>
<td>1-601</td>
</tr>
<tr>
<td>posixpath</td>
<td>1-604</td>
</tr>
<tr>
<td>preprocess_file</td>
<td>1-606</td>
</tr>
<tr>
<td>print</td>
<td>1-614</td>
</tr>
<tr>
<td>profile</td>
<td>1-622</td>
</tr>
<tr>
<td>propagate_access</td>
<td>1-634</td>
</tr>
<tr>
<td>read_tape</td>
<td>1-637</td>
</tr>
<tr>
<td>ready</td>
<td>1-640</td>
</tr>
<tr>
<td>remove_access</td>
<td>1-642</td>
</tr>
<tr>
<td>remove_default_access</td>
<td>1-644</td>
</tr>
<tr>
<td>rename</td>
<td>1-646</td>
</tr>
<tr>
<td>reserve_device</td>
<td>1-649</td>
</tr>
<tr>
<td>reset_eof</td>
<td>1-650</td>
</tr>
<tr>
<td>restore_object</td>
<td>1-653</td>
</tr>
<tr>
<td>save_object</td>
<td>1-658</td>
</tr>
<tr>
<td>send_message</td>
<td>1-661</td>
</tr>
<tr>
<td>set</td>
<td>1-664</td>
</tr>
<tr>
<td>set_cpu_time_limit</td>
<td>1-665</td>
</tr>
<tr>
<td>set_default_open_options</td>
<td>1-667</td>
</tr>
<tr>
<td>set_dir_limits</td>
<td>1-669</td>
</tr>
<tr>
<td>set_dir_type</td>
<td>1-672</td>
</tr>
<tr>
<td>set_expiration_date</td>
<td>1-675</td>
</tr>
<tr>
<td>set_external_variable</td>
<td>1-677</td>
</tr>
<tr>
<td>set_file_allocation</td>
<td>1-679</td>
</tr>
<tr>
<td>set_implicit_locking</td>
<td>1-681</td>
</tr>
<tr>
<td>set_index_flags</td>
<td>1-683</td>
</tr>
<tr>
<td>set_language</td>
<td>1-684</td>
</tr>
<tr>
<td>set_library_paths</td>
<td>1-686</td>
</tr>
<tr>
<td>set_line_wrap_width</td>
<td>1-689</td>
</tr>
<tr>
<td>set_open_options</td>
<td>1-691</td>
</tr>
<tr>
<td>set_owner_access</td>
<td>1-694</td>
</tr>
<tr>
<td>set_pipe_file</td>
<td>1-697</td>
</tr>
<tr>
<td>set_priority</td>
<td>1-699</td>
</tr>
<tr>
<td>set_ram_file</td>
<td>1-701</td>
</tr>
<tr>
<td>set_ready</td>
<td>1-703</td>
</tr>
<tr>
<td>set_safety_switch</td>
<td>1-705</td>
</tr>
<tr>
<td>set_second_tape</td>
<td>1-707</td>
</tr>
<tr>
<td>set_tape_drive_params</td>
<td>1-709</td>
</tr>
<tr>
<td>set_tape_file_params</td>
<td>1-712</td>
</tr>
<tr>
<td>set_tape_mount_params</td>
<td>1-717</td>
</tr>
<tr>
<td>set_terminal_parameters</td>
<td>1-721</td>
</tr>
<tr>
<td>set_text_file</td>
<td>1-731</td>
</tr>
<tr>
<td>set_time_zone</td>
<td>1-734</td>
</tr>
<tr>
<td>sleep, vsleep</td>
<td>1-738</td>
</tr>
<tr>
<td>sort, vsort</td>
<td>1-740</td>
</tr>
</tbody>
</table>
start_logging .......................................................... 1-751
start_process .......................................................... 1-757
stop_logging ............................................................. 1-760
stop_process ............................................................. 1-761
tail_file ................................................................. 1-764
temacs .............................................................. 1-767
text_data_merge .......................................................... 1-770
translate_links ............................................................ 1-778
truncate_file ............................................................. 1-781
unbundle .............................................................. 1-783
unlink .............................................................. 1-786
update_batch_requests .................................................. 1-788
update_print_requests ................................................ 1-792
update_process_cmd_limits ............................................ 1-799
use_abbreviations ..................................................... 1-812
use_message_file ...................................................... 1-814
vcc ........................................................ 1-816
vemacs .......................................................... 1-821
verify_posix_access .................................................. 1-824
verify_save ............................................................ 1-829
verify_system_access .................................................. 1-832
vospath ............................................................ 1-834
walk_dir ............................................................. 1-836
where_command .......................................................... 1-839
where_path ............................................................ 1-841
who_locked .............................................................. 1-843
write_tape .............................................................. 1-845

Appendix A. Setting and Displaying Tape Parameter Values ........................ A-1

Appendix B. General OpenVOS Software Limits and Numerical Definitions ....... B-1

Appendix C. Reducing Program Module Size When Using Shared Virtual Memory Databases .......................... C-1
   Using create_data_object to Organize Virtual Memory ..................... C-2
   Using Bind Directives to Organize Virtual Memory ......................... C-3
   Using Bind Directives to Create a Shared Virtual Memory Database .... C-4
      Making and Checking the Calculations .................................... C-6
      Reacting to Application and Compiler Changes ......................... C-6
      Using More Than One Shared Virtual Memory Region .................... C-7
   Example of Binder Control File Changes ................................... C-7
      Binder Control Files Using a Created Data Object ...................... C-7
      Changing Binder Control Files to Use SVMR .......................... C-8
      Using high_water_mark to Adjust the Size Reserved for the SVMR .... C-9

Index ............................................................. Index-1
Figures
Figure 2-1. Analyzing Program Modules in the Kernel and User Address Space . . . . .
Figure 2-2. Example of Standard Deviation . . . . . . . . . . . . . . . . . . . .
Figure 2-3. Process Address Space and Related bind Arguments . . . . . . . . . . .
Figure 2-4. ftServer Modules: Stack and Fence Size for Programs with Multiple Static Tasks
Figure 2-5. The Program Counter Sampling System . . . . . . . . . . . . . . . .
Figure 2-6. Default Process Address Space on ftServer Modules . . . . . . . . . . .
Figure 2-7. Effects of Setting Process Initial, Current, and Maximum Command Limits . .
Figure 2-8. Phases of the vcc Command . . . . . . . . . . . . . . . . . . . . .
Figure A-1. Duration of User and Default Values . . . . . . . . . . . . . . . . .
Figure C-1. Shared Virtual Memory Database in a Program Module . . . . . . . . . .
Figure C-2. Shared Virtual Memory Database Not in a Program Module . . . . . . . .

.
.
.
.
.
.
.
.
.
.
.

.
.
.
.
.
.
.
.
.
.
.

.
.
.
.

1-19
1-31
1-58
1-60
1-452
1-806
1-810
1-818
. A-3
. C-3
. C-4

OpenVOS Commands Reference Manual (R098)

xv


Tables

Table 1-1. Arithmetic, Logical, and String Expression Operators ................................... 1-14
Table 1-3. Scalar Numeric Values in (system_info) Output ................................................ 1-92
Table 1-5. Date and Time Input Keywords ........................................................................... 1-110
Table 2-1. Line, Module, and Function Statistics Columns of analyze_pc_samples ............. 1-29
Table 2-2. Module Summary Statistics Columns of analyze_pc_samples ............................ 1-31
Table 2-3. Command-Line Options of the bind Command .................................................... 1-50
Table 2-4. Summary of the Behavior of -extended_names/extended_names Values ............ 1-63
Table 2-5. Values for the options Directive of the bind Command ................................. 1-68
Table 2-6. Possible Destination File Names After Bundling ................................................ 1-82
Table 2-7. Predefined Preprocessor Variables .............................................................. 1-92
Table 2-8. Arguments Affecting Optimization Level .......................................................... 1-96
Table 2-9. cc Command: Short Options .............................................................................. 1-113
Table 2-10. cc Command: Optimization-Related Options and Arguments ......................... 1-122
Table 2-11. Predefined Preprocessor Variables .............................................................. 1-126
Table 2-12. Predefined Preprocessor Variables .............................................................. 1-152
Table 2-13. Extent-based Files: Advantages and Disadvantages ......................................... 1-215
Table 2-14. Block Environments ......................................................................................... 1-245
Table 2-15. Modifiers and Modifier Abbreviations ........................................................... 1-263
Table 2-16. Regions and Region Abbreviations ............................................................... 1-263
Table 2-17. Example Memory References ......................................................................... 1-264
Table 2-18. Example Relational Expressions ..................................................................... 1-267
Table 2-19. Requests That Are Modified in Machine Mode ............................................... 1-268
Table 2-20. Values for the endianSpecifier Argument ...................................................... 1-270
Table 2-21. Predefined Preprocessor Variables .............................................................. 1-431
Table 2-22. Commands That Measure Performance ......................................................... 1-450
Table 2-23. Data Items Displayed by the list_users Command ........................................ 1-512
Table 2-24. Matching Access Rights ................................................................................. 1-545
Table 2-25. Predefined Preprocessor Variables .............................................................. 1-583
Table 2-26. Predefined Preprocessor Variables .............................................................. 1-595
Table 2-27. Optimization-Related Arguments ................................................................... 1-597
Table 2-28. Preprocessing Operators and Lead-In Characters ........................................... 1-608
Table 2-29. Maximum Offset Values for DAE Files ............................................................ 1-652
Table 2-30. File Formats .................................................................................................. 1-714
Table 2-31. Tape File Format Names ................................................................................. 1-716
Table 2-32. ANSI and IBM Tape Label Fields .................................................................... 1-719
Table 2-33. Time Zones Supported in OpenVOS .............................................................. 1-735
Table 2-34. Embedded Replacement References ............................................................. 1-773
Table 2-35. Suffixes Appended by the bundle Command .................................................. 1-784
Table 2-36. Default Values on an ftServer Module .................................. 1-808
Table 2-37. The vcc Command Options ...................................................... 1-816
Table 2-38. Differences between the vcc and cc Commands .......................... 1-819
Table A-1. Setting Tape Parameters ........................................................... A-1
Table B-1. General OpenVOS Software Limits ............................................. B-1
Table B-2. OpenVOS Numerical Definitions ................................................. B-7
Tables
Chapter 1:
OpenVOS Command Functions

A \textit{command function} is a self-contained function, enclosed in parentheses, that you can use as an argument in a command line.

\textbf{Note:} A command line in a command macro can be up to 32,767 characters long. A command line at command level can be up to 300 characters long. Individual words (including quoted strings) can be up to 256 characters long. The output of a command function is no longer than 256 characters.

Before executing the rest of the command line in which a command function appears, the command processor evaluates the command function. The resulting value replaces the command function and its parentheses in the command line. The value returned by the command function has a 256-character limit. The command processor treats that resulting value as a string enclosed in apostrophes. For example, the command function \texttt{(time)} is replaced by the current time in the following command line.

\begin{verbatim}
display_line (time)
\end{verbatim}

In some cases, you may want the command function evaluated at a future time, rather than when the current command line executes. An example is using the \texttt{(current\_dir)} or \texttt{(referencing\_dir)} command function with a library path command. To prevent evaluation of a command function at the time you use it in a command line, enclose it in apostrophes.

The command processor expands abbreviations in the command line before evaluating command functions, because there may be command functions in the abbreviations.
This chapter alphabetically lists and describes in detail all of the available command functions, which follow.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S$, $S_1$, $S_2$, ..., $S_n$, $R$</td>
<td>A character string</td>
</tr>
<tr>
<td>$C$</td>
<td>A character or set of characters</td>
</tr>
<tr>
<td>$I$</td>
<td>An integer</td>
</tr>
<tr>
<td>$N$, $N_1$, $N_2$, ..., $N_n$</td>
<td>A number</td>
</tr>
</tbody>
</table>

A character string is an ordered set of characters. Positions in a character string are counted starting with the leftmost character. A substring is a string of any length that occurs within a longer string. An initial substring is a substring whose first character is also the first character in the containing string. Similarly, a final substring is a substring whose last character is also the last character in the containing string. For example, if there is the character string `abcd`, then `a` and `abcd` are two initial substrings of the string; likewise, `def` and `ef` are two final substrings of the full character string.
You must enclose character string arguments within apostrophes if the string contains one or more spaces, semicolons, or parentheses. If you need an apostrophe character within a character string, type two apostrophes (‘’).

See the tables in the Preface for an explanation of the other notation used in documenting the command functions.

Note that many of the examples of command functions in this chapter make the following assumptions:

- The current system is %s1.
- The current module is #m2 (path name %s1#m2).
- The current master disk is #d01 (path name %s1#d01).
- The current user is Smith in the group Sales (user name Smith.Sales).
- The current directory is %s1#d01>Sales>Smith (the current user’s home directory).
(abs)

Purpose
This command function returns the absolute value of a number.

Syntax
(abs $N$)

Explanation
The (abs) command function returns the absolute value of $N$. $N$ can be an integer or a floating point number.

Example
For example, if you specify (abs -2), the command function returns the value 2.

Related Information
See also the descriptions of the (calc), (ceil), (floor), (max), (min), and (mod) command functions.
**Purpose**

This command function returns a code specifying the access rights of a user to an object.

**Syntax**

\[
\text{(access path_name [user_name])}
\]

**Explanation**

The `(access)` command function returns a code specifying the access rights of a user to the object specified by `path_name`. If you specify a value for `user_name`, it must be of the form `person_name.group_name` (for example, `Smith.Sales`). If you omit the user name, the access rights returned are your own.

The possible codes the `(access)` command function returns, and their corresponding meanings for files and directories, are as follows:

<table>
<thead>
<tr>
<th>Directories</th>
<th>Files</th>
</tr>
</thead>
<tbody>
<tr>
<td>m for modify</td>
<td>w for write</td>
</tr>
<tr>
<td>s for status</td>
<td>r for read</td>
</tr>
<tr>
<td>n for null</td>
<td>e for execute</td>
</tr>
<tr>
<td></td>
<td>n for null</td>
</tr>
</tbody>
</table>

If none of the access rights defined for the object specified by `path_name` apply to the user specified by `user_name`, the `(access)` command function returns the code `u` (for undefined), which is effectively null access.

If the object specified by `path_name` does not exist, the returned value is `n`. Also, if the value for `user_name` does not specify a particular user by having the form `person_name.group_name`, the returned value is `n`.

**Example**

For example, if you specify `(access %s#d01>Sales>Smith Smith.Sales)`, the command function returns the value `m`. The user `Smith.Sales` has modify access rights to his or her home directory.
Related Information

See also the descriptions of the (group_name), (has_access), and (person_name) command functions.
Purpose

This command function returns the final characters in a string following a substring in the string.

Syntax

(after S1 S2)

Explanation

The (after) command function returns the final substring of S1 that follows the substring S2.

Example

For example, if you specify (after aabbcc.ddeeff c.d), the command function returns the value deeff.

Related Information

See also the descriptions of the (before), (break), (concat), (copy), (count), (index), (length), (ltrim), (quote), (reverse), (rtrim), (search), (string), (substitute), (substr), (translate), (trunc), (unique_string), (unquote), and (verify) command functions.
(ask)

Purpose

This command function returns a user-supplied input string in response to a specified query.

Syntax

(ask [prompt [response_qualifier]] [-no_echo])

Explanation

The (ask) command function:

1. Returns a user-supplied input string.

2. Writes the string specified by prompt and any string specified by response_qualifier to the terminal_output port (with no trailing new line).

3. Reads the user-supplied response string from the default_input port and displays it on a new line.

For example, the command function (ask 'Do you want to delete x?') first displays the prompt 'Do you want to delete x?'. On the same line you could respond by entering yes. Therefore, the value of the function is yes. Since no value for response_qualifier is specified, you could respond I doubt it. The value of the function would then be I doubt it.

If the string you specify for prompt contains spaces, semicolons, or parentheses that are not part of a command function, the string must be enclosed in apostrophes ('). If you omit a value for prompt, the null string is written to the terminal_output port.

You can specify a string value for response_qualifier to restrict the type of response the command function will accept.

To specify that the response must be one of a particular set of responses, each element of response_qualifier must have the form:

\[(S1 \ [S2 \ [S3] \ldots]) = R\]

In this case, you must enclose the entire response_qualifier string in apostrophes ('), and separate each Sn from the next with a comma. If you specify any Sn value as a response,
the command function returns \( R \). For example, to restrict the response allowed to either \textit{yes} or \textit{no}, the value of \textit{response_qualifier} might be:

\[
(\text{yes}, y) = \text{yes} \quad (\text{no}, n) = \text{no}
\]

If the response is \textit{yes} or \textit{y}, the returned value is \textit{yes}; if the response is \textit{no} or \textit{n}, the returned value is \textit{no}.

To specify that the response must have a particular data type, the value for \textit{response_qualifier} can be any of the following:

\[
\text{date\_time} \\
\text{module\_name} \\
\text{device\_name} \\
\text{system\_name} \\
\text{name} \\
\text{number} \\
\text{pathname} \ [\text{suffix}] \\
\text{string} \\
\text{user\_name}
\]

In any of these cases, \texttt{ask} displays the string specified by \textit{prompt} and a form of the string specified by \textit{response_qualifier}; the response entered must be of the type specified, or an error message results.

If you select the \texttt{-no\_echo} option, terminal input is not echoed to the screen, and the response to the prompt is suppressed.

\textbf{Example}

Consider this form of the \texttt{(ask)} command function:

\[
(\text{ask} \ '\text{Do you want to delete x?}?' \ (\text{yes}, y) = \text{yes} \ (\text{no}, n) = \text{no}')
\]

The prompt displayed is \texttt{Do you want to delete x? (yes, no)}. If the response is \texttt{yes} or \texttt{y}, the returned value is \texttt{yes}; if the response is \texttt{no} or \texttt{n}, the returned value is \texttt{no}.

The value of \texttt{(ask 'What is your name?' name)} might be \texttt{John}.  

\( (\text{ask}) \)
(before)

Purpose
This command function returns initial characters in a string that precede a substring in the string.

Syntax
(before S1 S2)

Explanation
The (before) command function returns the initial substring of S1 that precedes the substring S2.

Example
For example, if you specify (before aabbcc.ddeeff c.d), the command function returns the value aabbc.

Related Information
See also the descriptions of the (after), (break), (concat), (copy), (count), (index), (length), (ltrim), (quote), (reverse), (rtrim), (search), (string), (substitute), (substr), (translate), (trunc), (unique_string), (unquote), and (verify) command functions.
Purpose

This command function returns the length of the initial string that precedes the first instance in the string of a specified character.

Syntax

\[(\text{break } S \ C)\]

Explanation

The \(\text{(break)}\) command function returns the length of the initial substring of \(S\) that precedes the first instance in \(S\) of any character in the set \(C\).

Example

For example, if you specify \(\text{(break aabbcc.ddeeff ',.;:')}\), the command function returns the value 6.

Related Information

See also the descriptions of the \(\text{(after)}, \text{(before)}, \text{(concat)}, \text{(copy)}, \text{(count)}, \text{(index)}, \text{(length)}, \text{(ltrim)}, \text{(quote)}, \text{(reverse)}, \text{(rtrim)}, \text{(search)}, \text{(string)}, \text{(substitute)}, \text{(substr)}, \text{(translate)}, \text{(trunc)}, \text{(unique_string)}, \text{(unquote)}, \text{ and \(\text{(verify)}\)}\) command functions.
Purpose
This command function returns the ASCII character that corresponds to an integer in the
ASCII collating sequence.

Syntax
(byte I)

Explanation
The (byte) command function returns the ASCII character that corresponds to the integer
I in the ASCII collating sequence. The (byte) command function is the inverse of the
(rank) command function.

Example
For example, if you specify (byte 97), the command function returns the value a.

Related Information
See also the description of the (rank) command function.
Purpose

This command function returns the value that results from the command processor’s evaluation of the specified expression.

Syntax

(calc expression)

Explanation

The (calc) command function returns the value that results from the command processor’s evaluation of a specified expression. An expression is a series of one or more operands and zero or more operators that can be evaluated. Operands are the terms, or elements, of the expression to which the operators are applied. An operand can be a character string or a number (a number is any string that can be converted to a number). Operators are symbols that represent the actions to be performed on the operands. An infix operator appears between two operands. A prefix operator appears before an operand.

All operators associate left to right, except for the prefix of positive and negative signs and logical negation. When you write an expression, observe the following rules.

- You must enter spaces before and after the operators so that the operating system can distinguish between use of the symbols as operators and other uses of the symbols.

- You can use parentheses to change the order in which operators are evaluated. When you use parentheses to associate terms, you must enclose them in apostrophes.

The expression can be an arithmetic, logical, or string expression. An arithmetic expression contains an arithmetic operator and operands that can be evaluated and reduced to a single numeric value. A logical expression contains a logical operator and operands that can be evaluated and reduced to either 0 when the expression is false or 1 when it is true. A string expression contains the string operator (||) and operands that can be evaluated and reduced to a single character-string value.

Table 1-1 shows the operators that an expression can contain in order of decreasing precedence. Precedence is the order in which the operators in an expression are processed; operators with higher precedence are processed before operators with lower precedence.

Table 1-1 also shows which operators can be used in arithmetic, logical, and string expressions. Note that the equality operators are categorized only as logical operators. These operators are used with numeric and character-string operands; however, they are logical expressions since the resulting value is always 0 or 1.
Chapter 1: OpenVOS Command Functions

Note that the broken vertical bars that represent the concatenation operator and the logical or operator appear as solid vertical bars on some terminals and keyboards.

The arithmetic operations are performed with floating-point arithmetic to a precision of 15 decimal digits. When two terms are compared, they are converted to numbers if possible. If both terms cannot be converted to numbers, they are compared as character strings.

All calculations and conversions are performed by PL/I operators. See the OpenVOS PL/I Language Manual (R009) for the evaluation and conversion rules.

If you want to evaluate two operands and a string operator as a string expression that \( (\text{calc}) \) would otherwise evaluate as an arithmetic or logical expression, use the \( (\text{quote}) \) command function. For example:

\[
(\text{calc} \ (\text{quote} \ 1) \ || \ (\text{quote} \ 2))
\]

Similarly, in a logical or logical and a string expression, if you want to use one or more characters in an operand that \( (\text{calc}) \) would otherwise interpret as an operator, you can specify the operand as an argument to the \( (\text{quote}) \) command function. For example:

\[
(\text{calc} \ (\text{quote} \ *) \ < \ (\text{quote} \ /))
\]
Operators are evaluated from left to right, except for exponentiation and prefix operators, which are evaluated from right to left. For example, note how the expression $12 / 3 * 4$ produces a different result when it is evaluated from right to left instead of from left to right. (Division and multiplication operands have equal precedence.)

<table>
<thead>
<tr>
<th>Expression</th>
<th>Order of Evaluation</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$12 / 3 * 4$</td>
<td>Left to right</td>
<td>16</td>
</tr>
<tr>
<td>$12 / 3 * 4$</td>
<td>Right to left</td>
<td>1</td>
</tr>
</tbody>
</table>

**Examples**

Note how the result differs when parentheses are used to alter the precedence of multiplication over addition. For example, if you specify `(calc 3 + 4 * 7)`, the command function returns the value 31. If you specify `(calc ' ( ' 3 + 4 ' ) ' * 7)`, the command function returns the value 49.

The following examples illustrate the use of `(calc)` in an expression in a command macro:

```
&set a (calc (length &file&) - 4)
&set a (calc '( ' 2 + 2 ' ) ' / ' ( ' 3 + 2 ' ) ' )
&set a ' ( ' 2 + 2 ' ) ' / ' ( ' 3 + 2 ' ) ' 
&set count &count& + 1
&if &count& > 5
&then &goto exit
```

Note that `&set` statements imply `(calc)`. Some of the preceding examples use `(calc)` for clarity. For more information about the `&set` macro statement, see the *OpenVOS Commands User’s Guide* (R089).

**Related Information**

See also the descriptions of the `(abs)`, `(ceil)`, `(floor)`, `(max)`, `(min)`, and `(mod)` command functions.
(ceil)

**Purpose**

This command function returns the smallest integer greater than or equal to a specified number.

**Syntax**

\[(ceil \, N)\]

**Explanation**

The (ceil) command function returns the smallest integer greater than or equal to \( N \).

**Examples**

For example, if you specify \((ceil \, 1.5)\), the command function returns the value 2. If you specify \((ceil \, -1.5)\), the command function returns the value -1.

**Related Information**

See also the descriptions of the \((abs), (calc), (floor), (max), (min), and (mod)\) command functions.
(command_status)

Purpose
This command function returns the status code of the most recently executed command.

Syntax
(command_status)

Explanation
The (command_status) command function returns the status code of the most recently executed command. When a command begins to execute, the operating system sets the value of the status code to 0. If the command executes normally without errors, the value of the status code does not change. If an error occurs, the operating system sets the value of the status code to the value returned by the OpenVOS subroutine s$error or s$stop_program.

Example
The following example illustrates the use of the (command_status) command function in a command.

    display_error (command_status)

Related Information
See also the description of the (message) command function. For information about the values returned by the s$error and s$stop_program subroutines, see the OpenVOS Subroutines manuals.
(concat)

Purpose
This command function combines two or more strings.

Syntax
(concat S1 ...Sn)

Explanation
The (concat) command function returns the strings $S_1$ through $S_n$ combined into one string with all spaces between the components removed.

Example
For example, if you specify (concat aabbcc . ddeeff), the command function returns the value aabbcc.ddeeff.

Related Information
See also the descriptions of the (after), (before), (break), (copy), (count), (index), (length), (ltrim), (quote), (reverse), (rtrim), (search), (string), (substitute), (substr), (translate), (trunc), (unique_string), (unquote), and (verify) command functions.
Purpose
This command function returns the records in a file with each record separated by a space.

Syntax
\[
\text{(contents } \text{path\_name} \text{[line\_number] [open\_status]})
\]

Explanation
The (contents) command function returns the records of the file specified by path\_name, with each record separated from the next by a space. The command function can return up to 256 characters from a file. The value of path\_name can be a relative path name. The argument line\_number is an option to specify the number of the line you want to display.

The value of open\_status can be either -hold or -close. This argument can be used only within a command macro. The -hold option keeps the file open until the command macro terminates or until the macro processor encounters the -close option.

Examples
The following examples use the file >Sales>Smith>reminders, which contains these two lines:

1. Write the sales report.
2. Hire a temporary typist.

In the first example, the (contents) command function returns all of the records in the file.

```
ready: display_line (contents >Sales>Smith>reminders)
1. Write the sales report.
2. Hire a temporary typist.
```

In the second example, the (contents) command function returns the second record in the file.

```
ready: display_line (contents >Sales>Smith>reminders 2)
2. Hire a temporary typist.
```

Related Information
See also the descriptions of the (end\_of\_file), (exists), and (file\_info) command functions.
Purpose
This command function returns a specified string copied a specified number of times.

Syntax
(copy $ S I$)

Explanation
The (copy) command function returns the string $S$, copied $I$ times. The integer $I$ can be expanded as a binary, octal, decimal, or hexadecimal integer.

Example
For example, if you specify (copy ba 4), the command function returns the value babababa.

Related Information
See also the descriptions of the (after), (before), (break), (concat), (count), (index), (length), (ltrim), (quote), (reverse), (rtrim), (search), (string), (substitute), (substr), (translate), (trunc), (unique_string), (unquote), and (verify) command functions.
(count)

Purpose
This command function returns the length of the longest initial substring in a string that consists of a specified set of characters.

Syntax
(count S C)

Explanation
The (count) command function returns the length of the longest initial substring of S consisting entirely of characters in the set C.

Example
For example, if you specify (count aabbcc.ddeeff af), the command function returns the value 2.

Related Information
See also the descriptions of the (after), (before), (break), (concat), (copy), (index), (length), (ltrim), (quote), (reverse), (rtrim), (search), (string), (substitute), (substr), (translate), (trunc), (unique_string), (unquote), and (verify) command functions.
(current_dir)

**Purpose**
This command function returns the full path name of your current directory.

**Syntax**

```
(current_dir)
```

**Explanation**
The (current_dir) command function returns the full path name of your current directory.

**Example**
The following example illustrates the use of the (current_dir) command function in a command.

```
ready: display_dir_status (current_dir)
>Sales>Smith>prospects
```

**Related Information**
See also the descriptions of the (directory_name), (home_dir), (object_name), (path_name), (referencing_dir), and (where_path) command functions.
Purpose
This command function returns the name of your current module.

Syntax

```
(current_module)
```

Explanation
The `current_module` command function returns the name of your current module.

Example
The following example illustrates the use of the `current_module` command function in a command.

```
ready: display_line (current_module)
%s1#m2
```

Related Information
See also the descriptions of the `(module_info)`, `(master_disk)`, `(module_name)`, `(online)`, and `(system_name)` command functions.
(date)

Purpose
This command function returns a date.

Syntax
(date [date_string][ -long][ -standard])

Explanation
The (date) command function returns a date in the form yy-mm-dd. The date_string argument represents a date to be returned; if you omit it, the (date) command function returns the current date. The date_string must always be in the form defined in the system>configuration>languages.tin file. The date_string argument can use any of the following keywords. These keywords are described in Table 1-5 later in this chapter.

absolute_date
coming
relative_terms

A date in a date_string consists of one, two, or three tokens. A token can be either a numeric string or an alphabetic string matching one of the words defined in the system>configuration>languages.tin file. The tokens can be separated by delimiter characters such as spaces, periods (.), hyphens (-), or slashes (/). No delimiter characters are needed if the month is represented by a character string or you use the ISO/ANSI date formats. In a three-token date with two delimiter characters, at least one delimiter character must be a space or the two delimiter characters must be the same. Note that in two- or three-token dates, you can precede a year token with a comma (,).

Valid token formats are listed below:

- month name tokens can be strings in the form specified in system>configuration>languages.tin
- one or two digit tokens are a month, day, or year
- two digit tokens greater than or equal to 80 are a year from 1980 to 1999
- three digit tokens from 1 to 366 are a day in the year
- four digit tokens are a year
six digit tokens in the form `yymmdd` are ISO/ANSI dates

eight digit tokens in the form `yyyyymmdd` are ISO/ANSI dates

weekday name tokens in the form specified in `>system>configuration>languages.tin`

Years must be in the range 1980 to 2048.

The following are valid single token dates.

- an ISO/ANSI string of six or eight digits
- a month name, given that the day is assumed to be the first of the month and the year is the current year
- a four-digit year, given that the day is assumed to be January 1
- a two-digit year greater than or equal to 80, given that the day is assumed to be January 1
- a weekday name

Numbers less than 80 and three-digit numbers are not valid single token dates.

The following are valid two token dates.

- a year followed by three-digit day of the year
- a month and year, or year and month, given that the day is assumed to be the first of month
- a year and a two-digit number, or two-digit number and year, given that the number is assumed to be a month and that the day is assumed to be the first of the month
- a month and a one- or two-digit number, given that the number is assumed to be a day and that the year is the current year
- a number followed by a number, given that the two values are assumed to be the month and day as ordered in `>system>configuration>languages.tin` and that the year is current

A valid three token date consists of year, month, and day numbers. However, day-in-year numbers and ISO/ANSI strings are not valid. If it is obvious that a token represents a year (it is a four-digit year, a two-digit year greater than 80, or is preceded by a comma) or a month (it is a month name), then these tokens are interpreted as such. If a year token is identified, then the command function uses the forms defined in `>system>configuration>languages.tin` to determine the month and day. If a month token is identified, then the command function uses the forms defined in `>system>configuration>languages.tin` to determine the year and day. If a year and month are identified, the third value is assumed to be an day of the month in the range of 1 to 31.
For most time zones, this command function returns accurate values through December 31, 2048 (2048-12-31_23:59:59_gmt). However, because some local time zones can differ from GMT by up to 13 hours, the last full day for which this command function returns an accurate value is 2048-12-30 for those time zones.

With the -long argument, the date returned is in the form month day, year. With -standard, the (date) command function returns a date in the form defined in the system>configuration>languages.tin file.

Examples

Example 1.
The value of (date) can be 10-04-22. The value of (date -long) can be April 22, 2010. The value of (date -long -standard) can be 22 April 2010.

The following example illustrates the use of the (date) command function in a command.

```
set_expiration_date current_year_calendar (date coming 1/1)
```

Example 2.
The following example illustrates the use of the (date) command function in the Eastern Standard Time zone using the upper limit date.

```
ready 09:04:12
display_line (date 2048-12-31_23:59:59_gmt)
79-12-31
ready 09:15:47
```

The following example illustrates what happens when the upper limit is exceeded by a second. If the upper limit is exceeded by more than a second, the system displays: Invalid date or time.

```
ready 09:04:12
display_line (date 2049-01-01_00:00:00_gmt)
date: Invalid date or time.
ready 09:15:47
```

Related Information

See also the descriptions of the (date_time) and (time) command functions. For a list of the time zones supported by OpenVOS, see the set_time_zone command later in this manual.


**Purpose**

This command function returns a date and time.

**Syntax**

```
(date_time [date_string][-long][-standard])
```

**Explanation**

The (date_time) command function returns a date and time in the form `yy-mm-dd hh:mm:ss`. The `date_string` argument represents a date and time to be returned; if you omit it, the (date_time) command function returns the current date and time. The `date_string` must always be in the form defined in the `system>configuration>languages.tin` file. The `date_string` argument accepts any of the following keywords. These keywords are described in Table 1-5 later in this chapter.

```
absolute_date
absolute_time
coming
relative_terms
time_zone
```

For more information about the format of dates in the `date_string`, see the description of the (date) command function.

For most time zones, this command function returns accurate values through December 31, 2048 (2048-12-31_23:59:59_gmt). However, because some local time zones can differ from GMT by up to 13 hours, the last full day for which this command function returns an accurate value is 2048-12-30 for those time zones.

With the `-long` argument, the date and time returned is in the form `weekday, month day, year hh:mm:ss am/pm time_zone`. With `-standard`, the (date_time) command function returns a date in the form defined in the `system>configuration>languages.tin` file.
Examples

Example 1.
The value of (date_time) could be 94-10-09 15:25:45. The value of (date_time -long) could be Thursday, October 9, 1994 3:25 pm. The value of (date_time -long -standard) could be Venerdi’, 8 Dicembre 1994 12:14 del pomeriggio EST.

The following example illustrates the use of the (date_time) command function in a command.

    batch do_weekly_reports -defer_until (date_time Friday 6pm)

Example 2.
When the (date_time) command function executes with time_zone as an argument, the value returned is based on the current time in your own time zone. The value (date_time time_zone) returns is the time it will be in your own time zone when the time in the time zone specified by time_zone is the current time. The following example illustrates the (date_time) command function using time_zone (pst, Pacific Standard Time) as an argument issued in the Eastern Standard Time zone.

    ready 10:08:19
display_line (date_time pst)
97-11-19 13:08:39

The first ready prompt is the current time in Eastern Standard Time. The command function returns 13:08:39, which is the time it will be in Eastern Standard Time when the current time is 10:08:39 in Pacific Standard Time.

Example 3.
The following example illustrates the use of the (date_time) command function in the Eastern Standard Time zone using the upper limit date.

    ready 10:36:31
display_line (date_time 2048-12-31_23:59:59_gmt)
79-12-31 19:59:59
ready 10:39:50

The following example illustrates what happens when the upper limit is exceeded by a second. If the upper limit is exceeded by more than a second, the system displays: Invalid date or time.

    ready 10:39:50
display_line (date_time 2049-01-01_00:00:00_gmt)
date_time: Invalid date or time.
ready 10:41:35

Related Information
See also the descriptions of the (date) and (time) command functions. For a list of the time zones supported by OpenVOS, see the set_time_zone command later in this manual.
(decimal)

Purpose

This command function returns the decimal value of a binary, octal, decimal, or hexadecimal integer.

Syntax

(decimal I)

Explanation

The (decimal) command function returns the value of I expressed as a decimal integer. The integer I can be expressed as a binary, octal, decimal, or hexadecimal integer.

Example

The values of (decimal -1101b), (decimal -15o), (decimal -13d), and (decimal -dx) are -13.

The following example illustrates the use of the (decimal) command function in a command.

    display_line (decimal 0facex)

Related Information

See also the description of the (hexadecimal) command function.
(directory_name)

Purpose
This command function returns the full path name of the directory containing the specified object.

Syntax
(directory_name path_name)

Explanation
The (directory_name) command function returns the full path name of the directory containing the object specified by path_name. The input path name can be a relative path name.

Example
For example, if you specify (directory_name %s1#d01>Sales>Smith), the command function returns the value %s1#d01>Sales.

Related Information
See also the descriptions of the (current_dir), (home_dir), (object_name), (path_name), (referencing_dir), and (where_path) command functions.
Purpose
This command function determines whether the (contents) command function has read to the end of a file opened by a command macro.

Syntax
(end_of_file path_name)

Explanation
The (end_of_file) command function returns the value 1 if the last execution of the (contents) command function on a file held open in a macro returned $end_of_file; otherwise, the returned value is 0. If the file specified as path_name is not a held file, or if the function is invoked from outside a command macro, an error occurs.

Related Information
See also the descriptions of the (contents), (exists), and (file_info) command functions.
(exists)

Purpose

This command function determines whether a specified object exists.

Syntax

(exists path_name [object_type][chase_code])

Explanation

The (exists) command function returns the value 1 if the object specified by path_name exists, and you have status access to its containing directory, or non-null access to the object itself, and 0 otherwise.

You can meet either access requirement if you have non-null access to an object but no access to any of its containing directories. Inquiries about any of the containing directories would return 0, while inquiries about the object would return 1.

The path_name argument can be a star name. The value for the object_type argument can be -device, -file, -directory, or -link. If you omit a value for object_type, the operating system looks for all non-device types.

The value for chase_code can be either -chase or -no_chase. This argument controls whether the operating system chases a link to its ultimate target. If you omit chase_code, the operating system uses -chase, unless you specify -link, in which case -no_chase is used.

If you specify a star name as the value for path_name, you must specify -no_chase for chase_code.

You cannot use both -link for object_type and -chase for chase_code in the same command function, since the command function cannot look for a link and chase it at the same time. Therefore, if you specify -link, you must specify -no_chase.

Example

For example, the value of (exists %s1#d01>sales) is 1 if the object exists and you have appropriate access, and 0 if it does not exist.
Related Information

See also the descriptions of the (contents), (end_of_file), and (file_info) command functions.
Purpose
This command function determines whether extended-names support is enabled in the current execution environment.

Syntax
(extended_names [path_name])

Explanation
The (extended_names) command function determines whether extended-names support is enabled in the current execution environment.

The command function returns the value 1 if either of the following is true:

- path_name is not specified, and either version 1 or version 2 extended names is supported in the current execution environment

- path_name is specified, and the file system containing the object supports either version 1 or version 2 extended names

Otherwise, the command function returns the value 0.

Version 1 or version 2 extended names are enabled in OpenVOS releases unless a command macro has disabled this feature with the no_extended_names keyword of the &begin_parameters statement. See the OpenVOS Commands User’s Guide (R089) for more information about the &begin_parameters statement.

Example
For example, display_line (extended_names) displays either 1 or 0.

Related Information
See Using OpenVOS Extended Names (R631) for more information about extended-names support. See also the description of the (extended_names_version) command function.
(extended_names_version)

Purpose
This command function determines which version of extended-names support is enabled in
the current execution environment.

Syntax
(extended_names_version [path_name])

Explanation
The (extended_names_version) command function determines which version of
extended-names support is enabled in the current execution environment.

If path_name is not specified:

- The command function returns the value 2 if version 2 extended names is enabled.
- The command function returns the value 1 if version 1 extended names is enabled.
- Otherwise, the command function returns the value 0 (legacy names).

If path_name is specified and the object exists:

- The command function returns the value 2 if the file system containing that object
  supports version 2 extended names.
- The command function returns the value 1 if the file system containing that object
  supports version 1 extended names.
- Otherwise, the command function returns the value 0 (legacy names).

Version 1 or version 2 extended names are enabled in OpenVOS releases unless a command
macro has disabled this feature with the no_extended_names keyword of the
&begin_parameters statement. See the OpenVOS Commands User’s Guide (R089) for
more information about the &begin_parameters statement.

Example
For example, display_line (extended_names) displays either 2, 1, or 0.
Related Information

See Using OpenVOS Extended Names (R631) for more information about extended-names support. See also the description of the \((\text{extended\_names\_version})\) command function.
(file_info)

Purpose
This command function returns a specified piece of status information about a file.

Syntax
(file_info path_name key)

Explanation
The (file_info) command function returns a specified piece of information about the file specified by path_name, depending on the value of key. All dates returned by this command function use the form defined in the >system>configuration>languages.tin file. The following table shows the allowed values for key and the information returned by each value.

<table>
<thead>
<tr>
<th>Key</th>
<th>Returned Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>allocation_size</td>
<td>The allocation size set by the set_file_allocation command or 1, if that command is not used</td>
</tr>
<tr>
<td>author</td>
<td>The name of the file’s author</td>
</tr>
<tr>
<td>blocks_used</td>
<td>The size of the file, excluding the blocks used by indexes on that file</td>
</tr>
<tr>
<td>date_created</td>
<td>The date the file was created</td>
</tr>
<tr>
<td>date_modified</td>
<td>The date the file was last modified</td>
</tr>
<tr>
<td>date_saved</td>
<td>The date the file was last saved; returns the null string if there is no last-saved date</td>
</tr>
<tr>
<td>date_used</td>
<td>The date the file was last used; returns the null string if there is no last-used date</td>
</tr>
<tr>
<td>dynamic_extents</td>
<td>Returns 1 if the file has dynamic extents; returns 0 if it does not</td>
</tr>
<tr>
<td>expiration_date</td>
<td>The date the file expires; returns the null string if there is no expiration date</td>
</tr>
<tr>
<td>extent_size</td>
<td>Either the value of the extent size of a file, or 1 if the file is not an extent file, or -1 if the file has flexible extents (such a file is called a flex file)</td>
</tr>
<tr>
<td>implicit_locking</td>
<td>Returns 1 if the file has implicit locking; returns 0 if it does not</td>
</tr>
</tbody>
</table>
### Related Information

See also the descriptions of the `{contents}`, `{end_of_file}`, and `{exists}` command functions.

<table>
<thead>
<tr>
<th>Key</th>
<th>Returned Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>last_record</td>
<td>The last record number in a file with fixed, relative, sequential, extended sequential, stream, or 64-bit stream file organization; returns the null string for all other file organizations</td>
</tr>
<tr>
<td>organization</td>
<td>The type of file organization</td>
</tr>
<tr>
<td>pipe_file</td>
<td>Returns 1 if the file is a pipe file; returns 0 if it is not.</td>
</tr>
<tr>
<td>ram_file</td>
<td>Returns 1 if the file is a RAM file; returns 0 if it is not.</td>
</tr>
<tr>
<td>record_size</td>
<td>The maximum record size of the file if the file is fixed, relative, sequential, extended sequential, stream, or 64-bit stream; returns the null string for all other file organizations. If the file is a stream file, 0 is returned. If the file is a 64-bit stream file, a negative value is returned (-1 for 64-bit stream files and -2 for restricted 64-bit stream files).</td>
</tr>
<tr>
<td>sparse</td>
<td>Returns 1 if the file is a sparse file; returns 0 if it is not.</td>
</tr>
<tr>
<td>stream64_file</td>
<td>Returns 1 if the file is a 64-bit stream file; returns 0 if it is not.</td>
</tr>
<tr>
<td>transaction_file</td>
<td>Returns 1 if the file is a transaction file; returns 0 if it is not.</td>
</tr>
</tbody>
</table>
(floor)

**Purpose**

This command function returns the largest integer less than or equal to the specified number.

**Syntax**

(floor N)

**Explanation**

The (floor) command function returns the largest integer less than or equal to N.

**Examples**

For example, the value of (floor 1.5) is 1, and the value of (floor -1.5) is -2.

**Related Information**

See also the descriptions of the (abs), (calc), (ceil), (max), (min), and (mod) command functions.
Purpose
This command function indicates whether a command function parameter was supplied a value.

Syntax
(given parameter)

Explanation
The (given) command function returns the value 1 if the specified command macro parameter was supplied a value when the current command macro was issued, and 0 otherwise.

This command function can be used only in command macros.

Example
The value of (given source) is 1 when an argument was supplied that corresponds to the command macro parameter source, and 0 otherwise.
(group_name)

Purpose
This command function returns the name of your current group.

Syntax
(group_name)

Explanation
The (group_name) command function returns the name of your current group.

Example
For example, you can use the (group_name) command function in the login command as follows.

    login (group_name) -module %s1#m3

Related Information
See also the description of the (person_name) and (user_name) command functions.
(has_access)

**Purpose**

This command function indicates whether a user has the specified access rights to a specified object.

**Syntax**

(has_access path_name access_code [user_name])

**Explanation**

The (has_access) command function returns the value 1 if the user specified by user_name has at least the access rights specified by access_code to the object specified by path_name. Otherwise, the returned value is 0.

If the object specified by path_name does not exist, the returned value is 0.

If you specify a value for user_name, it must be of the form person_name.group_name. If you omit the user name, the access rights returned are your own.

The value you specify for access_code must be appropriate for the type of object you specify for path_name. The valid codes you can specify for access_code and their corresponding meanings for files and directories are as follows:

### Directories

- m for modify
- s for status
- n for null

### Files

- w for write
- r for read
- e for execute
- n for null

If you specify n as the value for access_code, the value the command function returns is always 1 since all users have at least null access to every object in the system.

**Example**

For example, if you specify (has_access %s#d01>Sales>Smith s Smith.Sales), the command function might return the value 1.
Related Information

See also the descriptions of the (group_name), (access), and (person_name) command functions.
Purpose
This command function returns the value of a binary, octal, decimal, or hexadecimal integer expressed as a hexadecimal integer.

Syntax
(hexadecimal I)

Explanation
The (hexadecimal) command function returns the value of I expressed as a hexadecimal integer. The integer I can be expressed as a binary, octal, decimal, or hexadecimal integer.

Example
The values of (hexadecimal 1101b), (hexadecimal 15o), (hexadecimal 13d), and (hexadecimal dx) are 0Dx.

The following example illustrates the use of the (hexadecimal) command function in a command.

    display_line (hexadecimal 1032)

Related Information
See also the description of the (decimal) command function.
(home_dir)

Purpose
This command function returns the full path name of your home directory.

Syntax
(home_dir)

Explanation
The (home_dir) command function returns the full path name of your home directory.

Example
For example, if your user name is Smith.Sales, the (home_dir) command function might return the value %s1#d02>Sales>Smith.

This command function might be used in the following command.

    emacs (home_dir)>abbreviations

Related Information
See also the descriptions of the (current_dir), (directory_name), (object_name), (path_name), (referencing_dir), and (where_path) command functions.
Purpose
This command function returns the position in a string of the first character in a substring.

Syntax
(index S1 S2)

Explanation
The (index) command function returns the position in the string S1 of the first character in
the substring S2. If S2 is not a substring of S1, the returned value is 0.

Example
For example, if you specify (index aabcc.ddeeff .), the command function returns the
value 6.

Related Information
See also the descriptions of the (after), (before), (break), (concat), (copy),
(count), (length), (ltrim), (quote), (reverse), (rtrim), (search), (string),
(substitute), (substr), (translate), (trunc), (unique_string), (unquote),
and (verify) command functions.
(iso_date)

Purpose
This command function returns a date using a four-digit year.

Syntax

(iso_date [date_string][-long][-standard])

Explanation
The (iso_date) command function returns a date in the form yyyy-mm-dd. The
date_string argument represents a date to be returned; if you omit it, the (iso_date)
command function returns the current date. The date_string must always be in the form
defined in the system>configuration>languages.tin file. The date_string
argument can use any of the following keywords. These keywords are described in Table 1-5
later in this chapter.

absolute_date
coming
relative_terms

A date in a date_string consists of one, two, or three tokens. A token can be either a
numeric string or an alphabetic string matching one of the words defined in the
>system>configuration>languages.tin file. The tokens can be separated by
delimiter characters such as spaces, periods (.), hyphens (-), or slashes (/). No delimiter
characters are needed if the month is represented by a character string or you use the
ISO/ANSI date formats. In a three-token date with two delimiter characters, at least one
delimiter character must be a space or the two delimiter characters must be the same. Note
that in two- or three-token dates, you can precede a year token with a comma (,).

Valid token formats are listed below.

- month name tokens can be strings in the form specified in
  >system>configuration>languages.tin
- one- or two-digit tokens are a month, day, or year
- two-digit tokens greater than or equal to 80 are a year from 1980 to 1999
- three-digit tokens from 1 to 366 are a day in the year
- four-digit tokens are a year
six-digit tokens in the form \( yymmdd \) are ISO/ANSI dates

- eight-digit tokens in the form \( yyyyymmdd \) are ISO/ANSI dates

- weekday name tokens in the form specified in

\[ \text{system>configuration>languages.tin} \]

Years must be in the range 1980 to 2048.

The following are valid single-token dates.

- an ISO/ANSI string of six or eight digits

- a month name, given that the day is assumed to be the first of the month and the year is the current year

- a four-digit year, given that the day is assumed to be January 1

- a two-digit year greater than or equal to 80, given that the day is assumed to be January 1

- a weekday name

Numbers less than 80 and three-digit numbers are not valid single token dates.

The following are valid two-token dates.

- a year followed by a three-digit day of the year

- a month and year, or year and month, given that the day is assumed to be the first of month

- a year and a two-digit number, or two-digit number and year, given that the number is assumed to be a month and that the day is assumed to be the first of the month

- a month and a one- or two-digit number, given that the number is assumed to be a day and that the year is the current year

- a number followed by a number, given that the two values are assumed to be the month and day as ordered in \[ \text{system>configuration>languages.tin} \] and that the year is the current year.

A valid three-token date consists of year, month, and day numbers. However, day-in-year numbers and ISO/ANSI strings are not valid. If a token represents a year (it is a four-digit year, a two-digit year greater than 80, or is preceded by a comma) or a month (it is a month name), then these tokens are interpreted as such. If a year token is identified, then the command function uses the forms defined in \[ \text{system>configuration>languages.tin} \] to determine the month and day. If a month token is identified, then the command function uses the forms defined in \[ \text{system>configuration>languages.tin} \] to determine the year and day. If a year and month are identified, the third value is assumed to be a day of the month in the range of 1 to 31.
For most time zones, this command function returns accurate values through December 31, 2048 (2048-31_23:59:59_gmt). However, because some local time zones can differ from GMT by up to 13 hours, the last full day for which this command function returns an accurate value is 2048-12-30 for those time zones.

With the -long argument, (iso_date) returns the date in the form month day, year.

With -standard, (iso_date) returns the date in the form defined in the system/configuration/languages.tin file. If you do not specify a value for this argument, (iso_date) returns the current date. If you specify the -standard argument, you can specify coming in your process language; for example, if your process language is Italian, the term might be prossimo. See Table 1-5 for more information.

**Examples**

**Example 1.**
The value of (iso_date) could be 1997-08-01. The value of (iso_date -long) could be August 1, 1997. The value of (iso_date -long -standard) could be 1 August 1997.

The following example illustrates the use of the (iso_date) command function in a command.

```bash
set_expiration_date current_year_calendar (iso_date coming 1/1)
```

**Example 2.**
The following example illustrates the use of the (iso_date) command function in the Eastern Standard Time zone using the upper limit date.

```bash
ready 09:04:12
display_line (iso_date 2048-12-31_23:59:59_gmt)
2048-12-31
ready 09:15:47
```

The following example illustrates what happens when the upper limit is exceeded by a second. If the upper limit is exceeded by more than a second, the system displays: Invalid date or time.

```bash
ready 09:04:12
display_line (iso_date 2049-01-01_00:00:00_gmt)
iso_date: Invalid date or time.
ready 09:15:47
```

**Related Information**

See also the descriptions of the (iso_date_time), (date_time), and (time) command functions. For a list of the time zones supported by OpenVOS, see the set_time_zone command later in this manual.
(iso_date_time)

Purpose
This command function returns a date using a four-digit year and a time.

Syntax

(iso_date_time [date_time_string][ -long][ -standard])

Explanation
The (iso_date_time) command function returns a date and time in the form yyyy-mm-dd hh:mm:ss. The date_time_string argument represents a date and time to be returned; if you omit it, the (iso_date_time) command function returns the current date and time. The date_time_string argument must always be in the form defined in the system>configuration>languages.tin file. The date_time_string argument accepts any of the following keywords. These keywords are described in Table 1-5 later in this chapter.

absolute_date
absolute_time
coming
relative_terms
time_zone

For more information about the format of dates in the date_time_string, see the description of the (iso_date) command function.

For most time zones, this command function returns accurate values through December 31, 2048 (2048-12-31_23:59:59_gmt). However, because some local time zones can differ from GMT by up to 13 hours, the last full day for which this command function returns an accurate value is 2048-12-30 for those time zones.

With -long, the date-time value is in the form weekday, month day, year hh:mm:ss am/pm time_zone. For example, if the value of (iso_date_time) is 2001-06-09 15:25:45, the value of (iso_date_time -long) would be Tuesday, June 9, 2001 3:25 pm est.

With -standard, the (iso_date_time) command function returns a date, and accepts a date_time_string argument formed in accordance with the date and time values in the language definition for the process language. If you give no value for date_time_string, the returned value is the current date and time. If you specify the -standard argument, you
can specify coming in your process language; for example, if your process language is Italian, the term might be prossimo. See Table 1-5 for more information.

Examples

Example 1.
The value of (iso_date_time) could be 1997-08-01 12:50:50. The value of (iso_date_time -long) could be Friday, August 1, 1997 12:50 pm edt. The value of (iso_date_time -long -standard) could be Friday, August 1, 1997 12:50 pm.

The following example illustrates the use of the (iso_date_time) command function in a command:

```
batch do_weekly_reports -defer_until (iso_date_time Friday 6pm)
```

Example 2.
When the (iso_date_time) command function executes with time_zone as an argument, the value returned is based on the current time in your own time zone. The value (iso_date_time time_zone) returns is the time it will be in your own time zone when the time in the time zone specified by time_zone is the current time. The following example illustrates the (iso_date_time) command function using time_zone (pst, Pacific Standard Time) as an argument issued in the Eastern Standard Time zone.

```
ready 10:08:19
display_line (iso_date_time pst)
97-11-19 13:08:39
ready 10:08:39
```

The first ready prompt is the current time in Eastern Standard Time. The command function returns 13:08:39, which is the time it will be in Eastern Standard Time when the current time is 10:08:39 in Pacific Standard Time.

Example 3.
The following example illustrates the use of the (iso_date_time) command function in the Eastern Standard Time zone using the upper limit date.

```
ready 09:04:12
display_line (iso_date_time 2048-12-31_23:59:59_gmt)
2048-12-31 19:59:59
ready 09:15:47
```

The following example illustrates what happens when the upper limit is exceeded by a second. If the upper limit is exceeded by more than a second, the system displays: Invalid date or time.

```
ready 09:04:12
display_line (iso_date_time 2049-01-01_00:00:00_gmt)
iso_date_time: Invalid date or time.
ready 09:15:47
```
Related Information

See also the descriptions of the \texttt{iso\_date}, \texttt{date\_time}, and \texttt{time} command functions. For a list of the time zones supported by OpenVOS, see the \texttt{set\_time\_zone} command later in this manual.
Purpose
   This command function returns the name of the language of your current process.

Syntax
   (language_name)

Explanation
   The (language_name) command function returns the name of the language of your current process.

Example
   The following example illustrates the use of the (language_name) command function.

   set_library_paths message
   %s1#d03>Sales>message_library>'(language_name)'

Related Information
   See the description of the set_language command and the OpenVOS Commands User’s Guide (R089) for more information on using languages.
(length)

Purpose
This command function returns the length of a string.

Syntax
(length [S])

Explanation
The (length) command function returns the length of the string S. If S contains spaces, the (length) command function treats S as a single string. If you omit S, the value of (length) is 0.

Examples
For example, if you specify (length aabbcc.ddeeff), the command function returns the value 13. If you specify (length abc.def), the command function returns the value 7.

You might use the (length) command function in a command macro, as follows:

   &if (length &message&) = 0
   &then &return

Related Information
See also the descriptions of the (after), (before), (break), (concat), (copy), (count), (index), (ltrim), (quote), (reverse), (rtrim), (search), (string), (substitute), (substr), (translate), (trunc), (unique_string), (unquote), and (verify) command functions.
(lock_type)

**Purpose**

This command function returns the type of lock on a file.

**Syntax**

```
(lock_type path_name)
```

**Explanation**

The `(lock_type)` command function returns the type of lock on the file specified by `path_name`. The command function returns one of the following values.

- dirty_readers
- implicit_lock
- no_lock
- read_lock
- record_lock
- region_locking
- transaction_file
- write_lock

**Related Information**

See also the description of the `(locked)` command function.
Purpose
This command function indicates whether a file is locked.

Syntax
(locked path_name)

Explanation
The (locked) command function returns the value 1 if the file specified by path_name is locked, and 0 if it is not locked.

Related Information
See also the description of the (lock_type) command function.
(ltrim)

Purpose

This command function returns the longest final substring in a string whose first character is not in a specified set of characters.

Syntax

(ltrim $S [C])

Explanation

The (ltrim) command function returns the longest final substring of $S$ whose first character is not in the set $C$. If you omit $C$, the operating system trims leading spaces from $S$.

Example

For example, if you specify (ltrim aabbcc.ddeeff af), the command function returns the value bbcc.ddeeff.

Related Information

See also the descriptions of the (after), (before), (break), (concat), (copy), (count), (index), (length), (quote), (reverse), (rtrim), (search), (string), (substitute), (substr), (translate), (trunc), (unique_string), (unquote), and (verify) command functions.
(master_disk)

Purpose
This command function returns the full path name of the master disk directory on a module.

Syntax
(master_disk [module_name])

Explanation
The (master_disk) command function returns the full path name of the master disk directory on the module module_name, or on the current module when you omit the argument.

Example
The following example illustrates the use of the (master_disk) command function.

display_disk_info (master_disk)
%s1#d01

Related Information
See also the descriptions of the (current_module), (module_name), (online), and (system_name) command functions.
(max)

Purpose
This command function returns the larger of two specified values.

Syntax
(max N1 N2)

Explanation
The (max) command function returns the larger of the two values specified in the arguments N1 and N2.

Example
For example, if you specify (max 33 36), this command function returns the value 36.

Related Information
See also the descriptions of the (abs), (calc), (ceil), (floor), (min), and (mod) command functions.
Purpose

This command function returns the text of a system message corresponding to a status code.

Syntax

(message status_code [S1 [S2 [S3]]] [-number][-no_number])

Explanation

The (message) command function returns the text of a system message corresponding to the value specified by status_code. The value of the status_code argument can be one of the following:

- an integer specifying a system status code
- the command function (command_status)
- the name of a system status message, beginning with e$, m$, r$, or q$

If you specify any of the optional string arguments (S1, S2, and S3), they replace any parameters defined in the text of the returned message. If there are no parameters defined in the text of the message, the string arguments are ignored.

The -number argument displays the status code number of status_code. The -no_number argument cancels the effect of a previous -number argument. If you specify both arguments, the last one specified on the command line takes effect. The default value is -no_number.

Examples

The following examples show the various ways that you can specify values for this command function.

To return the text of the system message e$object_not_found, issue the command function (message e$object_not_found). The command function returns the value Object not found.

To determine a status code number, issue the command function (message e$object_not_found -number). The command function returns the value 1032.

If you use the (command_status) command function as the value of status_code, the (message) command function returns the text of the message corresponding to the code returned by (command_status). Since none of the messages corresponding to codes
returned by (command_status) have parameters, you do not need to specify the optional string arguments. For example, if you specify (message (command_status)), the command function might return the value Object not found.

The (message 1434 Smith Sales m2) sample command function uses the optional string arguments. This command function returns the value Smith not registered for group Sales on module m2.

Related Information

See also the description of the (command_status) command function.
Purpose
   This command function returns the smaller of two specified values.

Syntax
   (min N1 N2)

Explanation
   The (min) command function returns the smaller of the two values specified in the arguments N1 and N2.

Example
   For example, if you specify (min 30 43), the command function returns the value 30.

Related Information
   See also the descriptions of the (abs), (calc), (ceil), (floor), (max), and (mod) command functions.
(mod)

Purpose
This command function returns the remainder of a division operation.

Syntax
(mod N1 N2)

Explanation
The (mod) command function returns the remainder of the division of N1 by N2.

Example
For example, if you specify (mod 14 4), the command function returns the value 2.

Related Information
See also the descriptions of the (abs), (calc), (ceil), (floor), (max), and (min) command functions.
**Purpose**
This command function returns particular status information about the current module.

**Syntax**
(module_info key)

**Explanation**
The (module_info) command function returns information about the current module, depending on the value of *key*. The following table shows the kind of information returned, based on the value of *key*.

<table>
<thead>
<tr>
<th>Key</th>
<th>Returned Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>bootloader_time</td>
<td>The date/time that the operating system started running.</td>
</tr>
<tr>
<td>cache_mem_percent</td>
<td>The percentage of memory over 128 MB to use for the cache.</td>
</tr>
<tr>
<td>cate_write_limit</td>
<td>A value shown for compatibility purposes only.</td>
</tr>
<tr>
<td>cpu_family</td>
<td>The identification string for the CPU architecture.</td>
</tr>
<tr>
<td>cpu_type</td>
<td>The identification string for the type of CPU that is in use.</td>
</tr>
<tr>
<td>disk_mod_limit</td>
<td>The maximum number of modified blocks per disk that the cache manager is allowed to hold in cache before initiating disk writes.</td>
</tr>
<tr>
<td>disk_write_limit</td>
<td>The maximum number of write requests that the cache manager is allowed to queue to each disk.</td>
</tr>
<tr>
<td>free_grace_time</td>
<td>The length of time, in seconds, before an unreferenced physical page is returned to the pool of physical pages available for virtual memory management.</td>
</tr>
<tr>
<td>free_kernel_vm</td>
<td>The number of 4096-byte pages of free kernel virtual memory.</td>
</tr>
<tr>
<td>max_buffers</td>
<td>The maximum number of physical disk cache buffers.</td>
</tr>
<tr>
<td>Key</td>
<td>Returned Value</td>
</tr>
<tr>
<td>---------------------------------</td>
<td>-------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>max_events_per_module</td>
<td>The maximum number of events allowed on the module.</td>
</tr>
<tr>
<td>max_events_per_process</td>
<td>The maximum number of user events, allowed per process, on the module.</td>
</tr>
<tr>
<td>max_events_per_task</td>
<td>The maximum number of events that a task can wait for.</td>
</tr>
<tr>
<td>max_local_devices_per_module</td>
<td>The maximum number of devices allowed on the module.</td>
</tr>
<tr>
<td>max_login_processes</td>
<td>The administrative limit on the maximum number of processes that can be created.</td>
</tr>
<tr>
<td>max_processes</td>
<td>The implementation limit on the maximum number of processes that can be created.</td>
</tr>
<tr>
<td>max_resident_percent</td>
<td>The percentage of the maximum cache size to make available for memory-resident files.</td>
</tr>
<tr>
<td>max_virtual_pages</td>
<td>The maximum number of virtual pages, used to map the physical pages, in the cache.</td>
</tr>
<tr>
<td>min_buffers</td>
<td>The minimum number of physical disk cache buffers.</td>
</tr>
<tr>
<td>min_cache_priority</td>
<td>A minimum priority for processes that are entitled to retain data in the cache.</td>
</tr>
<tr>
<td>modified_grace_time</td>
<td>The length of time, in seconds, that a modified block is left in memory before it is written to disk.</td>
</tr>
<tr>
<td>n_cpus</td>
<td>The number of CPUs that are supported by this module.</td>
</tr>
<tr>
<td>n_local_devices</td>
<td>The number of devices that are in use by this module.</td>
</tr>
<tr>
<td>n_processes</td>
<td>The number of processes that are in use by this module.</td>
</tr>
<tr>
<td>os_release</td>
<td>The identification string for the version of the operating system that is in use.</td>
</tr>
<tr>
<td>system_release</td>
<td></td>
</tr>
<tr>
<td>vos_release</td>
<td></td>
</tr>
<tr>
<td>recover_disk_priority</td>
<td>The default priority level of any process created to perform a recover_disk operation.</td>
</tr>
<tr>
<td>referenced_grace_time</td>
<td>The length of time, in seconds, before a referenced physical page is forcibly unreferenced.</td>
</tr>
<tr>
<td>total_kernel_vm</td>
<td>The total number of 4096-byte pages of kernel virtual memory.</td>
</tr>
</tbody>
</table>
Related Information

See also the descriptions of the (current_module), (master_disk), (module_name), (online), and (system_name) command functions.

<table>
<thead>
<tr>
<th>Key</th>
<th>Returned Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>transient_mod_grace_time</td>
<td>The length of time, in seconds, that the cache manager waits after a block from a transient file is modified before the cache manager writes the block to disk.</td>
</tr>
<tr>
<td>unreferenced_grace_time</td>
<td>The length of time, in seconds, before an unreferenced physical page is available for reuse, as a buffer, for a different disk block.</td>
</tr>
<tr>
<td>unused_dir_timeout</td>
<td>The number of seconds a directory can be unused before its contents are written to disk.</td>
</tr>
<tr>
<td>used_kernel_vm</td>
<td>The number of 4096-byte pages of used kernel virtual memory.</td>
</tr>
</tbody>
</table>
Purpose
This command function returns the full path name of a module.

Syntax
(module_name module_name)

Explanation
The (module_name) command function returns the full path name of a module specified by module_name. Examples of valid values for module_name include %s1#m2, #m2, and m2.

Example
For example, if you specify (module_name #m2), the command function might return the value %s1#m2.

Related Information
See also the descriptions of the (current_module), (master_disk), (module_info), (online), and (system_name) command functions.
(name_string)

Purpose
This command function returns the strings $S_1$ through $S_n$ combined into one string, with each component separated from the next by one space.

Syntax

\[(\text{name_string} \ S_1 \ldots S_n)\]

Explanation
The (name_string) command function returns the strings $S_1$ through $S_n$ combined into one string, with each component separated from the next by one space. You can use this function to combine multiple arguments into a single character-string argument when one of the arguments is a name containing one of the special command-line characters (embedded apostrophes, exclamation points, semicolons, spaces, and left and right parentheses) that must be quoted in the final result string.

Example
For example, if you specify \( (\text{name_string} \ 'a\ b' \ 'cde') \), the command function would return the value \( 'a\ b' \ cde \).

Related Information
See also the descriptions of the (after), (before), (break), (concat), (copy), (count), (index), (length), (ltrim), (quote), (reverse), (rtrim), (search), (string), (substitute), (substr), (translate), (trunc), (unique_string), (unquote), and (verify) command functions.
**Purpose**

This command function returns the name of an object.

**Syntax**

```
(object_name path_name [suffix])
```

**Explanation**

The `(object_name)` command function returns the name of the object specified by `path_name`, with the specified suffix, if any, appended.

**Example**

For example, if you specify `(object_name %sys#m1>sales>jones>abbreviations)`, the command function returns the value `abbreviations`.

If you specify `(object_name make_reports .cobol)`, the command function returns the value `make_reports.cobol`.

**Related Information**

See also the descriptions of the `(current_dir)`, `(directory_name)`, `(home_dir)`, `(path_name)`, `(referencing_dir)`, and `(where_path)` command functions.
(online)

Purpose
This command function indicates if a module is currently online.

Syntax

(online [module_name])

Explanation
The (online) command function returns the value 1 if the module specified by
module_name is currently online and available through the network. The command function
returns 0 if one of the following conditions exist:

- the specified module is not online
- the specified module does not exist
- the current user has no access to the specified module

If you do not specify a value for module_name, the command function returns the value 1.

Examples of valid values for module_name are %s1#m2, #m2, and m2.

Example
For example, if you specify (online #m5), the command function returns the value 1.

Related Information
See also the descriptions of the (module_info), (master_disk), (module_name), and
(system_name) command functions.
**Purpose**
This command function returns the full path name of an object.

**Syntax**

```
(path_name path_name [suffix])
```

**Explanation**
This command function returns the full path name of the object specified by `path_name`, with the specified suffix, if any, appended.

**Example**
For example, if you specify `(path_name make_reports .cobol)`, the command function returns the value `%s1#d01>Sales>Smith>make_reports.cobol`.

**Related Information**
See also the descriptions of the `(current_dir)`, `(directory_name)`, `(home_dir)`, `(object_name)`, `(referencing_dir)`, and `(where_path)` command functions.
(person_name)

Purpose
This command function returns your person name.

Syntax
(person_name)

Explanation
The (person_name) command function returns your person name.

Example
The following example illustrates the use of the (person_name) command function in a command.

```
stop_process -user (person_name)
Smith
```

Related Information
See also the description of the (group_name) and (user_name) command functions.
Purpose

This command function converts an OpenVOS path name into a POSIX path name.

Syntax

(posix_path path_name)

Explanation

The (posix_path) command function converts the path_name argument, which is a relative or full OpenVOS path name, into a full POSIX path name. The resultant POSIX path name always begins with a slash character (/).

The command function expands path_name into a full OpenVOS path name, which processes and removes any dot components ( . or ..). In the following explanation, %sys is the current system.

- If the expanded path name has the form %sys#null, the result is /dev/null.
- If the expanded path name has the form %sys#master_disk, and %sys#master_disk is the current module’s master disk, the result is /.
- If the expanded path name has the form %sys#name, the result has the form /%sys#name/.
- If the expanded path name has a form other than the preceding forms, it is processed to remove the name of the master disk (if present) and to convert all greater-than characters (>) to slash characters.
Example

The following examples assume that the master disk is `%s1#d01` and the current directory is `%s1#d01>SysAdmin`.

```
display_line (posix_path Sales>Jones)
/SysAdmin/Sales/Jones

display_line (posix_path %s1#d01)
/

display_line (posix_path <Sales>Jones)
/Sales/Jones

display_line (posix_path %s1#null)
/dev/null

display_line (posix_path (master_disk))
/

display_line (posix_path (master_disk)>system)
/system
```

The following example assumes that the current directory is `%s1#d02>Research`.

```
display_line (posix_path .)
/%s1#d02/Research
```

Related Information

See also the description of the (vos_path) command function.
(process_dir)

Purpose
This command function returns the full path name of the process directory of the current process.

Syntax
(process_dir)

Explanation
The (process_dir) command function returns the full path name of the process directory of the current process.

Example
The following example illustrates the use of the (process_dir) command function in a command.

    list (process_dir)>* -names_only
    %se#m29>process_dir_dir>pd.011D889B is empty.

Related Information
See also the descriptions the (process_info), (process_type), and (referencing_dir) command functions.
Purpose
This command function returns particular process-specific information.

Syntax
(process_info key)

Explanation
The (process_info) command function returns various kinds of process-specific information, depending on the value of key. The following table shows the allowed values for key and the information returned by each value.
### OpenVOS Commands Reference Manual (R098) 1-77

#### Related Information

See also the descriptions the (process_dir), (process_type), and (referencing_dir) command functions.

<table>
<thead>
<tr>
<th>Key</th>
<th>Returned Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>cpu_time</td>
<td>The amount of time, excluding time for page faults, the processor has spent running the process’s programs. The unit of time is 1/65,536 of a second.</td>
</tr>
<tr>
<td>disk_reads</td>
<td>The number of times the process has read data from the disk, except for page faults, since the process was created</td>
</tr>
<tr>
<td>disk_writes</td>
<td>The number of times the process has written data to the disk since the process was created</td>
</tr>
<tr>
<td>language</td>
<td>The name of the language of the current process.</td>
</tr>
<tr>
<td>login_time</td>
<td>The time that the process logged in</td>
</tr>
<tr>
<td>page_fault_time</td>
<td>The accumulated time the CPU has spent in page faults for the process. The unit of time is 1/65,536 of a second.</td>
</tr>
<tr>
<td>page_faults</td>
<td>The number of page faults the process has taken since it was created</td>
</tr>
<tr>
<td>parent_process_id</td>
<td>The process ID of the parent process, converted into a hexadecimal string</td>
</tr>
<tr>
<td>priority</td>
<td>A number from 0 to 9 giving the priority of the process</td>
</tr>
<tr>
<td>privileged</td>
<td>1 if the process is privileged; 0 otherwise</td>
</tr>
<tr>
<td>process_id</td>
<td>The current process ID, converted into a hexadecimal string</td>
</tr>
<tr>
<td>process_name</td>
<td>The name of the process</td>
</tr>
<tr>
<td>program_name</td>
<td>The entry name of the current program module, internal command, or command macro. In cases where a command macro runs a program module or internal command, the entry name refers to the .pm or internal command in preference to the command macro. If no .pm, .cm, or internal command is executing, the null string is returned.</td>
</tr>
<tr>
<td>sub_process_level</td>
<td>The current subprocess level. For a login process (created from pre-login), the subprocess level is 0; for a subprocess created from a login process or another subprocess, the level is the number of subprocesses created.</td>
</tr>
<tr>
<td>subsystem</td>
<td>The subsystem your process was created under when it logged in</td>
</tr>
</tbody>
</table>
Purpose
This command function returns the type of the current process.

Syntax
(process_type)

Explanation
The (process_type) command function returns the type of the current process. The possible returned values are as follows:

- batch
- interactive
- sub_process

The value of (process_type) for your login process is interactive when the process is created from pre-login, and sub_process when you log in a subprocess. The process type of a started process is batch.

Example
The following example illustrates the use of (process_type) in a command macro.

```
&if (process_type) = batch
 &then &goto batch
```

Related Information
See also the descriptions the (process_dir), (process_info), and (referencing_dir) command functions.
### Purpose
This command function concatenates a set of strings and encloses them in apostrophes (').

### Syntax
\[(quote \; S1 \; \ldots \; Sn)\]

### Explanation
The \(quote\) command function returns the strings \(S1\) through \(Sn\) combined into one string, with each component separated from the next by one space, and apostrophes (') added at the beginning and the end.

If the arguments contain apostrophes, the command processor removes one level of apostrophes before passing the arguments to the command function.

### Examples
The following table illustrates how the \(quote\) command function concatenates strings.

<table>
<thead>
<tr>
<th>Command Function</th>
<th>Returned Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>(quote ; aa ; bb)</td>
<td>'aa bb'</td>
</tr>
<tr>
<td>(quote ; 'aa ; bb')</td>
<td>'aa bb'</td>
</tr>
<tr>
<td>(quote ; aa''bb)</td>
<td>'aabb'</td>
</tr>
<tr>
<td>(quote ; 'aa''bb')</td>
<td>'aa''bb'</td>
</tr>
</tbody>
</table>

### Related Information
See also the descriptions of the \(after\), \(before\), \(break\), \(concat\), \(copy\), \(count\), \(index\), \(length\), \(ltrim\), \(reverse\), \(rtrim\), \(search\), \(string\), \(substitute\), \(substr\), \(translate\), \(trunc\), \(unique\_string\), \(unquote\), and \(verify\) command functions.
(rank)

Purpose
This command function returns the integer rank in the ASCII collating sequence of a character.

Syntax
(rank C)

Explanation
The (rank) command function returns the integer rank in the ASCII collating sequence of the character specified by C. The command function (rank) is the inverse of the command function (byte).

Example
For example, if you specify (rank a), the command function returns the value 97.

Related Information
See also the description of the (byte) command function.
Purpose
This command function returns the full path name of the directory containing a program module, if the current process is running a program module.

Syntax
(referencing_dir)

Explanation
The (referencing_dir) command function returns the full path name of the directory containing the currently loaded program module or command macro. If an internal command is executing, the null string is returned.

If a command macro is running a program module, the referencing directory refers to the program module; otherwise, it is the directory containing the command macro.

If no program module, internal command, or command macro is running, the referencing directory is null, the error message (No program is currently loaded.) is printed, and command_status is set to e$no_program_loaded (1805).

Example
The value of (referencing_dir) for the program module %s1#d03>Sales>tools>quota.pm is %s1#d03>Sales>tools. The following illustrates the use of the (referencing_dir) command function.

   set_library_paths message '(referencing_dir)'

Related Information
See also the descriptions the (command_status), (message), (process_dir), (process_info), and (process_type) command functions.
Purpose
This command function returns the reversed character pattern of a string.

Syntax
(reverse S)

Explanation
The (reverse) command function returns the reversed character pattern of the string S.

Example
For example, if you specify (reverse loot), the command function returns the value tool.

Related Information
See also the descriptions of the (after), (before), (break), (concat), (copy),
(count), (index), (length), (ltrim), (quote), (rtrim), (search), (string),
(substitute), (substr), (translate), (trunc), (unique_string), (unquote),
and (verify) command functions.
(rtrim)

Purpose
This command function returns the longest initial substring in a string whose last character is not in a specified set of characters.

Syntax

(rtrim S [C])

Explanation
The (rtrim) command function returns the longest initial substring of $S$ whose last character is not in the set $C$. If you omit $C$, the operating system trims trailing spaces from $S$.

Example
For example, if you specify (rtrim aabbcc.ddeeff af), the command function returns the value aabbcc.ddee.

Related Information
See also the descriptions of the (after), (before), (break), (concat), (copy), (count), (index), (length), (ltrim), (quote), (reverse), (search), (string), (substitute), (substr), (translate), (trunc), (unique_string), (unquote), and (verify) command functions.
(search)

Purpose
This command function returns the leftmost position in a string that contains a character in a specified set.

Syntax
(search S C)

Explanation
The (search) command function returns the leftmost position in the string S that contains a character in the set C. If no member of C is in S, the value of this function is 0.

Example
For example, if you specify (search aabbcc.ddeeff .), the command function returns the value 7. If you specify (search decipher code), the command function returns the value 1.

Related Information
See also the descriptions of the (after), (before), (break), (concat), (copy), (count), (index), (length), (ltrim), (quote), (reverse), (rtrim), (string), (substitute), (substr), (translate), (trunc), (unique_string), (unquote), and (verify) command functions.
(software_purchased)

Purpose
This command function returns the value 1 if the product denoted by
software_purchased_bit has been purchased; otherwise, the command function returns
the value 0.

Syntax
(software_purchased software_purchased_bit)

Explanation
The (software_purchased) command function returns 1 if the product denoted by
software_purchased_bit has been purchased. Otherwise, this command function
returns 0.

Example
For example, assume that you have purchased the OpenVOS PL/I compiler, whose
software-purchased bit number is S024. If you specify the command display_line
(software_purchased s024), the command function returns the value 1.
Purpose
This command function concatenates two or more strings.

Syntax
(string $1 ...$n)

Explanation
The (string) command function returns the strings $1 through $n combined into one string, with each component separated from the next by one space. You can use this function to combine multiple arguments into a single character-string argument that would otherwise require an additional level of apostrophes.

The following paragraphs describe how the command processor processes quotes, and how to use the (quote), (unquote), and (string) command functions.

Note: In the following examples, square brackets ([ ]) are used to delineate separate arguments.

Using Quotes in Command Functions
If a command line contains three separate unquoted arguments, the arguments are returned as separate and unquoted.

ready: display_line a b c

[a] [b] [c]

In the process of evaluating a command line and in evaluating the return value of a command function, the command processor strips one level of quotes. The second example returns three arguments, a b c, d e, and f.

ready: display_line ‘a b c’

[a b c]

ready: display_line ‘a b c’ ‘d e’ f

[a b c] [d e] [f]

The command processor regards quotes, semicolons, spaces, and left and right parentheses as special characters that must be quoted if they appear in arguments. If you do not place
semicolon in quotes, the command processor returns the message Parentheses are not balanced.

The command processor replaces the instance of the command function by its return value.

```
ready: display_line (date)
[ 94-12-07 ]
```

All command functions, except (unquote), return their result as a single argument. A quoted string is also a single argument. This protects any special characters present in the return value, and makes the return value subsequently appear as a single argument. The (unquote) command function allows a return value to be parsed as multiple arguments.

In the following examples, the return value is one argument.

```
ready: display_line (date_time)
[ 94-12-07 10:20:00 ]
ready: display_line ‘94-12-07 10:20:00’
[ 94-12-07 10:20:00 ]
```

In the following example, the return value is a special character which could be used as a single argument in another command function. Note that ASCII character 41 is the right parentheses.

```
ready: display_line (byte 41)
[ )]
```

In the following example, the return value is one argument, and it contains a special character.

```
ready: display_line (translate ‘a b c’ ‘;’ ‘b’)
[ a ; c ]
```

Using the (string), (quote), and (unquote) Command Functions

The (string), (quote), and (unquote) command functions first concatenate all of their input arguments and separate each argument by a single space. After this, each command function performs the following actions:

<table>
<thead>
<tr>
<th>Command Function</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>(string)</td>
<td>Returns the result as a single argument</td>
</tr>
<tr>
<td>(quote)</td>
<td>Adds a level of quotes, including doubling any internal quotes, then returns the result as a single argument</td>
</tr>
<tr>
<td>(unquote)</td>
<td>Returns the result as multiple arguments</td>
</tr>
</tbody>
</table>
In the following example, the (string) command function returns one argument.

```plaintext
ready: display_line (string a b c)
[a b c]
```

In the following example, the (quote) command function returns one argument.

```plaintext
ready: display_line (quote a b c)
['a b c']
```

In the following example, the (unquote) command function strips off the quotes and returns one argument. The command processor then tries to process the command line. This example is equivalent to typing pl1; bind at the command line.

```plaintext
ready: display_line (unquote pl1 ';' bind)
[pl1]
bind: A required argument is missing. One of the following required:
    object_modules, -control.
```

The (unquote) command function is best used to return new commands to be executed and to undo the effects of the (quote) command function. However, as shown the following examples, the result of using the (unquote (quote)) command functions is the same as the result of using the (string) command function. In both example, three arguments are returned. In this case, using the (string) command function makes the command line more easily readable.

```plaintext
ready: display_line (unquote (quote a ';' c))
[a ; c]
ready: display_line (string a ';' c)
[a ; c]
```

Using a $ (dollar sign) before a variable in a command macro is the same as using the (string) command function. Using &$variable& is the same as (string &variable&). The $ (dollar sign) allows a variable to contain special characters.

**Example**

The following example illustrates the use of the (string) command function in an abbreviations file.

```plaintext
first ad by analyze_system -request_line (string display &1&)
```
Related Information

See also the descriptions of the (after), (before), (break), (concat), (copy), (count), (index), (length), (ltrim), (quote), (reverse), (rtrim), (search), (substitute), (substr), (translate), (trunc), (unique_string), (unquote), and (verify) command functions.
Purpose
This command function replaces one substring with another in a specified string.

Syntax

(substitute S1 S2 S3 [-ignore_quotes][ -no_ignore_quotes])

Explanation
The (substitute) command function returns the string that results from locating each occurrence of S2 within S1, then replacing each occurrence with S3.

The command processor removes one level of apostrophes in the string S1 before passing it to the (substitute) command function. In the resulting string, no occurrence of S2 within a part of S1 enclosed in apostrophes is replaced.

The -ignore_quotes argument causes every occurrence of S2 within S1 to be replaced regardless of any part of S1 enclosed in apostrophes. The -no_ignore_quotes argument cancels the effect of a previous -ignore_quotes argument. These two arguments are mutually exclusive. The default value is -no_ignore_quotes.

Example
For example, if you specify (substitute 'aabbcc''aabbcc'' 'bb' 'BB'), the command function returns aaBBcc'aabbcc'. If you specify (substitute 'aabbcc''aabbcc'' 'BB' 'BB' -ignore_quotes), the command function returns aaBBcc'aabbcc'.

Related Information
See also the descriptions of the (after), (before), (break), (concat), (copy), (count), (index), (length), (ltrim), (quote), (reverse), (rtrim), (search), (string), (substr), (translate), (trunc), (unique_string), (unquote), and (verify) command functions.
(substr)

Purpose
This command function returns a substring contained in a string.

Syntax

(substr S I1 [I2])

Explanation
Returns the substring of the string S that begins in position I1 and extends for I2 characters or, if you omit I2, to the end of S.

Example
For example, if you specify (substr aabbcc.ddeeff 3 4), the command function returns bbcc.

Related Information
See also the descriptions of the (after), (before), (break), (concat), (copy), (count), (index), (length), (ltrim), (quote), (reverse), (rtrim), (search), (string), (substitute), (translate), (trunc), (unique_string), (unquote), and (verify) command functions.
(system_info)

**Purpose**
This command function returns information about object identifiers (OIDs).

**Syntax**

```
(system_info oid [-module module_name])
```

**Explanation**
The (system_info) command function returns information about OIDs.

The `oid` argument specifies a scalar, row, or table value for one or more OIDs. The command function searches the system kernel for information about the specified OIDs.

The `-module` argument specifies the module whose kernel (system_info) searches for OID information. By default, (system_info) searches the current module.

Table 1-3 lists scalar numeric values that may appear in the output.

<table>
<thead>
<tr>
<th>Type</th>
<th>Format</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>Integer</td>
<td>Unsigned</td>
<td>Formatted as an unsigned decimal number.</td>
</tr>
<tr>
<td>Integer</td>
<td>Hex (unsigned)</td>
<td>Formatted as a hexadecimal number.</td>
</tr>
<tr>
<td>Integer</td>
<td>All other integer values</td>
<td>Formatted as a signed decimal number.</td>
</tr>
<tr>
<td>Strings</td>
<td>(N/A)</td>
<td>Returned as strings.</td>
</tr>
<tr>
<td>Binary</td>
<td>IP addresses</td>
<td>Formatted with the <code>inet_ntop</code> function (that is, in numeric format).</td>
</tr>
<tr>
<td>Binary</td>
<td>All other binary values</td>
<td>Formatted as $2N$ hex digits with no spacing.</td>
</tr>
</tbody>
</table>

If the value specified for the `oid` argument is a table, the resulting scalar values are concatenated, separated by a space, in output. If the resulting scalar values exceed the maximum length supported by the command processor, the values are truncated. Specify a table value for the `oid` argument only when the table is a fixed size or is small.
Related Information

See OpenVOS STREAMS TCP/IP Administrator’s Guide (R419) and OpenVOS STREAMS TCP/IP Programmer’s Guide (R420).
Purpose
This command function returns the name of the current system.

Syntax
(system_name)

Explanation
The (system_name) command function returns the name of the current system.

Example
For example, if you specify display_line (system_name), the command function might return s1.

Related Information
See also the descriptions of the (current_module), (master_disk), (module_info), (module_name) and (online) command functions.
Purpose
This command function returns specific information about the current terminal.

Syntax
(terminal_info key)

Explanation
The (terminal_info) command function returns various kinds of terminal-specific information, depending on the value of key. If the process does not have a terminal port, the command function returns a null string. The following table shows the allowed values for key and the information returned by each value.
(terminal_info)

<table>
<thead>
<tr>
<th>Key</th>
<th>Returned Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>baud_rate</td>
<td>The speed at which data is transmitted to the terminal</td>
</tr>
<tr>
<td>continue_chars</td>
<td>The character(s) displayed to indicate a continued line</td>
</tr>
<tr>
<td>cursor_format</td>
<td>The format of the cursor; possible values are blinking_block,</td>
</tr>
<tr>
<td></td>
<td>blinking_underline, steady_block, steady_underline, and off</td>
</tr>
<tr>
<td>device_type</td>
<td>The name of the device driver for this terminal; possible values are</td>
</tr>
<tr>
<td></td>
<td>terminal, window_term, or vterm</td>
</tr>
<tr>
<td>escape_char</td>
<td>The escape character</td>
</tr>
<tr>
<td>flow_off_char</td>
<td>The character used to stop a stream of output to the terminal</td>
</tr>
<tr>
<td>flow_on_char</td>
<td>The character used to resume a stream of output to the terminal</td>
</tr>
<tr>
<td>line_length</td>
<td>The number of characters in a line on the screen</td>
</tr>
<tr>
<td>pause_chars</td>
<td>The message displayed on the pause line</td>
</tr>
<tr>
<td>pause_lines</td>
<td>The number of lines displayed on the screen before a pause message</td>
</tr>
<tr>
<td>prompt_chars</td>
<td>The message displayed to prompt for terminal input</td>
</tr>
<tr>
<td>screen_size</td>
<td>The number of lines displayed on the screen</td>
</tr>
<tr>
<td>tabs</td>
<td>The column positions where tabs are set for the terminal type</td>
</tr>
<tr>
<td>type</td>
<td>The terminal type, such as v105. Use the list_terminal_types command to list</td>
</tr>
<tr>
<td></td>
<td>all possible values of type.</td>
</tr>
</tbody>
</table>

**Related Information**

For more information, see the description of the (terminal_name) command function and the set_terminal_parameters command.
**Purpose**
This command function returns the full path name of the login terminal.

**Syntax**

```
(terminal_name)
```

**Explanation**
The `terminal_name` command function returns the full path name of the login terminal. If the process does not have a terminal port, the command function returns a null string. If the process is a noninteractive process, the command function returns a null string.

**Example**
The following example illustrates the use of the `terminal_name` command function in a command.

```
update_channel_info (terminal_name)
```

**Related Information**
For more information, see the description of the `terminal_info` command function and the `set_terminal_parameters` command.
Purpose
This command function returns the current time in a 24-hour format.

Syntax

(time [time_string][-long][-standard])

Explanation
The (time) command function returns the current time in 24-hour format of the form hh:mm:ss. However, if you specify the -long argument, the time returned is in 12-hour format of the form hh:mm followed by either am or pm.

Note: When using the 12-hour time format, you must specify pm to indicate that the hour is between noon and midnight. Otherwise, the returned time value is am.

The time_string argument represents a time to be returned; if you omit it, the (time) command function returns the current time. Note that any integer form of the input time string must include a colon to separate different elements of the time string.

If you specify the -standard argument, the (time) command function returns a time, and accepts a time_string argument, in the form defined in the system>configuration>languages.tin file.

The time_string argument can use any of the following keywords.

coming
absolute_time
relative_terms
time_zone

These keywords are described in Table 1-5.

For most time zones, this command function returns accurate values through December 31, 2048 (2048-12-31_23:59:59_gmt). However, because some local time zones can differ from GMT by up to 13 hours, the last full day for which this command function returns an accurate value is 2048-12-30 for those time zones.
Example

Example 1.
The value of `(time)` could be 15:25:45. The value of `(time -long)` could be 3:25 pm EST. The following example illustrates the use of the `(time)` command function in a command.

```
print log_file -defer_until (time +1 hour)
```

Example 2.
When the `(time)` command function executes with `time_zone` as an argument, the value returned is based on the current time in your own time zone. The value `(time time_zone)` returns is the time it will be in your own time zone when the time in the time zone specified by `time_zone` is the current time. The following example illustrates the `(time)` command function using `time_zone(pst, Pacific Standard Time)` as an argument issued in the Eastern Standard Time zone.

```
ready  14:36:55
display_line (time pst)
  17:37:03
ready  14:37:03
```

The first ready prompt is the current time in Eastern Standard Time. The command function returns 17:37:03, which is the time it will be in Eastern Standard Time when the current time is 14:37:03 in Pacific Standard Time.

Example 3.
The following example illustrates the use of the `(time)` command function in the Eastern Standard Time zone using the upper limit date.

```
ready  11:16:18
display_line (time 2048-12-31_23:59:59_gmt)
  19:59:59
ready  11:16:53
```

The following example illustrates what happens when the upper limit is exceeded by a second. If the upper limit is exceeded by more than a second, the system displays: Invalid date or time.

```
ready  11:16:53
display_line (time 2049-01-01_00:00:00_gmt)
  time: Invalid date or time.
ready  11:17:16
```

Related Information
See also the descriptions of the `(date)` and `(date_time)` command functions. For a list of the time zones supported by OpenVOS, see the `set_time_zone` command later in this manual.
(translate)

Purpose
This command function replaces one set of characters with another in a string.

Syntax
(translate \( S_1 \ [S_2 \ [S_3]\] \))

Explanation
The (translate) command function returns the character string \( S_1 \) translated according to the characters in strings \( S_2 \) and \( S_3 \). Each occurrence of a character in \( S_3 \) that is also in the string \( S_1 \) is replaced by a character in \( S_2 \) corresponding to the same position in \( S_3 \).

If \( S_3 \) is longer than \( S_2 \), spaces are added on the right of \( S_2 \) until the length of \( S_2 \) equals that of \( S_3 \).

If you omit \( S_3 \), \( S_3 \) is assumed to be the ASCII collating sequence.

If you omit both \( S_2 \) and \( S_3 \), \( S_2 \) is assumed to be the list of characters with ranks of 96 through 126 in the ASCII character set (for example, the lowercase letters), and \( S_3 \) is assumed to be the list of characters with ranks of 64 through 94 in the ASCII character set (for example, the uppercase letters). In this case, the (translate) command function is used to transpose lowercase letters to uppercase.

If \( S_1 \) is the null string, the result is the null string.

Example
For example, if you specify (translate aabbcc.ddeeff ABCDEF abcdef), the command function returns AABBCC.DDEEFF.

Related Information
See also the descriptions of the (after), (before), (break), (concat), (copy), (count), (index), (length), (ltrim), (quote), (reverse), (rtrim), (search), (string), (substitute), (substr), (trunc), (unique_string), (unquote), and (verify) command functions.
(trunc)

**Purpose**
This command function returns the integer part of a specified number.

**Syntax**
(trunc N)

**Explanation**
The (trunc) command function returns the integer part of N.

**Examples**
For example, if you specify (trunc 1.5), the command function returns 1. If you specify (trunc -1.5), the command function returns -1.

**Related Information**
See also the descriptions of the (after), (before), (break), (concat), (copy), (count), (index), (length), (ltrim), (quote), (reverse), (rtrim), (search), (string), (substitute), (substr), (translate), (unique_string), (unquote), and (verify) command functions.
(unique_string)

Purpose
This command function returns a unique character string.

Syntax
(unique_string)

Explanation
The (unique_string) command function returns a unique character string.

Example
For example, if you specify display_line (unique_string), the command function may return the value _aaaaabbbckooxxxx.

Related Information
See also the descriptions of the (after), (before), (break), (concat), (copy), (count), (index), (length), (ltrim), (quote), (reverse), (rtrim), (search), (string), (substitute), (substr), (translate), (trunc), (unquote), and (verify) command functions.
(unquote)

**Purpose**
This command function returns a string with any leading and trailing apostrophes removed and with any doubled apostrophes within the string reduced to single apostrophes.

**Syntax**
```
(unquote S)
```

**Explanation**
The `unquote` command function returns the string `$S$` with any leading and trailing apostrophes removed and with any doubled apostrophes within the string reduced to single apostrophes. The command processor removes one level of apostrophes before passing the argument to the command function.

**Example**
For example, if you specify `(unquote ''aa bb'' )`, the command function returns `aa bb`.

**Related Information**
See also the descriptions of the `after`, `before`, `break`, `concat`, `copy`, `count`, `index`, `length`, `ltrim`, `quote`, `reverse`, `rtrim`, `search`, `string`, `substitute`, `substr`, `translate`, `trunc`, `unique_string`, and `verify` command functions.
Purpose
This command function returns your current user name.

Syntax
(user_name)

Explanation
The (user_name) command function returns your current user name.

Example
The following example illustrates the use of the (user_name) command function in a command.

```
list_users (user_name)
  * Smith.Sales
```

Related Information
See also the description of the (group_name) and (person_name) command functions.
Purpose
This command function returns the leftmost position of the character in a string that is not in a specified set.

Syntax
(verify S C)

Explanation
The (verify) command function returns the leftmost position of the character in the string \( S \) that is not in the set \( C \). If all of the characters in \( S \) are in \( C \), the returned value is 0.

Example
For example, if you specify (verify aabbcc.ddeeff af), the command function returns 3.

Related Information
See also the descriptions of the (after), (before), (break), (concat), (copy), (count), (index), (length), (ltrim), (quote), (reverse), (rtrim), (search), (string), (substitute), (substr), (translate), (trunc), (unique_string), and (unquote) command functions.
(vos_path)

Purpose

This command function converts a POSIX path name into an OpenVOS path name.

Syntax

(vos_path path_name)

Explanation

The (vos_path) command function converts the path_name argument, which is a relative or full POSIX path name, into a full OpenVOS path name. The resultant OpenVOS path name always begins with a percent-sign character (%).

If path_name has the form /dev/null, the result is %sys#null, where %sys is the current system. Otherwise, path_name is expanded into a full OpenVOS path name, which processes and removes any dot components (.. or ..).

Example

The following examples assume that the master disk is %s1#d01 and the current directory is %s1#d01>SysAdmin.

\[\text{display_line (vos_path Sales/Jones)} \]
\ [%s1#d01>SysAdmin>Sales>Jones

\[\text{display_line (vos_path /Sales/Jones)} \]
\ [%s1#d01>Sales>Jones

\[\text{display_line (vos_path ../Sales/Jones)} \]
\ [%s1#d01>Sales>Jones

\[\text{display_line (vos_path /dev/null)} \]
\ [%s1#null

\[\text{display_line (vos_path /)} \]
\ [%s1#d01

The following example assumes that the current directory is %s1#d02>Research.

\[\text{display_line (vos_path .)} \]
\ [%s1#d02>Research
Related Information

See also the description of the (posix_path) command function.
**Purpose**

This command function returns specific information about the location of an object.

**Syntax**

```
(where_path path_name [type] [chase_code])
```

**Explanation**

The `(where_path)` command function returns the full path name, full module name, or object type of the object specified by `path_name`.

The value for the argument `type` can be `-path_name`, `-object_type`, or `-module_name`. If you omit `type`, the value `-path_name` is used.

The value of `(where_path)` is determined as follows.

- When `path_name` is a link, the information returned by `(where_path)` is related to the ultimate target of the link, rather than to the link itself (unless you specify `-no_chase`).
- When `type` is `-path_name`, `(where_path)` returns the full path name of the object specified by `path_name`.
- When `type` is `-object_type`, `(where_path)` returns the object type of the object specified by `path_name`. The object types that can be returned are `device` for an I/O device, `file` for a file, `directory` for a directory, and `link` for a link (if you specify `-no_chase`).
- When `type` is `-module_name`, `(where_path)` returns the name of the system and module containing the object specified by `path_name`.

The value for the argument `chase_code` can be `-chase` or `-no_chase`. The `-no_chase` value returns information about a link itself rather than its target when `path_name` is a link.
Examples

The following are examples of the use of the \texttt{(where\_path)} command function.

<table>
<thead>
<tr>
<th>Command Function</th>
<th>Returned Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>(where_path smith_memos)</td>
<td>%s1#d01&gt;Sales&gt;Smith&gt;smith_memos</td>
</tr>
<tr>
<td>(where_path smith_memos -object_type)</td>
<td>file</td>
</tr>
<tr>
<td>(where_path smith_memos -module_name)</td>
<td>%s1#d01</td>
</tr>
<tr>
<td>(where_path smith_memos -object_type -no_chase)</td>
<td>link</td>
</tr>
</tbody>
</table>

Related Information

See the descriptions of the \texttt{(current\_dir)}, \texttt{(directory\_name)}, \texttt{(home\_dir)}, \texttt{(object\_name)}, \texttt{(path\_name)}, and \texttt{(system\_name)} command functions. See also the description of the \texttt{where\_path} command.
Date and Time Keywords

Table 1-5 shows the keywords that can be used in the input strings for the (date), (iso_date), (date_time), (iso_date_time), and (time) command functions. If you use the -standard argument with any of these command functions, additional values for the keywords, if defined, are accepted when your process has another language set (see the set_language command).

Table 1-5. Date and Time Input Keywords

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>absolute_date</td>
<td>A date in one of the input formats that the (date_time) or (iso_date_time) command function accepts. An absolute date string can be partial or complete. A complete date string specifies a year, month, and day. A partial date is one that can be interpreted unambiguously, such as friday, which is assumed to be Friday in the current week, or 1/1, which is the first day of the first month of the current year.</td>
</tr>
<tr>
<td>absolute_time</td>
<td>A portion of the date/time string that the (date_time) or (iso_date_time) command function accepts as representing the time in hours and minutes. It can be complete or partial.</td>
</tr>
<tr>
<td>coming</td>
<td>A keyword specifying that the input date/time is in the future</td>
</tr>
<tr>
<td>relative_terms</td>
<td>A string in the form +N specifier or -N specifier, where N is an integer and specifier is any of the terms listed below. The value of N indicates the number of units by which the output date/time is to be offset. The allowed specifier values are as follows: sec, secs, second, seconds, min, mins, minute, minutes, hr, hrs, hour, hours, day, days, wk, wks, week, weeks, mo, mos, month, months, yr, yrs, year, years.</td>
</tr>
<tr>
<td>time_zone</td>
<td>The portion of the date/time string that expresses the time zone.</td>
</tr>
</tbody>
</table>
Chapter 2:
OpenVOS User and Programming Commands

This chapter describes the commands available to OpenVOS users.

Note that it is possible that all users at your site may not have access to all commands described in this chapter. If you are unable to invoke a particular command you need, you should contact your system administrator to discuss obtaining access to that command.

**Note:** Where this manual refers to *sequential files*, it also refers to *extended sequential files*, unless otherwise noted. The two are identical except that the maximum record size, which is meaningless for normal sequential files, indicates record offset unit size for extended sequential files, thereby allowing a greater growth potential. For more information about extended sequential files, see the description of the `create_file` command.

**Note:** Where this manual refers to *stream files*, it also refers to *64-bit stream files*, unless otherwise noted. The two are identical except that 64-bit stream files can be sparse; they cannot contain indexes, cannot be pipe files, and cannot be SAE files; and their size can grow beyond 2 GB. For more information about 64-bit stream files, see the description of the `create_file` command.
add_entry_names

Purpose
This command looks for all the entry points in an object library and creates links to the object modules that contain those entry points. These links are necessary for binding programs that call entry points within the object modules. The binder locates the object modules containing the entry points when it searches the object library.

Display Form

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>object_path_name</td>
<td>current_dir</td>
</tr>
<tr>
<td>-unlink_first</td>
<td>yes</td>
</tr>
<tr>
<td>-entry_links</td>
<td>yes</td>
</tr>
<tr>
<td>-ext_data_links</td>
<td>no</td>
</tr>
<tr>
<td>-long</td>
<td>no</td>
</tr>
</tbody>
</table>

Command Line Form

```
add_entry_names [object_path_name]
    [-no_unlink_first]
    [-no_entry_links]
    [-ext_data_links]
    [-long]
```

Arguments

- `object_path_name`
  The path name of a single object module or of a directory containing object modules whose entry points are to be linked. If you give the file name of an object module, the suffix `.obj` is optional. By default, the command operates on object modules in your current directory.

- `-no_unlink_first`
  Retains all existing links to the entry points of the specified object module or modules, and creates any additional links if necessary. By default, the command unlinks all links created by previous `add_entry_names` commands and creates a new set of links.

- `-no_entry_links`
  Adds entry points without creating links to all the object modules in a library. By default, the command creates the links.
**add_entry_names**

- **-ext_data_links**
  Creates links to all external data names defined by the specified object module or modules. This argument applies *only* to OpenVOS C programs. By default, the command does not create links to external data names.

- **-long**
  Lists each link and its target as it is created. By default, the command does not list the links it creates.

**Explanation**

The `add_entry_names` command creates a link name with the suffix `.obj` for each entry name defined in the specified object module or modules. The command can create links to the entries defined by a single object module as well as for all the object modules in a directory (also called a library).

A compiler creates an object module whose file name has the suffix `.obj`. The object module, however, may contain many different entry points, and other object modules may contain external references to any of these entry points. Also, OpenVOS C programs may have modules defining data only. To bind object modules into a program module, the binder must find a definition for every entry point and external data definition referenced in the program module.

To resolve such external references, the binder searches directories for path names that correspond to each external reference in the object modules being bound together. To resolve an external reference in one object module to an external name defined by another, the binder must find a name of the form `entry.obj` in the object library. Thus, an external name must be identified by a link to an object module, or by the object module itself.

The `add_entry_names` command adds the name of each entry point in an object module as an additional name on the object module file. It adds a link to the object module using the name of the entry point. This command should be run on any directory that is to be included in the object library search paths. The binder searches each directory in the object library search list by looking for an object module with the same name as the unresolved entry name. This command guarantees that all entry names are visible to the binder.

The `-unlink_first` argument removes obsolete links. For example, when you replace an object module or object modules, a replacement module might have fewer external definitions than the existing version, or some object modules might have been deleted from the library. Specifying the `-no_unlink_first` argument in cases where no obsolete links exist can speed processing; for example, when you add a new object module to a library, or know that no external names have been deleted from any object modules currently in the library.

When the `object_path_name` argument specifies a directory, the `-unlink_first` argument unlinks at the start of processing all links whose targets are object modules in the specified directory. Next, the command finds and unlinks all links whose targets are object modules in other directories. The command then creates new links for external names defined by object modules in the directory `object_path_name`. Finally, it creates links in the directory `object_path_name` for external names defined by object modules in other directories.
When the `object_path_name` argument specifies a single object module, the
-`unlink_first` argument unlinks all links to the object module in the directory containing
it. Then the command creates links in that directory for external names defined by the current
version of the module.

The `-entry_links` and `-ext_data_links` arguments control the type of external data
definition that causes a link to be created. The `-entry_links` argument applies to all
languages; an entry name that differs from an object module name causes a link to be created.
The `-ext_data_links` argument applies to OpenVOS C and OpenVOS Standard C object
modules only; an external data definition that differs from the object module name causes a
link to be created. Note that message code symbols (those names beginning with `e$`, `m$`, `q$`,
and `r$`) do not cause links to be created. The binder treats these symbols as a special case,
and they do not require links.

**Access Requirements**

You need read access to the object modules and modify access to the containing directory to
create the links.

**Examples**

Consider a command macro named `install_runtime.cm` that updates the directory
>system>object_library from the master copies of the object modules in various >ldd
directories. It then updates >system>c_object_library. Finally, it executes
add_entry_names for each directory in order to guarantee that the links are up-to-date.

```plaintext
&begin_parameters
    module    module: module_name,req,=''
    debug     switch(-debug),=0
&end_parameters
&
&if ^ &debug&
&then &goto BEGIN
&
&mode no_execute
&echo command_lines macro_lines
&
&label BEGIN
&set_string release_base >ldd
&set_string object_library (master_disk &module&) >system>object_library
&
&if (exists &object_library& -directory -chase)
&then &goto OBJECT_LIB_EXISTS
&
&display_line &object_library& does not exist.
&return
&
&label OBJECT_LIB_EXISTS

(Continued on next page)
(Continued)

!copy_file &release_base&>basic>runtime>obj>* &object_library& -delete &
!copy_file &release_base&>cobol>runtime>obj>* &object_library& -delete
!copy_file &release_base&>fms>runtime>obj>* &object_library& -delete
!copy_file &release_base&>fortran>runtime>obj>* &object_library& -delete
!copy_file &release_base&>lang>runtime>obj>* &object_library& -delete
!copy_file &release_base&>pascal>runtime>obj>* &object_library& -delete
!copy_file &release_base&>pl1>runtime>obj>* &object_library& -delete
!copy_file &release_base&>tp>runtime>obj>* &object_library& -delete &
!copy_file &release_base&>c>runtime>obj>* &+ (master_disk &module&)>system>c_object_library -delete &
!copy_file &release_base&>runtime>obj>* &object_library& -delete &
!add_entry_names &object_library& -unlink_first
!add_entry_names (master_disk &module&)>system>c_object_library &+ -unlink_first -ext_data_links
add_library_path

Purpose
This command adds path names to the list of directories that define a specified library.

Display Form

```
add_library_path
```

Command Line Form

```
add_library_path library_name
  library_path_names...
  [-before existing_library_path_name]
  [-after existing_library_path_name]
  [-first]
  [-no_check]
  [-ignore_duplicates]
```

Arguments

- **library_name** [CYCLE] Required
  The name of a library to which the path name of a directory is to be added. There are four possible values for `library_name`.
  - include
  - object
  - command
  - message

  By default, the command adds the directory to the `include` library.

- **library_path_names** Required
  One or more directory path names to be added to the library. The path names can include the command functions `(current_dir)` or `(home_dir)`. If `library_name` is `message`, the path names can also include `(referencing_dir)` and
(language_name). If enclosed in apostrophes, the command functions are evaluated when the path name is used.

- **-before existing_library_path_name**
  Inserts `library_path_names` in the library list before the specified `existing_library_path_name`. You cannot use the `-after` or `-first` arguments if you select `-before`. By default, the command adds the directories to the end of the list.

- **-after existing_library_path_name**
  Inserts `library_path_names` in the library list after the specified `existing_library_path_name`. You cannot use the `-before` or `-first` arguments if you select `-after`. By default, the command adds the directories to the end of the list.

- **-first**
  Inserts `library_path_names` at the beginning of the library list. You cannot use the `-before` or `-after` arguments if you select `-first`. By default, the command adds the directories to the end of the list.

- **-no_check**
  Omits checking the existence of the ultimate target of each path name specified for `library_path_names`. By default, the command checks that the ultimate target of each name is an existing directory.

- **-ignore_duplicates**
  Ignores any library path names that are already present in the list of current library path names. The command adds the remaining path names that are not duplicates. By default, the command diagnoses attempts to add a library path name that is already present and does not add any path names. The command checks for an attempt to add a duplicate path name but does not check that the specified path name is in the appropriate position.

**Explanation**

The `add_library_path` command allows you to add path names anywhere in the list of directories that define a specified library.

A *library* is set of directories that the operating system searches for objects of a particular type. Each module has the following libraries.

- include library
- object library
- command library
- message library

The compilers search the include library for include files; the binder searches the object library for object modules; the command processor searches the command library for commands and the message library for message files associated with individual commands.

A *library* is defined by an ordered sequence of path names. The order of the list reflects the order in which the operating system searches the directories of a library.
For each library, the search for an object begins in the first directory on the library list. If the object is not in that directory, the search proceeds to the second directory on the list, then to the third, and so on. The module’s default list of directories for each library serves as the guide to where to find objects. The add_library_path command enables you to insert path names anywhere in the list of directories for a given library, so you can control the order in which the operating system searches for an object. Use the -before, -after, or -first argument to determine where in the list to place the additional path names (you can select only one). If you use none of those arguments, path names are added to the end of the list. A directory cannot appear twice on the library list, so to reorder an existing directory, use the delete_library_path command first, or use the set_library_paths command.

If you specify the -before, -after, or -check argument, add_library_path checks the specified library path name to see if it exists. If you specify the -no_check argument and do not specify the -before or -after argument, add_library_path does not check for the existence of the specified library path name; instead, the command checks that the library path name has the correct format for a directory.

The path name of the directory library_path_names can include the command functions (current_dir) or (home_dir); if library_name is message, the path name can also include (referencing_dir) and (language_name). (Note that you must enclose a command function in apostrophes in order to prevent its evaluation by the command processor when the add_library_path command is executed.)

The list of libraries defined by the add_library_path command remains in effect only for the life of your process.

**Examples**

Suppose you use the list_library_paths command to list your include library and the following information is displayed.

```
include library directories:
  (current dir)
  %s1#d03>Sales>incl
  %s1#d04>system>include_library
```

To add your own work directory, execute the following command.

```
add_library_path include work -before >system>include_library
```

The system expands the relative path name. Now if you list your include library, the following information is displayed.

```
include library directories:
  (current dir)
  %s1#d03>Sales>incl
  %s1#d03>Sales>Smith>work
  %s1#d04>system>include_library
```

**Related Information**

For information about other commands that can affect libraries, see delete_library_path, list_library_paths, and set_library_paths in this
manual. See also add_default_library_path, delete_default_library_path, and list_default_library_paths in the OpenVOS System Administration: Administering and Customizing a System (R281). See the OpenVOS Commands User’s Guide (R089) for information about search rules.
add_profile

Purpose
This command adds the profile information from two profile data files, accumulating the sum in the second profile data file. This command is useful when multiple runs of the same program are required to meter or test a program fully.

Display Form

```
COMMAND LINE FORM
---------------------------------- add_profile ---------------------------------
profile_file_name: [output_file_name]
```

Command Line Form

```
add_profile profile_file_name [output_file_name]
```

Arguments

- **profile_file_name** Required
  The path name of a profile file. Profile files can have either a fixed or stream file organization. The name is the same as that of the program module compiled with `-ql` or `-qc` (for an OpenVOS Standard C program) or with `-profile` or `-cpu_profile` (for a program compiled with one of the other OpenVOS compilers), with the `.pm` suffix replaced by `.profile`.

- **output_file_name**
  The path name of another profile file from the same program module as `profile_file_name`. It can also be the name of a non-existent file. If you provide the name of a non-existent file, the command copies the file named by `profile_file_name` to the file named by `output_file_name`. By default, the system generates a name with the same name as `profile_file_name`, with the `.profile` suffix replaced by `.sum.profile`.

Explanation

The `add_profile` command adds all of the counts, CPU times, and page fault values on a statement-by-statement basis, in the `profile_file_name` and `output_file_name` files, and writes the result to `output_file_name`.

Each profile file must have been made from the same version of the bound program, and must represent the same type of profile.
If an output file with the name `output_file_name` does not exist, it is created and the `profile_file_name` file is copied to it. The resulting profile can be analyzed by the `profile` command.

**Note:** If you stop `add_profile` command execution by typing <Control>-c, you should consider the contents of the output file data invalid since the command will not have finished adding the execution counts.

**Access Requirements**

You need read access to the profile files you specify as `profile_file_name` and `output_file_name`, and modify access to the current directory.

**Examples**

This example shows how to use the `profile` and `add_profile` commands to examine the execution history of the following `sample.pl1` source module. Depending on input from the user, this source module either converts a set of Celsius temperatures to Fahrenheit or a set of Fahrenheit temperatures to Celsius. In performing either conversion, only part of the source code is executed, and therefore, the contents of the `.profile` file is different for each type of temperature conversion. The `add_profile` command enables you to combine these two different `.profile` files.

```
1 sample:
2   procedure;
3
4   declare   fahrenheit float bin;
5   declare celsius   float bin;
6   declare degrees fixed bin;
7   declare option   fixed bin;
8
9   put edit ('Type 1 (Celsius to Fahrenheit) or 2 (Fahrenheit to Celsius): ')
10   (a (61));
11   get list (option);
12
13   if option = 1
14     then do;
15     put edit ('   CELSIUS', ' FAHRENHEIT')
16     (a (12), x (1), a (11));
17     put skip;
18
19     do degrees = 0 to 100 by 10;
20       celsius = degrees;
21       fahrenheit = 9 * celsius / 5 + 32;
22       put edit (celsius, fahrenheit)
23         (f (10), x (1), f (11, 1));
24     put skip;
25   end;
26 end;
27
(Continued on next page)
```
else if option = 2
then do;
   put edit (' FAHRENHEIT', 'CELSIUS')
       (a (14), x (2), a (10));
   put skip;
   do degrees = 0 to 212 by 20;
      fahrenheit = degrees;
      celsius = 5 / 9 * (fahrenheit - 32);
      put edit (fahrenheit, celsius)
          (f (11), x (1), f (10, 1));
   put skip;
   end;
end;
end sample;

Perform the following steps to combine and examine the execution histories for the sample.pl1 source module.

1. Compile the sample.pl1 source code using the -profile or -cpu_option option.

   pl1 sample -profile

2. Bind the sample object module.

   bind sample

3. Execute the sample.pm program module and when prompted, type 1 to convert a set of Celsius temperatures to Fahrenheit. When you execute the sample.pm program module, it generates a sample.profile file that contains the execution history for the Celsius to Fahrenheit conversion.

   sample

   Type 1 (Celsius to Fahrenheit) or 2 (Fahrenheit to Celsius) 1

   CELSIUS    FAHRENHEIT
   0          32.0
   10         50.0
   20         68.0
   30         86.0
   40        104.0
   50        122.0
   60        140.0
   70        158.0
   80        176.0
   90        194.0
   100       212.0
4. Rerunning the sample.pm program module creates another sample.profile file which will overwrite the first sample.profile file. Rename the first sample profile file sample.sum.profile. You can rename the file in one of two ways.

    move_file sample.profile sample.sum.profile
    add_profile sample.profile

5. Execute the sample.pm program module again and when prompted, type 2 to convert a set of Fahrenheit temperatures to Celsius. When you execute the sample.pm program module, it generates a sample.profile file that contains the execution history for the Fahrenheit to Celsius conversion.

    sample
    Type 1 (Celsius to Fahrenheit) or 2 (Fahrenheit to Celsius) 2
    FAHRENHEIT   CELSIUS
    0             -17.8
    20            -6.7
    40            4.4
    60            15.6
    80            26.7
    100           37.8
    120           48.9
    140           60.0
    160           71.1
    180           82.2
    200           93.3

6. Issue the add_profile command to add the sample.pm execution histories contained in the sample.sum.profile file (for Celsius to Fahrenheit conversions) and the sample.profile file (for Fahrenheit to Celsius conversions).

    add_profile sample.profile sample.sum.profile
7. Issue the `profile` command to generate a `sample.sum.plist` file from the `sample.sum.profile` file and then issue the `display_file` command to display the contents of the `sample.sum.plist` file.

```sh
profile sample.sum.profile
display_file sample.sum.plist
```

Profile of: sample

<table>
<thead>
<tr>
<th>STATEMENT</th>
<th>COUNT</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>2</td>
</tr>
<tr>
<td>9</td>
<td>2</td>
</tr>
<tr>
<td>11</td>
<td>2</td>
</tr>
<tr>
<td>13</td>
<td>2</td>
</tr>
<tr>
<td>15</td>
<td>1</td>
</tr>
<tr>
<td>17</td>
<td>1</td>
</tr>
<tr>
<td>19</td>
<td>1</td>
</tr>
<tr>
<td>20</td>
<td>11</td>
</tr>
<tr>
<td>21</td>
<td>11</td>
</tr>
<tr>
<td>22</td>
<td>11</td>
</tr>
<tr>
<td>24</td>
<td>11</td>
</tr>
<tr>
<td>25</td>
<td>11</td>
</tr>
<tr>
<td>26</td>
<td>1</td>
</tr>
<tr>
<td>28</td>
<td>1</td>
</tr>
<tr>
<td>30</td>
<td>1</td>
</tr>
<tr>
<td>32</td>
<td>1</td>
</tr>
<tr>
<td>34</td>
<td>1</td>
</tr>
<tr>
<td>35</td>
<td>11</td>
</tr>
<tr>
<td>36</td>
<td>11</td>
</tr>
<tr>
<td>37</td>
<td>11</td>
</tr>
<tr>
<td>39</td>
<td>11</td>
</tr>
<tr>
<td>40</td>
<td>11</td>
</tr>
<tr>
<td>43</td>
<td>2</td>
</tr>
</tbody>
</table>

TOTALS: 128

**Related Information**

See also the `profile` command.
analyze_pc_samples

Purpose
This command analyzes the program counter (PC) raw data file created by the harvest_pc_samples command and generates a statistical report on the execution of one or more specified program modules.

Display Form

```
- analyze_pc_samples
  pc_sample_file_paths:
  - partition_number: 2
  - master_disk:
  - pm_file_path:
  - output_path: pc_analysis.current_date_and_time
  - program_names: *
  - process_names: *
  - process_ids:
  - modules_only: no
  - percent_format: absolute
  - ignore_percent_below: 0.01
  - transactions_per_second:
  - line_number_analysis: yes
  - hex_granularity: 16
  - start_code_ref: 0
  - end_code_ref: 2147483647
  - pm_offset_address:
```
analyze_pc_samples

Command Line Form

```
analyze_pc_samples pc_sample_file_paths path_name...

- partition_number number
- master_disk module_name
- pm_file_path path_name
- output_path path_name
- program_names name...
- process_names name...
- process_ids id...
- modules_only
- percent_format
- ignore_percent_below percent
- transactions_per_second number
- no_line_number_analysis
- hex_granularity byte_size
- start_code_ref integer
- end_code_ref integer
- pm_offset_address offset
```

Arguments

- **pc_samples_file_paths path_name...**
  The path names of one or more raw data files generated by the harvest_pc_samples command. These are the files that you want to analyze with the analyze_pc_samples command. The path names cannot contain extended names.

- **-partition_number**
  Specifies the boot partition number of the OpenVOS release executing while the harvest_pc_samples command was running. The default is the boot partition of the module on which you issued the analyze_pc_samples command. If you give both this argument and the -pm_file_path argument, the command ignores the value you supply for this argument.

- **-master_disk module_name**
  Specifies the name of the system module on which you issued the harvest_pc_samples command. The default is the master disk of the module on which you issued the analyze_pc_samples command. If you give both this argument and the -pm_file_path argument, the command ignores the value you supply for this argument.

- **-pm_file_path path_name**
  Specifies the path name of a program module. You can specify only one program module, such as vos.pm or test.pm. If you specify a value for this argument, the command ignores the -partition_number and -master_disk arguments. The path name cannot be an extended name. For a discussion of the relationship of this argument to the -program_names, -process_names, and -process_ids arguments, see the Explanation section.

- **-output_path path_name**
  Specifies the path name of the file that will contain the report generated by the command. The path name must not be an extended name. The default name of the file is pc_analysis.current_date_and_time.
-program_names name...
  Specifies one or more program names. Program names cannot be extended names. The
  command will only examine program counter samples collected from processes
  running these programs. An example of a program name is office.pm. If you were to
  specify that value, the command would analyze all processes running Office. For a
  discussion of the relationship of this argument to the -pm_file_path,
  -process_names, and -process_ids arguments, see the Explanation section.

-process_names name...
  Specifies one or more process names. The command will only examine program
  counter samples collected from these processes. An example of a process name is
  testing. If you were to specify that value, the command would analyze only the
  testing process. For a discussion of the relationship of this argument to the
  -program_names, -pm_file_path, and -process_ids arguments, see the
  Explanation section.

-process_ids id...
  Specifies one or more process IDs. The command will only examine program counter
  samples collected from these processes. An example of a process ID is 01002001x. If
  you were to specify that value, the command would analyze only the process
  01002001x. For a discussion of the relationship of this argument to the
  -program_names, -process_names, and -pm_file_path arguments, see the
  Explanation section.

-modules_only
  Specifies that the command generate the list of program modules that are contained in
  the PC raw data file. For each program module, the command calculates the percentage
  of total samples. In addition, if you specify a value for the
  -transactions_per_second argument, the command calculates the length of each
  transaction (in microseconds) in each module. For a detailed description of the output
  generated when you specify this argument, see the Sample Output section.

-percent_format
  Specifies which method the analyze_pc_samples command uses to report the total
  number of hits per program module function or statement. You can specify a
  cumulative or absolute report. If you specify cumulative, the command adds the
  previous line’s total percentage to the current line’s percentage. If you specify
  absolute, the command reports the total number of hits per program module function
  or statement as an absolute percentage of the total number of hits. The default value is
  absolute. For a detailed description of the output generated when you specify this argument, see the Sample Output section.

-ignore_percent_below percent
  Specifies the smallest percentage value that the command includes in the output data.
  Use this argument to remove statistically insignificant data from the report. If you
  specify a value of cumulative for the -percent_mode argument, the command
  ignores any value you specify for this argument. The default value is 0.01 percent.

-transactions_per_second number
  Specifies the number of transactions per second processed by the target program
  module while the harvest_pc_samples command was running. The
analyze_pc_samples command uses this value to estimate the execution time (in microseconds) of each program module (associated with the target program module) that helps process one transaction. For a sample of the output generated when you specify this argument, see the Sample Output section. An example of a value for this argument is 22.57. Typically, a program such as TPC-A or TPC-B returns a transactions per second value that you can then specify as a value for this argument. If you do not specify the -modules_only argument, the command ignores any value that you specify for this argument.

**Note:** A transaction is a sequence of operations that is performed as a unit. Typically, a transaction involves updating one or more pieces of data in a database. If your target program module does not update a database or perform similar transaction-oriented processing, specifying a value for this argument will not provide any useful information.

- **-line_number_analysis**
  Specifies whether the command uses the available source code line number information. If you specify yes, the command includes line numbers in the report when they are available. If line numbers are not available (as is the case with assembly language program modules), the command includes hexadecimal byte offsets in the output. The default value is yes. If you specify the -modules_only argument, the command ignores any value that you specify for this argument.

- **-hex_granularity byte_size**
  Specifies a byte value that changes the byte interval between assembly language program module analyses shown in the report. The specified byte value determines how much address space is blocked in the same bucket. The default value is 16 bytes. If you specify the -modules_only argument, the command ignores any value that you specify for this argument.

- **-start_code_ref integer**
  Specifies the record number in the raw data file at which the command will begin data analysis. If you specify a record number for this argument, the command ignores all the data before the record number. The default value is 0.

- **-end_code_ref integer**
  Specifies the record number in the raw data file at which the command will stop data analysis. If you specify a record number for this argument, the command ignores all the data after the record number. The default value is 2147483647, which is the largest signed integer value supported by OpenVOS.

- **-pm_offset_address offset**
  Specifies the entry point for a relocatable program module located in the OpenVOS kernel (these program modules are listed in >system>kernel_loadable_library). Enter the address offset at which the target program module was executed. You can determine the proper offset by using the dump_eit (dump executable image table) request in the analyze_system command on the same module on which you executed the harvest_pc_samples command. You can find this value in the request’s base vm at: output field. For a description of the dump_eit request, see the OpenVOS System Analysis Manual (R073).
Explanation

The `analyze_pc_samples` command reads the raw data file created by the `harvest_pc_samples` command and selects information about one or more specified program modules. It then computes statistics about the location and quantity of program counter samples that reference the specified program module and places the results in a file. See the Sample Output section for a description of the contents of this file.

The `-pm_file_path`, `-program_names`, `-process_names`, or `-process_ids` arguments let you focus the analysis on OpenVOS, on an application program module, or on the relationship of an application program module to OpenVOS. Figure 2-1 illustrates each of these cases.

![Diagram of OpenVOS program modules in kernel and user address space]

**Figure 2-1. Analyzing Program Modules in the Kernel and User Address Space**

A To analyze the OpenVOS program module, do not specify a value for the `-program_names`, `-process_names`, or `-process_ids` arguments, since OpenVOS is the only program module which runs in the kernel address space. By default, the command reads the OpenVOS program module directly from the boot partition of the disk specified in `-master_disk`. If you copied the OpenVOS program module from a boot partition to a file, specify the path name of the file in `-pm_file_path`.

B To analyze only a user program module, specify the program module’s name (for example, `office.pm`) as the value for both the `-pm_file_path` and the `-program_names` arguments. Note that instead of specifying a value for the `-program_names` argument, you can specify the value of the process name or ID (`-process_names` or `-process_ids` arguments) that executes the program module.

**Note:** If you specify the program module name as the `-pm_file_path`, but fail to specify a corresponding `-program_name`, `-process_name`, or `-process_id` value, the `analyze_pc_samples` command analyzes the program counter samples from all user processes in the user address space. This happens because all user programs execute in the same virtual address space. The command needs a `-program_name`, `-process_name`, or `-process_id` value in order to analyze a particular target program module in the user address space.

C To analyze the relationship of a user program module to OpenVOS, follow the steps in A above, and specify the value of the `-program_names` argument as the name of the
analyze_pc_samples

target program module (for example, test.pm). Note that instead of specifying a value for the -program_names argument, you can specify the value of the process name or ID (-process_names or -process_ids arguments) that executes the program module.

In case B or C, you can specify redundant values for the -process_names, -process_ids, and -program_names arguments. For example, if the command module foo.cm calls the program module test.pm, you could specify foo.cm as the value of the -process_names argument and test.pm as the value of the -program_names argument.

Examples

The following examples show several different ways of specifying the arguments to the analyze_pc_samples command.

Example 1. In this example, the raw data file is called pc_samples, and the output file is called target.analysis. To analyze the execution of a target program module in the OpenVOS user address space, both the -program_names and the -pm_file_path arguments are specified with the same program module name. The analyze_pc_samples command will generate a default report. For an example of this type of report, see the Sample Output section.

    analyze_pc_samples pc_samples
    -output_path target.analysis
    -program_names target.pm
    -pm_file_path target.pm

Example 2. This example differs from the previous example in that it uses two raw data files, which must have been created on the same machine and at the same sampling frequency. Analyzing more than one raw data file lets you increase the accuracy of the analysis statistics by concatenating two or more data files.

    analyze_pc_samples pc_samples1 pc_samples2
    -output_path target.analysis
    -program_names target.pm
    -pm_file_path target.pm

Example 3. In this example, the analyze_pc_samples command examines all the program modules executing on OpenVOS (using the current module’s OpenVOS boot partition as the target program module) and generates a default report.

    analyze_pc_samples pc_samples
    -output_path vos.analysis

Example 4. In this example, the analyze_pc_samples command examines the portions of OpenVOS exercised by two target program modules executing in the OpenVOS kernel address space and generates a default report.

    analyze_pc_samples pc_samples
    -output_path vos.analysis
    -program_names user1.pm user2.pm
Example 5. In this example, the `analyze_pc_samples` command examines a target program module executing in the OpenVOS user address space and generates a report containing a list of program modules and an estimated transaction time for each program module. For an example of this type of report, see the Sample Output section. The command ignores functions that are executed less than one percent of the time, and uses a value of 28 transactions per second to estimate the execution time of each program module that helps process the transaction.

```bash
analyze_pc_samples pc_samples
-output_path target.analysis
-pm_file_path target.pm
-program_names target.pm
-modules_only
-ignore_percent_below 1
-transactions_per_second 28
```

Example 6. This example differs from the previous example in that the `analyze_pc_samples` command examines all the program modules executing on OpenVOS. The command generates a report containing a list of program modules and an estimated transaction time for each program module.

```bash
analyze_pc_samples pc_samples
-output_path vos.analysis
-modules_only
-ignore_percent_below 1
-transactions_per_second 28
```

Sample Output

The output file produced by `analyze_pc_samples` contains the following information:

- Specified, default, and environmental values used by the `analyze_pc_samples` command to generate the report

- A summary of each raw data file used to generate the report. This summary includes the following:
  - specified, default, and environmental values used by the `harvest_pc_samples` command to create the raw data file
  - a list of all the processes and associated programs running on OpenVOS during the period that the `harvest_pc_samples` command logged data
  - a list of all the processes that executed different programs during the period that the `harvest_pc_samples` command logged data. These processes failed `harvest_pc_samples` program validation.
  - a summary of the statistics and CPU usage contained in the raw data file

- If more than one raw data file is used to generate the report, the report adds the statistics and CPU usage information from each raw data file and presents the cumulative information.
analyze_pc_samples

- A detailed execution report. The default is a line, module, and function execution report that shows the number of samples (hits) taken per source code line, program module name, and function name. An alternative module summary report is generated if you specify the -modules_only argument. This alternative report lists the percentage of hits per program module.

**Note:** Unless you bind the specified program module with the -retain_all argument of the bind command, the analyze_pc_samples command divides its output only by program module (source file) names. If you also want the analyze_pc_samples command to divide its output by function names and source code line numbers, execute the bind command to rebind the program module with the -retain_all argument set to yes before running the harvest_pc_samples and analyze_pc_samples commands.

The following sections describe examples of each section of the report.

The **analyze_pc_samples Command Argument and Environmental Values Section**

This section of the report provides information about the analysis, including the time of analysis, the version of analyze_pc_samples, the list of files that were analyzed, and the specified analyze_pc_samples argument values.

<table>
<thead>
<tr>
<th>Analysis Time</th>
<th>91-08-02 11:13:20 EDT</th>
</tr>
</thead>
<tbody>
<tr>
<td>analyze_pc_samples version</td>
<td>1.1</td>
</tr>
<tr>
<td>PC Analysis of</td>
<td>%es#d01&gt;Guest&gt;User&gt;new_sampler&gt;vos_11.pm</td>
</tr>
<tr>
<td>PM Offset Address</td>
<td>0x0</td>
</tr>
<tr>
<td>PC Sample File(s)</td>
<td>%es#d01&gt;Guest&gt;User&gt;new_sampler&gt;example.samples</td>
</tr>
<tr>
<td>Options</td>
<td>-program_name(s) : *</td>
</tr>
<tr>
<td></td>
<td>-process_name(s) : harvest_pc_samples</td>
</tr>
<tr>
<td></td>
<td>-process_id(s) :</td>
</tr>
<tr>
<td></td>
<td>-modules_only : no</td>
</tr>
<tr>
<td></td>
<td>-percent_format : cumulative</td>
</tr>
<tr>
<td></td>
<td>-ignore_percent_below : 0.01</td>
</tr>
<tr>
<td></td>
<td>-transactions_per_second :</td>
</tr>
<tr>
<td></td>
<td>-line_number_analysis : yes</td>
</tr>
<tr>
<td></td>
<td>-hex_granularity : 16</td>
</tr>
<tr>
<td></td>
<td>-start_code_ref : 0</td>
</tr>
<tr>
<td></td>
<td>-end_code_ref : 2147483647</td>
</tr>
<tr>
<td></td>
<td>-pm_offset_address :</td>
</tr>
</tbody>
</table>
Raw Data File Summary Section

This section of the report provides information about each raw data file used by the analyze_pc_samples command. This section of the report consists of four subsections.

The harvest_pc_samples Command Argument and Environmental Values Subsection

This subsection lists the specified, default, and environmental values used by the harvest_pc_samples command to create the raw data file.

<table>
<thead>
<tr>
<th>Statistics from</th>
<th>%es#d01&gt;Guest&gt;User&gt;new_sampler&gt;example.samples</th>
</tr>
</thead>
<tbody>
<tr>
<td>Harvest Version</td>
<td>1.1</td>
</tr>
<tr>
<td>VOS Version</td>
<td>VOS Release 11.p</td>
</tr>
<tr>
<td>Number of CPU's</td>
<td>1</td>
</tr>
<tr>
<td>Requested Freq.</td>
<td>85 samples/sec/cpu</td>
</tr>
<tr>
<td>Timing Jitter</td>
<td>2</td>
</tr>
<tr>
<td>Program Validation</td>
<td>30 sec</td>
</tr>
</tbody>
</table>

The Process List Subsection

This subsection lists the processes and associated programs running on OpenVOS during the period that the harvest_pc_samples command logged data in the raw data file.

<table>
<thead>
<tr>
<th>PROCESS LIST</th>
</tr>
</thead>
<tbody>
<tr>
<td>PROCESS ID</td>
</tr>
<tr>
<td>--------------</td>
</tr>
<tr>
<td>0x01113062</td>
</tr>
<tr>
<td>0x01113046</td>
</tr>
<tr>
<td>0x01112001</td>
</tr>
<tr>
<td>0x0111200C</td>
</tr>
<tr>
<td>0x01113060</td>
</tr>
<tr>
<td>0x011120D5</td>
</tr>
<tr>
<td>0x011120D1</td>
</tr>
<tr>
<td>0x01112030</td>
</tr>
<tr>
<td>0x011120FB</td>
</tr>
<tr>
<td>0x011120F1</td>
</tr>
<tr>
<td>0x011120F9</td>
</tr>
<tr>
<td>0x01112011</td>
</tr>
<tr>
<td>0x01112004</td>
</tr>
</tbody>
</table>
The Changed Program Names (Program Validation) Subsection

This subsection lists the processes that executed different programs during the period specified by the -program_validation_period argument of the harvest_pc_samples command.

<table>
<thead>
<tr>
<th>PROCESS ID</th>
<th>PROCESS NAME</th>
<th>PROGRAM NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x011120D5</td>
<td>login</td>
<td>list_users.pm</td>
</tr>
<tr>
<td>0x011120ED</td>
<td>login</td>
<td>login</td>
</tr>
<tr>
<td>0x011120D5</td>
<td>login</td>
<td>emacs.pm</td>
</tr>
<tr>
<td>0x01113061</td>
<td>login</td>
<td>copy_kernel.pm</td>
</tr>
<tr>
<td>0x0111120F1</td>
<td>login</td>
<td></td>
</tr>
<tr>
<td>0x0111120FB</td>
<td>login</td>
<td></td>
</tr>
<tr>
<td>0x011120F1</td>
<td>login</td>
<td>xemacs.pm</td>
</tr>
<tr>
<td>0x011120FB</td>
<td>login</td>
<td>emacs.pm</td>
</tr>
<tr>
<td>0x01113061</td>
<td>login</td>
<td>office.pm</td>
</tr>
<tr>
<td>0x01113060</td>
<td>login</td>
<td></td>
</tr>
<tr>
<td>0x01113061</td>
<td>login</td>
<td>list_users.pm</td>
</tr>
<tr>
<td>0x01113060</td>
<td>login</td>
<td>emacs.pm</td>
</tr>
<tr>
<td>0x01113061</td>
<td>login</td>
<td></td>
</tr>
</tbody>
</table>

The PC Sample Statistics Summary and CPU Usage Subsection

This subsection provides a summary of the statistics and CPU usage derived from the raw data file.

<table>
<thead>
<tr>
<th>PC SAMPLE STATISTICS</th>
<th>CPU USAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total Code References : 25554</td>
<td>User CPU Time : 45.19 sec</td>
</tr>
<tr>
<td>OS Code References : 24814</td>
<td>System CPU Time : 30.04 sec</td>
</tr>
<tr>
<td>User Code References : 740</td>
<td>Server CPU Time : 7.51 sec</td>
</tr>
<tr>
<td>Target Region Refs : 55</td>
<td>Interrupts Time : 48.83 sec</td>
</tr>
<tr>
<td>Target Refs Missed PM : 7</td>
<td>Idle Time : 131.29 sec</td>
</tr>
<tr>
<td>Target Refs Hit PM : 48</td>
<td>Total CPU Time : 262.86 sec</td>
</tr>
<tr>
<td>PM Hits / Total Refs : 0.19 %</td>
<td></td>
</tr>
<tr>
<td>Harvest Duration : 300.02 sec</td>
<td></td>
</tr>
<tr>
<td>Actual Sampling Freq. : 85.18 samp/sec</td>
<td></td>
</tr>
</tbody>
</table>
The following fields appear in the PC sample statistics subsection.

**Total Code References:**
The total number of program counter samples contained in the raw data file(s).

**OS Code References:**
The total number of program counter samples in the raw data file(s) that map to OpenVOS kernel address space. For an illustration of the OpenVOS kernel address space, see Figure 2-1.

**User Code References:**
The total number of program counter samples in the raw data file(s) that map to the OpenVOS user address space. For an illustration of the OpenVOS user address space, see Figure 2-1.

**Target Region Refs:**
The total number of program counter samples used in the analysis. This value is equal to the total number of program counter samples associated with processes having the specified `-program_name`, `-process_name`, or `-process_id` value. It is also equal to the sum of the Target Refs Hit PM and the Target Refs Missed PM fields. If you did not specify a value for the `-program_name`, `-process_name`, or `-process_id` arguments, the value of this field is the same as Total Code References.

**Target Refs Missed PM:**
The total number of program counter samples in the target region which did not map into the address space of the target program module. If this number is not close to zero, either you specified a target program module name for `-pm_file_path`, but did not specify a value for the `-program_name`, `-process_name`, or `-process_id` arguments, or the target program module only executed for part of the time that the `harvest_pc_samples` command was executing. For a description of the need to specify a value for the `-program_name`, `-process_name`, or `-process_id` arguments for user target program modules, see the description of part B of Figure 2-1.

**Target Refs Hit PM:**
The total number of program counter samples in the target region which mapped into the address space of the target program module.

**PM Hits / Total Refs:**
The value of the Target Refs Hit PM field divided by the value of the Total Code References field.

**Harvest Duration:**
The length of time that the `harvest_pc_samples` command executed. This number should be equal to the specified `harvest_pc_samples` duration value. If the difference between the Harvest Duration and the specified duration is more than one second, `harvest_pc_samples` may not have been running at a high enough priority.
Actual Sampling Freq.:

The value of the Total Code References divided by the value of the Harvest Duration field. This number should be equal to the specified harvest_pc_samples -sampling_frequency times the number of logical processors on the system on which you ran harvest_pc_samples. If the difference between the Actual Sampling Freq. and the specified -sampling_frequency value is greater than one or two, the harvest_pc_samples command may have been running at too low a priority.

The following fields appear in the CPU usage subsection.

User CPU Time:
The amount of CPU time used by user processes during the execution of harvest_pc_samples.

System CPU Time:
The amount of CPU time used by OpenVOS processes during the execution of harvest_pc_samples.

Server CPU Time:
The amount of CPU time used by the StrataNET network server processes during the execution of harvest_pc_samples.

Interrupts Time:
The amount of CPU time spent handling interrupts during the execution of harvest_pc_samples.

Idle Time:
The amount of CPU time in which no processes ran during the execution of harvest_pc_samples.

Total CPU Time:
The sum of all the above times.

\[
\text{Total CPU Time} = \text{User CPU Time} + \text{System CPU Time} + \text{Server CPU Time} + \text{Interrupts Time} + \text{Idle Time}
\]

This number should be equal to Harvest Duration times the number of logical processors on the system on which the harvest_pc_samples command ran. However, discrepancies in metering sometimes cause the two values to be quite different.
Sample File Statistics Summary Section
If you specify more than one raw data file as input to the analyze_pc_samples command, the command generates this section. It contains the sum of the PC Sample Statistics and CPU Usage statistics from all the raw data files.

<table>
<thead>
<tr>
<th>SAMPLE FILE STATISTICS SUMMARY</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>PC SAMPLE STATISTICS</strong></td>
</tr>
<tr>
<td>Total Code References : 1210842</td>
</tr>
<tr>
<td>OS Code References : 1209170</td>
</tr>
<tr>
<td>User Code References : 1672</td>
</tr>
<tr>
<td>Target Region Refs : 1210842</td>
</tr>
<tr>
<td>Target Refs Missed PM : 3358</td>
</tr>
<tr>
<td>Target Refs Hit PM : 1207484</td>
</tr>
<tr>
<td>PM Hits / Total Refs : 99.72 %</td>
</tr>
<tr>
<td>Harvest Duration : 1200.02 sec</td>
</tr>
</tbody>
</table>

The Line, Module, and Function Statistics Section
By default, the analyze_pc_samples command generates sampling statistics by source code line number, program module, and function name. In this example, the value of the -percent_format argument has been specified as cumulative. Descriptions of the columns in this example are given in Table 2-1.

**Note:** If you specified the value of the -line_number_analysis as no, the output would look the same as this example, except that all line number references would be hexadecimal offsets from the beginning of the program module address space.

<table>
<thead>
<tr>
<th>LINE#</th>
<th>HITS</th>
<th>%TOTAL</th>
<th>MODULE NAME</th>
<th>FUNCTION NAME</th>
<th>HITS/FN</th>
</tr>
</thead>
<tbody>
<tr>
<td>359</td>
<td>1</td>
<td>2.08</td>
<td>cache_manager</td>
<td>cache_manager</td>
<td>1</td>
</tr>
<tr>
<td>5498</td>
<td>3</td>
<td>8.33</td>
<td>cache_manager</td>
<td>cm_post</td>
<td>2</td>
</tr>
<tr>
<td>5500</td>
<td>1</td>
<td>10.42</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>475</td>
<td>1</td>
<td>12.50</td>
<td>cache_manager_utils</td>
<td>seq_get_block</td>
<td>1</td>
</tr>
<tr>
<td>484</td>
<td>1</td>
<td>14.58</td>
<td>disk Allocators</td>
<td>withdraw_cate_pages</td>
<td>1</td>
</tr>
<tr>
<td>969</td>
<td>1</td>
<td>16.67</td>
<td>disk Allocators</td>
<td>withdraw_file_pages_mdn0</td>
<td>2</td>
</tr>
</tbody>
</table>
analyze_pc_samples

<table>
<thead>
<tr>
<th>Address</th>
<th>Frequency</th>
<th>Function Name</th>
<th>Description</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>1486</td>
<td>1</td>
<td>sched_queue_utils</td>
<td>lock_sched_q</td>
<td>2</td>
</tr>
<tr>
<td>486</td>
<td>2</td>
<td>give_up_cpu_i</td>
<td>thread_ready</td>
<td>1</td>
</tr>
<tr>
<td>169</td>
<td>1</td>
<td>lock_i</td>
<td>unlock</td>
<td>1</td>
</tr>
<tr>
<td>134</td>
<td>1</td>
<td>sim_interrupt</td>
<td>timer_runout</td>
<td>1</td>
</tr>
<tr>
<td>428</td>
<td>1</td>
<td>sim_interrupt</td>
<td>rel_sim_int_check_real</td>
<td>1</td>
</tr>
<tr>
<td>364</td>
<td>1</td>
<td>update_load_meters</td>
<td>meter_pf</td>
<td>1</td>
</tr>
<tr>
<td>260</td>
<td>1</td>
<td>pc_sampler</td>
<td>s$$get_pc_sample</td>
<td>3</td>
</tr>
<tr>
<td>275</td>
<td>2</td>
<td>get_utils_i</td>
<td>rscaneq</td>
<td>1</td>
</tr>
<tr>
<td>125</td>
<td>1</td>
<td>get_lock_word</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>20</td>
<td>1</td>
<td>rbreak</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>89</td>
<td>1</td>
<td>mirrored_spin_lock_i</td>
<td>unlock_mirrored_spin_loc</td>
<td>1</td>
</tr>
<tr>
<td>72</td>
<td>1</td>
<td>ass_even_to_vcs_pop</td>
<td>assign_even_to_vcs</td>
<td>1</td>
</tr>
<tr>
<td>56</td>
<td>1</td>
<td>index_chars_1_pop</td>
<td>index_chars_1</td>
<td>1</td>
</tr>
<tr>
<td>A0h</td>
<td>1</td>
<td>mod_int_pop</td>
<td>mod_int</td>
<td>1</td>
</tr>
<tr>
<td>B0h</td>
<td>1</td>
<td>move_halfwords_pop</td>
<td>move_halfwords</td>
<td>1</td>
</tr>
<tr>
<td>61</td>
<td>1</td>
<td>scanne_1_pop</td>
<td>scanne_1</td>
<td>2</td>
</tr>
<tr>
<td>63</td>
<td>1</td>
<td>scanne_1</td>
<td></td>
<td>2</td>
</tr>
<tr>
<td>130h</td>
<td>1</td>
<td>ventnor_interrupt</td>
<td>ventnor_interrupt</td>
<td>4</td>
</tr>
<tr>
<td>4C0h</td>
<td>1</td>
<td>ventnor_waf_and_sis</td>
<td>kernel_trap_handler</td>
<td>1</td>
</tr>
<tr>
<td>540h</td>
<td>1</td>
<td>ventnor_fast_timer</td>
<td>s$$read_jiffy_clock</td>
<td>1</td>
</tr>
<tr>
<td>550h</td>
<td>1</td>
<td>ventnor_meter</td>
<td>meter2_ssq_completion</td>
<td>1</td>
</tr>
<tr>
<td>50h</td>
<td>1</td>
<td>ventnor_switch_process</td>
<td>switch_process</td>
<td>3</td>
</tr>
<tr>
<td>250h</td>
<td>3</td>
<td>file_io</td>
<td></td>
<td>3</td>
</tr>
<tr>
<td>3220</td>
<td>1</td>
<td>process_control</td>
<td>s$$get_process_meters</td>
<td>1</td>
</tr>
<tr>
<td>1959</td>
<td>1</td>
<td>dir_status</td>
<td>s$match_star_name</td>
<td>2</td>
</tr>
<tr>
<td>2036</td>
<td>1</td>
<td>file_io</td>
<td>file_io</td>
<td>1</td>
</tr>
<tr>
<td>3777</td>
<td>1</td>
<td>file_io</td>
<td>write_raw_file</td>
<td>3</td>
</tr>
<tr>
<td>8314</td>
<td>1</td>
<td>file_io</td>
<td></td>
<td>3</td>
</tr>
<tr>
<td>8885</td>
<td>1</td>
<td>file_io</td>
<td></td>
<td>3</td>
</tr>
<tr>
<td>D0h</td>
<td>3</td>
<td>kernel_trap_i</td>
<td>validate_write_ptr</td>
<td>1</td>
</tr>
</tbody>
</table>

2-28
Table 2-1 describes the columns in the preceding example.

**Table 2-1. Line, Module, and Function Statistics Columns of analyze_pc_samples**

<table>
<thead>
<tr>
<th>Column</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LINE#</td>
<td>Either the source code line number (displayed without a trailing letter h) in the source file specified in MODULE NAME, or the hexadecimal byte offset from the beginning of the assembly language routine named in FUNCTION NAME.</td>
</tr>
<tr>
<td>#HITS</td>
<td>The number of samples collected at the line number or hexadecimal offset specified in LINE#. To obtain an approximate idea of the accuracy of this value in terms of a percentage, divide the square root of the value by the value, multiply the quotient by 100, and subtract the product from 100. For example, if four hits were harvested on a line, then the value is approximately 50% accurate ((100 - ((\sqrt{4} / 4) * 100))). If 1600 hits were harvested on a line, then the value is approximately 98% accurate ((100 - ((\sqrt{1600} / 1600) * 100))).</td>
</tr>
<tr>
<td>%TOTAL</td>
<td>As specified by -percent_format cumulative, each value in this column contains the cumulative total percentage of sample hits measured up to that point. To determine the absolute percentage of sample hits measured for a function or module, subtract the cumulative percentage on the previous line from the cumulative percentage for the function or module. For example, if module B has a cumulative percentage of 2.0, and module C has a cumulative percentage of 6.0, the absolute percentage for module C is C’s percentage minus B’s percentage, or 4.0.</td>
</tr>
<tr>
<td>MODULE NAME</td>
<td>The module name from the module map. The module map lists all of the files in a program. Note that binding with -retain_all preserves the module map.</td>
</tr>
<tr>
<td>FUNCTION NAME</td>
<td>The entry name from the entry map. The entry map lists all of the functions in a program. Note that binding with -retain_all preserves the entry map.</td>
</tr>
<tr>
<td>HITS/FN</td>
<td>The total number of samples collected from the function.</td>
</tr>
</tbody>
</table>
Program Module Summary Statistics Section

If you specify the value of the -modules_only argument as yes, the command generates a program module summary statistics section instead of the default line, module, and function statistics section. The module summary section lists the program module names and the percentage of total hits for each module. This section is shorter and much less detailed than the alternative line, module, and function statistics section.

In addition, for this example, a value for the -transactions_per_second argument is specified. If you do not specify a value for this argument, the command cannot calculate the MICROSECONDS PER TRANS. column and does not display it.

<table>
<thead>
<tr>
<th>MICROSECONDS</th>
<th>PERCENT OF</th>
<th>TOTAL REFS</th>
<th>MODULE NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>4995 +/- 36</td>
<td>42.55</td>
<td>idle_process</td>
<td></td>
</tr>
<tr>
<td>160 +/- 9</td>
<td>1.37</td>
<td>cache_manager</td>
<td></td>
</tr>
<tr>
<td>242 +/- 10</td>
<td>2.06</td>
<td>scheduling</td>
<td></td>
</tr>
<tr>
<td>254 +/- 11</td>
<td>2.16</td>
<td>sched_queue_utils</td>
<td></td>
</tr>
<tr>
<td>335 +/- 12</td>
<td>2.86</td>
<td>give_up_cpu_i</td>
<td></td>
</tr>
<tr>
<td>129 +/- 8</td>
<td>1.10</td>
<td>lock_i</td>
<td></td>
</tr>
<tr>
<td>184 +/- 9</td>
<td>1.57</td>
<td>sim_interrupt</td>
<td></td>
</tr>
<tr>
<td>203 +/- 10</td>
<td>1.73</td>
<td>mirrored_spin_lock_i</td>
<td></td>
</tr>
<tr>
<td>319 +/- 12</td>
<td>2.72</td>
<td>ventnor_interrupt</td>
<td></td>
</tr>
<tr>
<td>361 +/- 13</td>
<td>3.07</td>
<td>ventnor_waf_and_sis</td>
<td></td>
</tr>
<tr>
<td>229 +/- 10</td>
<td>1.95</td>
<td>ventnor_fast_timer</td>
<td></td>
</tr>
<tr>
<td>163 +/- 9</td>
<td>1.39</td>
<td>ventnor_meter</td>
<td></td>
</tr>
<tr>
<td>251 +/- 11</td>
<td>2.14</td>
<td>ventnor_cpu_hw_info_1</td>
<td></td>
</tr>
<tr>
<td>878 +/- 19</td>
<td>7.48</td>
<td>ventnor_switch_process</td>
<td></td>
</tr>
<tr>
<td>119 +/- 7</td>
<td>1.01</td>
<td>kernel_trap_i</td>
<td></td>
</tr>
<tr>
<td>151 +/- 8</td>
<td>1.28</td>
<td>sysdb_man</td>
<td></td>
</tr>
<tr>
<td>8972</td>
<td>76.42</td>
<td>sum of modules listed above</td>
<td></td>
</tr>
</tbody>
</table>

Note: The last line of output, sum of modules listed above, displays the total number of microseconds per transaction spent by the modules in the list, and the percent of total hits generated by all of the modules in the list.
Table 2-2 describes the columns in the preceding example.

**Table 2-2. Module Summary Statistics Columns of analyze_pc_samples**

<table>
<thead>
<tr>
<th>Column</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MICROSECONDS PER TRANS.</td>
<td>An estimate of the microseconds spent by each program module processing a transaction. The command also calculates the standard deviation (+/-) of this value. In this case, 66 percent of the time, the value in this column is within +/- X of the real value. Figure 2-2 illustrates the standard deviation. To increase confidence in a value, collect more samples by rerunning the harvest_pc_samples command at a faster sampling rate or for a longer period of time.</td>
</tr>
<tr>
<td>PERCENT OF TOTAL REFS</td>
<td>The percent of the total samples collected from a module.</td>
</tr>
<tr>
<td>MODULE NAME</td>
<td>The module name corresponding to the values calculated in the first two columns.</td>
</tr>
</tbody>
</table>

**Figure 2-2** shows an example of a standard deviation. Standard deviation assumes a bell curve distribution of data points. Most data points occur close to the mean; the further away from the mean, the less likely an event is to occur. If the mean execution time for a module is 119 milliseconds, and the standard deviation is +/- 7 milliseconds, then, by definition, the shaded area under the bell curve marked by the lower and upper standard deviation values is 66 percent of the area under the bell curve. In this example, this means that there is only a 33 percent chance that time spent by the program module on the transaction is either less than 112 milliseconds or greater than 126 milliseconds.

**Figure 2-2. Example of Standard Deviation**
Related Information

For information on how to create the raw data file that this command analyzes, see the description of the `harvest_pc_samples` command in this manual. For information on related performance measuring tools, see the descriptions of the `profile` and `add_profile` commands in this manual.
attach_default_output

Purpose

This command attaches your default_output port to a specified file or I/O device.

Display Form

<table>
<thead>
<tr>
<th>attach_default_output</th>
<th>path_name: default_output.out</th>
</tr>
</thead>
<tbody>
<tr>
<td>-append: no</td>
<td></td>
</tr>
</tbody>
</table>

Command Line Form

```
attach_default_output [path_name]

[-append]
```

Arguments

- `path_name`
  - The path name of a file or output device. By default, the command attaches your
    default_output port to the file `default_output.out` in your current directory,
    creating it if it does not already exist.

- `-append`
  - Appends the output to the file `path_name`. By default, the command truncates the file
    before writing to it. If `path_name` is a device, `-append` is ignored.

Explanation

The `attach_default_output` command attaches the default_output port to the file or
output device `path_name`. You can nest up to eight `attach_default_output` commands.
(Each will use a different `path_name`.)

For an interactive process, the output of all commands you invoke is written to the
default_output port, which is connected to your terminal. For a batch process, the output
port is attached to a file, and the output of commands executed by the batch processor is
written to the file. When you invoke the `batch` command, any command output that normally
appears on the terminal is written to `path_name`, and does not appear on the terminal.

If you specify `path_name`, but it is not an existing file or output device, the
`attach_default_output` command creates a sequential file with that path name and
attaches the default_output port to it. In this case, the `-append` argument is redundant.
If you specify `path_name`, and it is an existing file, the `attach_default_output` command attaches the `default_output` port to the file. In this case, if you do not specify the `-append` argument, the command truncates the file before writing to it. If you specify the `-append` argument, the command appends the output to the contents of the file.

**Access Requirements**

You must have write access to an existing file or output device to specify it as `path_name`. By default, you have write access to all devices, unless your system administrator has restricted that access.

**Examples**

**Example 1.**

The `attach_default_output` command creates a file named `default_output.out` in your current directory, if none exists, and attaches your `default_output` port to it. If a file with that name already exists, the `attach_default_output` command truncates it first and then writes the output to the file. To append data to the file, use the `-append` argument.

In this example, the command writes the names of the objects in a directory to a file.

```
attach_default_output jones_customers
list %s1#d02>Sales>Jones>customers -all
detach_default_output
```

The first command attaches your `default_output` port to the file `jones_customers` in your current directory. The second lists the contents of the directory `%s1#d02>Sales>Jones>customers`. Because `default_output` is attached to a file, you do not see the output from the `list` command. The last command detaches `default_output` from the file and reattaches the port to its previous attachment, in this case your terminal.

**Example 2.**

In the previous example, you could have attached your `default_output` port to a printer by using the following commands.

```
attach_default_output %s1#p1.48
list %s1#d02>Sales>Jones>customers -all
detach_default_output
```

**Related Information**

See the description of the `detach_default_output` command for information about how to detach the `default_output` port from the current file or output device and reattach it to the original default output attachment. See the `list_devices` command for a list of output devices to which you can attach your `default_output` file. You can use only the `attach_default_output` and `detach_default_output` commands to change the attachment of your process’s `default_output` port; it is an error to use the `attach_port` and `detach_port` commands. See also the descriptions of the `attach_port`, `detach_port`, `start_logging`, and `stop_logging` commands.
attach_port

Purpose
This command creates and names an I/O port and attaches it to a file or I/O device you specify.

Display Form

<table>
<thead>
<tr>
<th>port_name:</th>
<th>path_name:</th>
</tr>
</thead>
<tbody>
<tr>
<td>-hold_open:</td>
<td>no</td>
</tr>
<tr>
<td>-high_port:</td>
<td>no</td>
</tr>
</tbody>
</table>

Command Line Form

```
attach_port port_name
    path_name
        [-hold_open]
        [-high_port]
```

Arguments

- **port_name**  
  The name of the new port.  
  
- **path_name**  
  The path name of the file or I/O device to which the command is to attach the port.  
  
- **-hold_open**  
  Keeps the attached port open after the termination of the program doing I/O through the port. By default, the command closes the port after the program terminates.  
  
- **-high_port**  
  Assigns the new port from the high port range. By default, the new port is assigned from the low port range. See the description of the `s$attach_port` subroutine for more information on low and high port ranges.

Explanation

The `attach_port` command creates and names a port in your process and attaches the port to a file or an I/O device. You can then refer to the file or device through the port. The port remains attached until you issue `detach_port`, or until your process stops.

For information about the maximum number of ports per module, see Appendix B, “General OpenVOS Software Limits and Numerical Definitions.”
The following tape commands can implicitly attach a port if you do not first explicitly attach a port with the `attach_port` command.

- `write_tape`
- `read_tape`
- `list_tape`
- `copy_tape`
- `save`
- `save_object`
- `restore`
- `restore_object`
- `list_save_tape`

**Access Requirements**

By default, you have write access to a tape device. If your system administrator restricts access to the tape device, you need read access to read from tapes, or write access to read from and write to tapes.

**Examples**

**Example 1.**
The following command attaches the port `magtape` to the tape drive `tape.1.0`.

```
attach_port magtape %s1#tape.1.0
```

(Use the `list_devices` command to list the names of I/O devices, such as tape drives.)

Programming languages have conventions for resolving references to files and devices in programs. If you create and name a port that represents a file or device, and execute a program that references that port, the operating system uses the port name as an open connection to the file or device.

**Example 2.**
Suppose you attach the port `my_file` to the file `>Sales>Jones>reports>this_week` with the following command.

```
attach_port my_file >Sales>Jones>reports>this_week
```

Use the name `my_file` in a program to refer to a file. When you execute the program, a reference to the port named `my_file` will be a reference to the file `>Sales>Jones>reports>this_week`. You can detach the port named `my_file` and attach it to another file before processing a second file with the program.

**Related Information**

See the description of the `detach_port` and `attach_default_output` commands.
**batch**

**Purpose**
This command places a request in a batch queue.

**Display Form**

```
command_line: batch
-process_name:
-output_path: 
-process_priority: 4
-privileged: no
-restart: yes
-queue: normal
-module:
-current_dir: current_dir
-defer_until:
-control:
-after:
-cpu_limit:
-notify: no
```

**Command Line Form**

```
batch command_line
    [-process_name process_name]
    [-output_path output_path_name]
    [-process_priority process_priority]
    [-queue_priority queue_priority]
    [-privileged]
    [-no_restart]
    [-queue queue_name]
    [-module module_name]
    [-current_dir path_name]
    [-defer_until date_time]
    [-control control_file_name]
    [-after process_name...]
    [-cpu_limit cpu_time]
    [-notify]
```
Arguments

- **command_line**  
The command line to be executed by the batch process.

- **-process_name process_name**  
  Specifies the name of the batch process. By default, batch gives the process a name derived from the first word in `command_line`. If it is a valid path name, the command uses the file-name portion with any suffixes removed. If the resulting name is longer than 32 characters, the command uses the first 32 characters. If the resulting file name contains an apostrophe or is invalid, the command uses the name `batch`.

- **-output_path output_path_name**  
  Attaches the default output port of the batch process to the file or device specified by `output_path_name`. If you do not specify this argument, the default output port is attached to a file on the current directory. The command derives the name of the file from the first word in the `command_line` argument. If it is a valid path name, the command uses the file-name portion with any suffixes removed and appends the suffix `.out`. If the resulting file name is too long, the command uses the first 28 characters and appends the `.out` suffix. If the resulting file name contains an apostrophe (`) or is invalid, the command uses the name `batch.out`.

- **-process_priority process_priority**  
  Specifies the priority of the batch process. The range of process priorities is 0 to 9, with 9 being the highest. By default, the command assigns the process the same priority as your current process.

- **-queue_priority queue_priority**  
  Specifies the queue priority of the batch request. The queue priority must be in the range 0 to 9, with 9 being the highest. By default, the command assigns a queue priority of 4.

- **-privileged**  
  Makes the batch process a privileged process, which can execute privileged commands. By default, the process is unprivileged. Only a privileged process can request a privileged batch process.

- **-no_restart**  
  Discontinues the batch process if the processing module stops and then starts again. By default, the batch processor starts another batch process to execute `command_line` from its beginning when the module restarts.

- **-queue queue_name**  
  Specifies the name of the batch queue that holds the batch request until its execution. By default, the command puts your batch request in the `normal` queue, either on the module `module_name` or on the current module.

- **-module module_name**  
  Specifies the name of the module containing the specified queue. By default, the command uses the current module.
-current_dir path_name

Sets the current directory of the batch process to path_name. By default, the operating system sets the current directory of the batch process to the current directory of the calling process. Specifying a new current directory does not change the default location of the output path.

-defer_until date_time

Defers running the batch process until after date_time. The date_time term can be a character string in the standard form:

```
yy-mm-dd_hh:mm:ss
```

The term can also be a character string in any form accepted by the (date_time) command function. In this case, the string must be enclosed in apostrophes. See Chapter 1, “OpenVOS Command Functions,” for examples of acceptable date/time input strings.

By default, the batch processor starts the batch process as soon as possible.

-control control_file_name

Specifies a batch control file to control the execution of the batch process. A batch control file must have the suffix .batch, though you can omit this suffix when you specify the file name. By default, the command uses the other command arguments to determine how the batch process will run.

-after process_name

Specifies one or more process names, or star names, of requests submitted by (user_name).* It does not refer to all requests in the queue. The newly issued batch request is not executed until the processes identified by process_name have finished executing. By default, the operating system executes the batch request after executing requests that have a higher priority in the queue. In the command line form of the command, command_line must precede -after.

-cpu_limit cpu_time

Limits the amount of CPU time a batch process is allowed to consume. This value is specified as hh:mm:ss, defaulting to seconds. Thus, a value of nn is interpreted as seconds, and a value of nn:nn is interpreted as minutes and seconds. By default, the command puts no limit on the amount of CPU time the batch process can use.

-notify

Notifies you when the batch process terminates by sending a message to the status line. By default, the batch processor does not notify you when the batch process is done.

**Explanation**

The batch command puts the command line command_line in the batch queue queue_name. The batch processor starts a process to execute command_line. That executing process acquires its process priority, if unspecified, and its language from the process submitting the batch request.

If the command line contains spaces, you must enclose the line in apostrophes.
Batch requests are ordinarily added to the end of a batch queue. However, if you use the `-queue_priority` argument with a queue priority higher than that of requests already in the queue, the newly issued batch request enters the queue ahead of those requests with a lower queue priority and behind requests with the same or higher queue priority. A batch queue has 10 queue priorities, numbered 0 to 9, with 9 being the highest. When a batch process can be started, the operating system starts the request at the head of the queue, which is defined as the request whose queue priority is highest. If two or more requests in the queue have the same queue priority, the operating system starts the oldest request first.

If you specify the `-output_path` argument, the batch processor attaches the default_output port to the file or device `output_path_name`. If the file `output_path_name` already exists, the batch processor truncates it before the batch process writes to it; if the file does not exist, the batch processor creates it.

The `-after` argument has no effect on the position of the batch request in the queue, since the position of the request in the queue is determined by its queue priority. However, it allows a request with a queue priority of 6, for example, to be executed after a request with a priority of 4. The request with the priority of 6 is inserted in the queue ahead of the request with the priority of 4, but the lower priority request will execute first. Also, the `-after` argument does not refer to all batch processes or requests, but only to those submitted by `(user_name)`.

The `-restart` argument requeues your batch request if the batch process terminates without signaling process completion (for example, if a system shutdown occurs during execution). Thus, you must specify the `-no_restart` argument if you use the `batch` command to start a process to execute the `shutdown` command. (See OpenVOS System Administration: Starting Up and Shutting Down a Module or System (R282) for information on the `shutdown` command.)

Examples

Example 1.
To submit a batch request for execution of the program module `payroll.pm` to a default batch queue, issue the following command.

```
batch #accounting>pay>payroll.pm
```

The batch processor executes the program module.

Example 2.
To enter a COBOL compilation batch request in the batch queue named `io_bound`, issue the following command.

```
batch 'cobol make_reports -list' -queue io_bound
```
Example 3.
The following commands, used together, ensure that the specified files are bound after the compilation.

```
batch 'cobol make_reports'
batch 'bind make_reports weekly_rpts' -after cobol
```

Related Information
For a detailed discussion of batch processing, see the *OpenVOS Commands User’s Guide* (R089). For information about currently executing batch processes, specify the `display_batch_status` command. To list the batch process you have submitted, specify the `list_batch_requests` command. The batch processor gives each batch process a queue sequence number, which is the position of the process in the batch queue. The `list_batch_requests` command displays the queue sequence number. To cancel a batch process request, specify the `cancel_batch_requests` command. See also the command descriptions of `update_batch_requests`, `reserve_device`, `move_device_reservation`, and `cancel_device_reservation`. 
Purpose

This command binds one or more object modules into one executable program module. One of the compilers or the assembler translates a source module into an object module. You must bind the object module or set of object modules into a program module before you can execute the program.

Display Form

```
bind

object_modules:
- control:
- search:
- define:
- entry:
- load_point: default
- max_heap_size: default
- max_program_size: default
- max_stack_size: default
- pm_name:
- processor: default
- size:
- stack_fence_size: default
- stack_size: 65536
- target_module: current_module
- version: Pre-release
- align_mod16: no - define_main: no
- dynamic_tasking: yes - extended_names: default
- map: no - number_of_tasks: 1
- private_heap: no - private_stack: yes
- profile_alignment_faults: no - retain_all: no
- shared: no - statistics: no
- subroutines_are_functions: no - table: yes
```
Command Line Form

bind { object_modules...[ -control control_file_name ] ]
   -control control_file_name

[ -search [path_name...] ]
[ -define variable_name... ]
[ -entry entry_point ]
[ -load_point load_point ]
[ -max_heap_size maximum_heap_size ]
[ -max_program_size maximum_program_size ]
[ -max_stack_size maximum_stack_size ]
[ -pm_name path_name ]
[ -processor processor_string ]
[ -size size ]
[ -stack_fence_size stack_fence_size ]
[ -stack_size size ]
[ -target_module module_name ]
[ -version version ]
[ -align_mod16 ]
[ -define_main ]
[ -no_dynamic_tasking ]
[ -extended_names extended_names_string ]
[ -load_in_kernel ]
[ -map ]
[ -number_of_tasks number ]
[ -private_heap ]
[ -private_stack ]
[ -profile_alignment_faults ]
[ -references_kernel ]
[ -relocatable ]
[ -retain_all ]
[ -shared ]
[ -statistics ]
[ -subroutines_are_functions ]
[ -no_table ]

Arguments

► object_modules
   One or more path names or star names of object modules to be bound. If you do not specify object_modules, you must use -control.

   Note: Unlike object modules named in a control file, object_modules are not subject to search path rules; they must be relative or full path names.

► -control control_file_name
   Specifies a binder control file to control the binding. The name of the binder control file must have the suffix .bind, though you can omit this suffix when you specify the path name. If you do not specify -control, you must specify object_modules.
-search [path_name]

Searches a specified directory or directories for object modules named as entry points in the object modules being bound. The binder first searches any directories specified in this argument, then any specified in the binder control file, and finally the directories specified by the process’s current object library search rules. If you include -search on a command line without specifying a directory, the binder first searches the current directory. By default, the binder searches the directory master_disk>system>object_library, unless you have redefined the object library paths for your process. The binder allows as many command line and/or binder control file search directories (whether specified or default) as memory allows.

See Using OpenVOS Dynamic Linking and Shared Libraries (R648) for more information about the search rules for binding an object module or shared library.

-define variable_name

Defines variables to be used during binder control file preprocessing. Preprocessor variables can contain letters, digits, or the underline character (_), in any position. (See the Explanation section of this command description or the description of the preprocess_file command for details.)

-entry entry_point

Enables you to define a main entry point of an executable program rather than to use the first procedure in the first source module specified in the bind command.

Do not use this argument for a C program that defines a function called main. (That is, do not specify bind -entry main.) Instead, bind in the main function first.

The -entry and -define_main arguments are mutually exclusive.

-load_point load_point

Specifies the lowest address for the object modules. The value of -load_point can be any unsigned 32-bit value. For example, if the default load point for the kernel is 80000000x, specify the following command to change the load point.

    bind prog1 -load_point 80026000x

The default value is default, which is determined by the processor type and whether you are building a user or kernel program. On an ftServer module, use the default value.

-max_heap_size maximum_heap_size

Specifies the maximum byte size to which the heap can grow. Specify an integer value from 0 to 32,768. The default value is default. If no value is specified, the value of maximum_heap_size is 0.

Note: A zero value does not imply that the heap’s size is 0; instead, during binding, the bind command assumes that the maximum heap size is equal to 32,768 bytes for the purpose of checking the size of the address space, and during runtime, OpenVOS assumes that the heap size is unlimited.

-max_program_size maximum_program_size

For user programs, specifies the maximum amount of code and data the program can contain including its symbol tables. For kernel-loadable programs, specifies the
maximum amount of code and data the program can contain excluding its symbol tables. The value of this argument can be any unsigned 32-bit value. The default value is default. If no value is specified, the binder checks the amount of code and data against the address space specified in -size.

- **-max_stack_size maximum_stack_size**
  Specifies the total amount of memory that all static tasks’ stacks and fences may occupy. The value of maximum_stack_size cannot be less than 32,768. The default value is default. If you use the default value, the binder calculates the maximum stack size value.

- **-pm_name path_name**
  Specifies the name of the generated program module (.pm file).

  **Note:** The binder names the program module depending on which command line arguments (object_modules, -control, or -pm_name are specified. See the Explanation section for information on using these arguments to name a program module.

- **-processor processor_string**
  Controls conditional-preprocessing symbol definition, and also validates that object modules are compatible with the target processor. The following are values of processor_string.

  - default
  - pentium4

  These values are site-settable by your system administrator, so some of them might not appear when you invoke the command’s display form. To determine the default value, issue the display_error m$default_processor command. By default, processor_string is the processor family of the current module. If you specify default, the bound file will be of the same processor family as the module you are running on. See the Explanation section for more information.

- **-size size**
  Specifies the size of the address space for which the binder is to bind the object modules. You can specify any numeric value for the -size argument, as well as the values small (to specify a 2-megabyte address space), large (to specify an 8-megabyte address space), and default. The value of default depends on the target architecture and the whether you are binding a user or kernel program. For an ftServer module, the user program size is 128 megabytes, and the kernel program size is 4096 megabytes. See the Explanation section for a description of the syntax of the numerical values that you can specify for the -size argument.

- **-stack_fence_size stack_fence_size**
  Specifies the size, in bytes, of the fence to be placed after each static task’s stack. A fence is an unmapped area of memory; its purpose is to prevent runaway stacks from overwriting other data. The default value is default, which corresponds to a 4096 byte stack fence. If you do not want a stack fence, specify a value of 0. Note that even if you specify a value of 0, the system still allocates a stack fence of 32,768 bytes for the last static task.
-stack_size size

Specifies the number of bytes of storage to reserve for the stack. On ftServer modules, the value of size must be divisible by 16. During bind-time error checking, the binder uses this value to check that the program’s stack will fit in the defined address space. During program execution, this value is the amount of space allocated to the stack. This argument interacts with the -number_of_tasks argument as follows:

- If number_of_tasks is one, the stack size is the minimum stack size.
- If number_of_tasks is greater than one, the stack size is the maximum size of the stack for each task.

By default, the binder allocates 65,536 bytes of stack space for each task.

-target_module module_name

Specifies the name of a module on the current system or on another accessible system. This is useful when binding for a different architecture or a different software release, when the kernel entry points may not be the same, and the bind may result in undefined system entry points. By specifying the -target_module argument, the binder checks the spelling of all system subroutine names called by the object modules to ensure that they match all known kernel entry points on the host system.

-version version

Specifies the release string that appears in the program module header. By default, the binder always initializes version to Pre-release.

-align_mod16

Causes the binder to align the code from each object module on a 16-byte boundary, which may produce faster code. By default, the binder aligns the code on an 8-byte boundary for ftServer modules.

-define_main

Controls whether the binder creates a main function. By default, the binder does not create a main function.

The value cc directs the binder to create a main function as if the VOS c, cc, or OpenVOS vcc compiler had created the object module. Specify this value for applications using the POSIX runtime environment, but not the GNU environment.

The value gcc directs the binder to create a main function as if the OpenVOS gcc or g++ compiler had created the object module. Specify this value for applications using both the POSIX runtime and GNU environments.

The -entry and -define_main arguments are mutually exclusive.

-no_dynamic_tasking

An argument that depends on whether the program is a dynamic tasking program. A dynamic task is created anytime during program execution. Space for a dynamic task is allocated from the heap. If a program module uses dynamic tasking, specify -dynamic_tasking. The binder includes relocation information, which is needed by a program module that uses varying numbers of tasks during execution. If a program
module does not use dynamic tasking, specify \texttt{-no\_dynamic\_tasking}. The binder effectively decreases the size of the program module and suppresses certain warnings.

\begin{itemize}
  \item \texttt{-extended\_names extended\_names\_string} \hfill \texttt{CYCLE}
    Causes the binder to create a program module that enables extended-names support. The supported values are \texttt{default}, \texttt{version1}, \texttt{version2}, and \texttt{disabled}. See “Enabling Extended-Names Support for a Program Module” in the \textit{Explanation} for more information.

    By default, the binder determines whether the program module supports extended names, depending on the type of the application.

  \item \texttt{-load\_in\_kernel} \hfill \texttt{CYCLE}
    Tells the binder to produce a program module that can be loaded with the \texttt{load\_kernel\_program} command. The \texttt{load\_kernel\_program} command loads a program module, such as a library of user programs, separately into the operating system kernel. See \textit{OpenVOS System Administration: Administering and Customizing a System} (R281) for more information on the \texttt{load\_kernel\_program} command.

    \textbf{Note:} The \texttt{-load\_in\_kernel} argument is used primarily for Stratus internal development. Most users should \textbf{not} use this argument. If you need to use it, you can specify it on the command line.

  \item \texttt{-map} \hfill \texttt{CYCLE}
    Creates a bind map. The bind map lists the directories searched, minimum stack size, main entry points, external name definitions, and the starting address and length of the following regions.

    \begin{itemize}
      \item code
      \item symbol table
      \item unshared static storage
      \item shared static storage
    \end{itemize}

    By default, the binder does not create a bind map.

  \item \texttt{-number\_of\_tasks number}
    Specifies the number of static tasks the binder is to create in the program module. \textit{Static tasks} are created when a program module begins execution and is not deleted until the program module stops executing. Memory for a static task is allocated from the stack. The number of static tasks is limited only by the total size of the program module. Each static task has its own stack, its own fence (to prevent tasks from overwriting each other), and its own copy of static storage. The last task’s stack always has a fence size of at least 32,768 bytes. By default, the binder creates one static task.

    See the description of tasking in any of the OpenVOS Transaction Processing Facility Reference manuals for more information.

  \item \texttt{-private\_heap} \hfill \texttt{CYCLE}
    Enables you to move the process heap to the process’s private address space. This allows the compiler to place the address spaces of otherwise identical processes into
different cache locations, often resulting in better system performance by reducing the amount of cache line contention. The default value is -no_private_heap.

You cannot use this argument if your program uses the $s$connect_vm_region2 or $s$connect_vm_region3 subroutine to connect shared virtual memory to the program’s heap area. If your program uses $s$connect_vm_region2 or $s$connect_vm_region3 for this purpose and you specify -private_heap, the subroutine will return an error message. This argument operates independently of the -private_stack argument, which means that you can specify one argument or both arguments simultaneously.

This argument is ignored when binding programs compiled for ftServer modules.

- **-private_stack**
  Enables you to move the process stack to the process’s private address space. This allows the compiler to place the address spaces of otherwise identical processes into different cache locations, often resulting in better system performance by reducing the amount of cache line contention. For non-POSIX programs, the default value is -no_private_stack; for POSIX programs, the default value is -private_stack.

  You cannot use this argument if your program uses the $s$connect_vm_region2 or $s$connect_vm_region3 subroutine to connect shared virtual memory to the program’s stack area. If your program uses $s$connect_vm_region2 or $s$connect_vm_region3 for this purpose and you specify -private_stack, the subroutine will return an error message. This argument operates independently of the -private_heap argument, which means that you can specify one argument or both arguments simultaneously.

  **Note:** Do not specify the -no_private_stack argument when you bind an application that uses OpenVOS POSIX.1 support, as some POSIX features will not work properly.

  This argument is ignored when binding programs compiled for ftServer modules.

- **-profile_alignment_faults**
  Instructs the operating system to count the number of alignment faults that occur during program execution. This data is reported by the profile command. The alignment fault count replaces the page fault count in program modules that have been compiled with the -cpu_profile argument (or the -qc option in OpenVOS Standard C).

  This argument is ignored when binding programs compiled for ftServer modules.

  By default, the binder does not instruct the operating system to count alignment faults.

  For information about how alignment fault information is logged, see the description of the profile command.

- **-references_kernel**
  Causes the binder to allow virtually all external references that can be resolved in the kernel to be resolved. All external variables should be initialized if the -references_kernel argument is used.
Note: The -references_kernel argument is used primarily for Stratus internal development. Most users should not use this argument. If you need to use it, you can specify it on the command line.

- relocatable
  Causes the binder to produce a program module that can be loaded at any address. Only kernel-loadable programs should be bound with this argument.

  Note: The -relocatable argument is used primarily for Stratus internal development. Most users should not use this argument. If you need to use it, you can specify it on the command line.

- retain_all
  Places all external entry names in an entry map for the program module. This map contains the entry value for each name.

- shared
  Directs the binder to generate a shared library. You cannot specify this argument if you also specify any of the following values of the options binder control-file directive: kernel, load_in_kernel, or relocatable.

- statistics
  Displays various statistics about the binding. By default, the binder displays no statistics.

- subroutines_are_functions
  Suppresses the message that can occur in OpenVOS C and OpenVOS Standard C programs when a function is being called as a subroutine, or vice versa. This argument is generally not used to bind PL/I or other programs.

- no_table
  Omits symbol table information in the program module. If you specify -no_table, you cannot use the OpenVOS Symbolic Debugger in a symbolic mode. If you do not specify -no_table when binding and compile the source module using either -table or -production_table, you can use the OpenVOS Symbolic Debugger in a symbolic mode.

Other Options

In addition to the arguments that appear when you specify the bind command with the -form or -usage argument, the bind command has other options that you can specify only on the command line. Table 2-3 describes these options.
Table 2-3. Command-Line Options of the bind Command

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>--as-needed</td>
<td>Controls whether the binder checks to see if a shared library satisfies an outstanding external symbol reference and then loads it only if it does. This is called as-needed mode. By default, as-needed mode is turned off (--no-as-needed). Each option affects only those shared libraries that appear after it on the command line.</td>
</tr>
<tr>
<td>--no-as-needed</td>
<td></td>
</tr>
<tr>
<td>-Bdynamic</td>
<td>Controls whether -lname looks for libname.so before or after it looks for libname.a in a given directory. This option affects only subsequent -lname arguments, so the location of -Bdynamic and -Bstatic on the command line is significant. You can specify these options multiple times on the command line. When building a program module, the default value is -Bstatic; when building a shared library, the default value is -Bdynamic. For example, the following command looks first for libfirst.a and libsecond.so: bind -Bstatic -lfirst -Bdynamic -lsecond...</td>
</tr>
<tr>
<td>-Bstatic</td>
<td></td>
</tr>
<tr>
<td>-Bsymbolic</td>
<td>Changes the way symbol resolution is handled within a shared library. Normally, symbol precedence is based on the runtime load order of shared libraries, with the main program module having the highest precedence. If you specify this option when building a shared library, the shared library itself has first precedence when resolving its own symbol references. This option has no effect if you are not building a shared library.</td>
</tr>
<tr>
<td>--export-dynamic</td>
<td>Forces a main program module to export all non-hidden global symbol definitions. When a program module or shared library exports a symbol, it makes its definition of that symbol visible to other shared libraries in the program. A shared library normally exports all non-hidden global symbol definitions. But a main program module, by default, only exports global symbols referenced by the shared libraries it was bound with. This could cause problems in some applications (for example, those that call the OpenVOS POSIX dlopen function) unless you specify this option. This option has no effect on shared libraries because all non-hidden symbols in a shared library are always exported.</td>
</tr>
<tr>
<td>-export-dynamic</td>
<td></td>
</tr>
</tbody>
</table>
The binder searches for these files as follows:

1. First, it searches all directories specified with `bind -search`.
2. Next, it searches all directories specified with the `search` binder control-file directive.
3. Finally, it searches all directories on the object library path list.

This is the same search order that the binder uses when searching object libraries to satisfy undefined references.

Within each object directory, the binder looks first for `lib name.so` if `-Bdynamic` is in effect, or `lib name.a` if `-Bstatic` is in effect. If neither `-Bdynamic` nor `-Bstatic` is specified, the binder looks only for `lib name.a` and ignores `lib name.so`.

See *Using OpenVOS Dynamic Linking and Shared Libraries* (R648) for more information about the search rules for binding an object module or shared library.

### Table 2-3. Command-Line Options of the `bind` Command (Continued)

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>-l</code> <code>name</code></td>
<td>Binds <code>libname.so</code> or <code>libname.a</code> into the specified program module. The binder searches for these files as follows:</td>
</tr>
</tbody>
</table>
|                               | 1. First, it searches all directories specified with `bind -search`.  
|                               | 2. Next, it searches all directories specified with the `search` binder control-file directive.  
|                               | 3. Finally, it searches all directories on the object library path list.                                                                                                                                 |
|                               | This is the same search order that the binder uses when searching object libraries to satisfy undefined references.                                                                                          |
|                               | Within each object directory, the binder looks first for `lib name.so` if `-Bdynamic` is in effect, or `lib name.a` if `-Bstatic` is in effect. If neither `-Bdynamic` nor `-Bstatic` is specified, the binder looks only for `lib name.a` and ignores `lib name.so`. |
| `-rpath path[:path]...`      | Specifies one or more directory path names for the dynamic linker to look in for the shared libraries on which the resulting program module or shared library depends. If you specify this option multiple times on the command line, the path names are concatenated in the order in which they appeared. |
| `-rpath=path[:path]...`      |                                                                                                                                                                                                          |
| `--rpath path[:path]...`     |                                                                                                                                                                                                          |
| `--rpath=path[:path]...`     |                                                                                                                                                                                                          |
| `-soname name`               | Specifies the name that the dynamic linker uses when looking for this shared library at runtime. The `name` value can be a simple file name or a path name. If you do not specify this option, the dynamic linker uses the same file name that the binder was given. |
| `-soname=name`               |                                                                                                                                                                                                          |
| `--soname name`              |                                                                                                                                                                                                          |
| `--soname=name`              |                                                                                                                                                                                                          |
| `-version-script path`       | Specifies the path name of a version script file, a text file containing symbol visibility and version directives. To specify multiple version script files, repeat this option. |
| `-version-script=path`       |                                                                                                                                                                                                          |
| `--version-script path`      |                                                                                                                                                                                                          |
| `--version-script=path`      | If the binder cannot locate `path` relative to the current directory, but `path` is not an absolute path name, the binder looks in the object file search directories for `path`.                                  |
bind

Table 2-3. Command-Line Options of the bind Command (Continued)

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-whole-archive</td>
<td>Toggles whole-archive mode on and off. Normally, when the binder processes an archive (.a) library file, it takes from it only what it needs to satisfy outstanding undefined symbol references. If you specify whole-archive mode (-whole-archive and --whole-archive), the binder takes everything from an archive. You can specify these options repeatedly on the command line. Each option affects archives that appear after it on the command line.</td>
</tr>
<tr>
<td>--whole-archive</td>
<td></td>
</tr>
<tr>
<td>-no-whole-archive</td>
<td></td>
</tr>
<tr>
<td>--no-whole-archive</td>
<td></td>
</tr>
<tr>
<td>-zdefs</td>
<td>Specifies whether the binder reports undefined symbol errors. When the binder builds a program module, the -zdefs option is the default (that is, the binder reports undefined symbol errors). When the binder builds a shared library, the -znodefs option is the default (that is, the binder does not report undefined symbol errors).</td>
</tr>
<tr>
<td>-znodefs</td>
<td></td>
</tr>
<tr>
<td>-ztext</td>
<td>Specifies whether the binder reports non-position-independent code. When the binder builds a program module, the -ztext option is the default (that is, the binder reports non-position-independent code). When the binder builds a shared library, the -znotext option is the default (that is, the binder does not report non-position-independent code).</td>
</tr>
<tr>
<td>-znotext</td>
<td></td>
</tr>
</tbody>
</table>

Explanation

The bind command binds a specified set of object modules into one program module.

Resolution of Directive, Argument, and Default Values

For a given argument or binder control-file directive, the binder uses the following rules to set the value.

- If the directive is specified, the binder uses the specified value.
- If the directive is not specified, the binder uses the value specified in the argument.
- If no value is specified, the binder uses a processor-specific default value.
Syntax of Numerical Binder Values

Many of the binder command-line arguments and control file directives accept numerical values to represent addresses and sizes. By default, the binder assumes that any specified numerical value is a decimal value. You can change the value’s base by specifying one of the suffixes listed in the following table.

<table>
<thead>
<tr>
<th>Suffix</th>
<th>Base</th>
</tr>
</thead>
<tbody>
<tr>
<td>b or B</td>
<td>binary</td>
</tr>
<tr>
<td>d or D</td>
<td>decimal</td>
</tr>
<tr>
<td>o or O</td>
<td>octal</td>
</tr>
<tr>
<td>x or X</td>
<td>hexadecimal</td>
</tr>
</tbody>
</table>

For example, if you specify `-load_point 80026000x`, the binder interprets 80026000 as a hexadecimal value and changes the load point to that address.

The binder also allows you to use one of the suffixes in the following table to specify a multiplier value.

<table>
<thead>
<tr>
<th>Suffix</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>kb</td>
<td>The binder multiplies the specified value by 1024.</td>
</tr>
<tr>
<td>mb</td>
<td>The binder multiplies the specified value by 1,048,576.</td>
</tr>
<tr>
<td>gb</td>
<td>The binder multiplies the specified value by 1,073,741,824.</td>
</tr>
</tbody>
</table>

For example, if you specify `-size 8mb`, the binder sets the size of the process address space to 8,388,608 bytes.

You cannot specify a multiplier suffix and a base suffix for the same numerical value.

Specifying Object Modules

The object module or modules are specified in the command or in a binder control file. Each object module name normally has the suffix `.obj`, which you can omit when you give the `object_modules` argument or when you list the object modules in the binder control file. Object modules can also have a `.o`, `.a`, or any other suffix, but in this case, you must explicitly specify the suffix. If an explicitly named suffix is `.obj`, `.o`, or `.a`, `bind` uses the file name as specified. If you specify a different suffix or no suffix, `bind` attempts to find the file with the `.obj` suffix first, then it attempts to find the file with no suffix if the first attempt fails.

If the main entry point of the program (usually the first object module specified) accepts parameters, the binder issues a warning. Thus the binder alerts you if you inadvertently list a subroutine instead of the main program as the main entry point.
The binder also warns you if variables have been initialized to different values in different object modules. Although the binder will generate the program module, you should edit and recompile the object modules containing the incorrect initial value.

**Binder Case Sensitivity**

The `bind` command is case sensitive; it does not have a `-mapcase` argument. If you compile a source module using `-mapcase`, you may not be able to bind the resulting object module. In particular, if the source module contains an external variable name or entry name, and the name has one or more uppercase letters, the binder will not recognize the original name and its lowercase version as the same name. (References to the original name might appear in a binder control file.)

**Controlling Binder Directory Searches**

The `-search` argument tells the binder where to find object modules that are entry points in `object_modules` or object modules named in the binder control file. The binder first searches any directories specified in this argument in the order in which they are listed. Next, it searches any directories specified in the binder control file. Finally, it searches the directories specified in the process’s object library paths.

The binder searches a directory only once, even if it is specified more than once.

If you include the `-search` argument on a command line without specifying a search directory, the binder searches the current directory. This is a convenient way to search your current directory before any directories specified in the binder control file. See Using OpenVOS Dynamic Linking and Shared Libraries (R648) for more information about the search rules for binding an object module or shared library.

**How the Binder Initializes Uninitialized External Variables**

If your program declares as an external variable a name that is identical to a message name in the current message file, and if the program does not assign an initial value to the variable, the binder initializes it to the message code corresponding to the message name. For example, if you declare `e$end_of_file(1025)` as an external 2-byte integer, and if you are using the standard message file, the binder initializes the variable to 1025. In the same way, if you set a nonstandard message file with the `use_message_file` command, the binder can assign the status code number of a message in that message file to an external variable whose name is the same as the status code name of a message in that file.

External variables whose names begin with `e$`, `m$`, `q$`, or `r$` are shared in a dynamic tasking program.
Naming of Program Modules
The binder generates a program module (.pm file), puts it in your current directory by default, and names it. The binder names the program module depending on which command line arguments (object_modules, -control, and -pm_name) are specified. If more than one of these arguments are specified, the binder names the program module according to the following rules, which are specified in descending order (for example, rule 1 overrides any other rule specified, rule 2 overrides rules 3 and 4 but is overridden by rule 1, and so on.).

1. If you use the -control argument with the name binder control file directive, the binder names the program module with the name specified in name.
2. If you specify the -pm_name name argument, the binder names the program module with the name specified in name.
3. If you use -control argument without the name binder control file directive, the binder names the program module with the name of the binder control file.
4. If you specify neither the -pm_name argument nor the -control argument, the binder names the program module with the first name specified in the object_modules command line argument.

The binder searches nested links before forming a program module name from a file name.

Specifying an Entry Point
You can use the -entry argument (or the entry binder control-file directive) to define a main entry point of an executable program rather than using the first procedure in the first source module specified in the bind command. However, do not use this argument for a C program that defines a function called main. (That is, do not specify bind -entry main.) Instead, bind in the main function first. For more information, see the OpenVOS Standard C User’s Guide (R364).

Preprocessing the Binder Control File
The -define argument defines variables to be used during preprocessing of the binder control file. For example, the following command line defines __IA32__ in the control file program1.bind as a preprocessor variable.

```
bind -control program1.bind -define __IA32__
```

You use preprocessor variables with preprocessor statements to conditionally preprocess the binder control file. Conditional preprocessing enables you to switch on or off various statements in a binder control file. There are six preprocessor statements.

- $define
- $undefine
- $if
- $else
- $elseif
- $endif

Preprocessor statements must begin in the first column of the control file. Therefore, indentation of nested $if statements is not allowed.
A preprocessor statement must be contained on a single line. A line containing a preprocessor statement cannot contain comments or directives. (An exception is the $endif statement, which ignores any text following it on the line, thus allowing you to comment on the statements.)

To comment out a preprocessor statement, the comment delimiters must surround the statement on the same line, or the comment delimiters must open and close on lines surrounding the preprocessor statement. A comment delimiter cannot appear on the same line as the statement if the corresponding comment delimiter appears on a different line.

Examples of valid and invalid comments follow.

```
Valid
/*$define pentium4*/
/*$define pentium4*/

Invalid
/*$endif*/
```

For more information on the preprocessor, see the description of the `preprocess_file` command.

**Using the `-processor` Argument**

The `-processor` argument performs two functions. First, it controls conditional-preprocessing variable definition. Depending on the value specified in the `-processor` argument or the `processor` binder control-file directive, the binder automatically defines one or two preprocessor variables for the processor family. If the value indicates a processor from the IA-32 family, the binder defines `__IA32__` and `__i386` as preprocessor variables.

Second, `-processor` and `processor` validate object modules. If an object module is not compatible with the specified processor value, the binder issues a warning.

Note that the `processor` binder control-file directive overrides the `-processor` command line argument. This means that if the `processor` directive is specified, the preprocessor variable automatically defined by the `-processor` argument is redefined to be the value specified by the `processor` directive. (See the description of the `processor` directive later in this command description for more information.)

**Creating a Bind Map**

If you specify the `-map` argument, the binder generates a bind map and puts it in your current directory. The name of the bind map is the name of the program module with the suffix changed from `.pm` to `.map`. For a description of the contents of the `.map` file, see the VOS PL/I User’s Guide (R145) or the OpenVOS Standard C User’s Guide (R364).

**Displaying Binder Statistics**

If you specify the `-statistics` argument, the binder displays various statistics about the binding of your object modules. The statistics are displayed in a table, as shown in the
following example. For a description of the binder statistics fields, see the VOS PL/I User’s Guide (R145) or the OpenVOS Standard C User’s Guide (R364).

Binder Statistics

<table>
<thead>
<tr>
<th>Phase</th>
<th>Seconds</th>
<th>Paging</th>
<th>Disk I/O</th>
<th>CPU</th>
</tr>
</thead>
<tbody>
<tr>
<td>pass1</td>
<td>8</td>
<td>45</td>
<td>19</td>
<td>7</td>
</tr>
<tr>
<td>pass2</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>map</td>
<td>0</td>
<td>4</td>
<td>11</td>
<td>0</td>
</tr>
<tr>
<td>TOTAL</td>
<td>9</td>
<td>50</td>
<td>30</td>
<td>7</td>
</tr>
</tbody>
</table>

Total bytes: 33294
Bytes per minute: 285377
Bytes removed: 5236
Size of program: 11

Controlling Stack and Heap Growth in ftServer Modules

On ftServer modules, stack frames grow down and heaps grow up. Controlling stack and heap growth is a safety measure. For example, programs with recursive algorithms could, through a programming error, cause runaway use of stack or heap space. The result of this uncontrolled growth could be an application or module interruption.

With ftServer modules, you can use the -size, -max_program_size, -load_point, -stack_size, -stack_fence_size, -max_stack_size, and -max_heap_size arguments (or corresponding binder control-file directives) to control the stack and heap growth of your program module.

**Note:** You can change the program, heap, and stack space values for new processes on an ftServer module with the update_default_cmd_limits command and display them with the list_default_cmd_limits command. For a description of these two commands, see the manual OpenVOS System Administration: Administering and Customizing a System (R281). You can change the program, heap, and stack space values for existing processes on an ftServer module with the update_process_cmd_limits command and display them with the list_process_cmd_limits command.

Figure 2-3 shows the relationships between the -size, -max_program_size, -load_point, -stack_size, -stack_fence_size, -max_stack_size, and -max_heap_size arguments and the elements of the process address space on ftServer modules. Following the figure, each argument is explained.
Figure 2-3. Process Address Space and Related bind Arguments

Note: Figure 2-3 shows the regions in the process address space as contiguous. In most cases, they will not be contiguous.

Setting the Highest Boundary of a Process’s Address Space
Use the -size argument to set the process address space size. When you use the -size argument to set the size of a user program, you are specifying the highest boundary of a process’s address space. The value of size plus the address of the lowest boundary equals the address of the highest boundary. The lowest boundary is the boundary between the user and kernel address spaces. A program module can access addresses down to the lowest boundary. The address of the lowest boundary can be equal to the load point. The load point is the address at which the binder begins allocating code and data for a program module. The load point and lowest boundary address differ, depending on the processor type. For example,
on an ftServer module, the default load point is 0002000x, and the lowest boundary of the
process address space is 00000000x. If you specify a size of 40000000x (1 gigabyte) for a
program to run on an ftServer module, the highest boundary of the process’s address space is
40000000x.

If you do not specify a value for the -size argument or the size directive, the binder
determines the default value to be 128 megabytes for a user program and 4096 megabytes for
a kernel program.

Setting the Code and Data Size of a Process
As indicated in Figure 2-3, you set the code and data size of a process with the
-max_program_size argument. Code and data include the code region, the shared and
unshared static regions, the maximum heap size, and the maximum stack size. For user
programs, code and data include the symbol table. For kernel-loadable programs, code and
data exclude the symbol table. If you do not specify a value for the -max_program_size
argument or the max_program_size directive, the binder checks the amount of code and
data against the address space size (specified with -size).

Specifying the Load Point for an Object Module
The -load_point argument assigns a lowest address for the object module. The value of
-load_point can be any unsigned 32-bit value. For example, if the default load point for
the kernel is 80000000x, specify the following command to change the load point.

    bind prog1 -load_point 80026000x

If you do not specify a value for the -load_point argument or directive, the binder
determines the default value to be 00002000x for either a user program or a kernel program.

Setting the Size of the Stack for a Process with One Static Task
As shown in Figure 2-3, you use the -stack_size argument or the stack_size directive
to specify the maximum size of a stack and fence for a program with one static task. The size
of the fence for a program with one static task is fixed at 65,536 bytes.

Setting the Size of the Stack for a Process with Multiple Static Tasks
Each static task has one stack. As shown in Figure 2-4, you use the -stack_size argument
or the stack_size directive to specify the maximum size of the stack for each static task.
By default, the binder allocates 65,536 bytes of stack space for each static task.

Also as shown in Figure 2-4, you use the -max_stack_size argument or the
max_stack_size directive to specify the maximum space allocated for all static tasks and
their fences.
Setting the Size of the Fence for a Process with Multiple Tasks
As shown in Figure 2-4, the -stack_fence_size argument specifies the size, in bytes, of the fence to be placed after each static task’s stack. If you do not specify a value for the -stack_fence_size argument or the stack_fence_size directive, the binder sets the stack fence size to 4096 bytes except for the last fence, which it sets to 32,768 bytes. If you specify a value larger than 32,768 bytes, the binder sets all stack fences equal to this size. If you do not want a stack fence, specify a value of 0. Note that even if you specify a value of 0, the system still allocates a stack fence of 32,768 bytes for the last static task. For more information on fences, see the VOS PL/I User’s Guide (R145) or the OpenVOS Standard C User’s Guide (R364).

Setting the Heap Size
As shown in Figure 2-3, the -max_heap_size argument specifies the maximum byte size to which the heap can grow. If you do not specify a value for the -max_heap_size argument or the max_heap_size directive, the value of max_heap_size is 0.

Note: A zero value does not imply that the heap’s size is 0. Instead, during binding, the bind command assumes that the maximum heap size is equal to 32,768 bytes for the purpose of checking the size of the address space, and during runtime, OpenVOS assumes that the heap size is unlimited.

Source File Migration and Stack Size
When you recompile source programs to move them to a different architecture, it is likely that the layout of automatic storage and the size of the stack frames will change.
Each architecture has slightly different stack-frame overhead requirements, different data alignment requirements, and, in some cases, different directions of growth. Also, depending on the compiler optimization level that you select, and the capabilities of the particular code generator in use, some automatic variables may not need storage. The opposite is also true: some variables that did not need storage on the previous architecture may require storage on the target architecture. The compilers also create hidden, temporary automatic variables to hold the results of expressions. These temporary variables may also have different allocations on each architecture.

The result of these compiler and machine architecture differences is that valid source code and binder control files that have been fully tested and heavily used for many years may nonetheless require changes when moving from one architecture to another. You should develop and retain a comprehensive, automated test suite of your application code. The ability to exercise all of the software functions at any performance level is crucial to a successful transition and deployment.

**Defining the main Function**

The `-define_main` argument facilitates using legacy OpenVOS applications (written in PL/I, COBOL, and so on) in POSIX and GNU runtime environments.

Legacy OpenVOS applications usually begin execution at the start of the user program (the first subroutine or a location specified by the `-entry` argument). In POSIX and GNU runtime environments, program execution always starts at a function named `main`. In releases prior to OpenVOS Release 17.1.1, you needed to create a small `main` function to call the OpenVOS application program. This needed to be done for each OpenVOS program in an application suite of programs.

The `-define_main` argument directs the binder to create these small `main` functions and to begin execution with them.

Legacy VOS programs set their exit status by calling `$error`. In the POSIX runtime environment, when a program terminates by exiting from `main`, the return value is the exit status. To ensure legacy application compatibility, the binder-generated `main` function always returns -1 as its exit status. This directs the VOS POSIX runtimes not to set the exit status of `main`, but instead to leave the exit status set by `$error` intact.

A program that uses the GNU runtime environment must be bound using the `gcc` or `g++` command. It cannot be bound by invoking the binder directly. To pass the `-define_main` argument through `gcc` or `g++` to the binder, use the `-Wl` option as follows:

```
gcc userprogram.obj -Wl,-define_main,gcc ....
```

For more information about the `gcc` or `g++` command, see *GNU Tools for OpenVOS: User’s Guide* (R453).
Enabling Extended-Names Support for a Program Module

A given release of the operating system supports one of the following combinations of name types:

- Extended names (version 1 and version 2) and legacy names
- Version 1 extended names and legacy names
- Legacy names

Extended names can be longer than legacy names and can contain characters not allowed in legacy names. Version 2 extended names support the greatest number of characters. See Using OpenVOS Extended Names (R631) for detailed information about extended-names support.

When you enable version 1 or version 2 extended-names support for a program module, the program module can access objects with different levels of extended names.

If you want to enable extended-names support in a program module, you must first make sure that extended-names support is enabled for the source module; see Using OpenVOS Extended Names (R631) for more information. In addition, if you are binding a POSIX application, you may need to use the binder’s -extended_names argument or extended_names binder control-file option, depending on the application. The binder uses the following rules to set the value.

- If the binder control-file option is specified, the binder uses the specified value.
- If the binder control-file option is not specified, the binder uses the value specified in the command-line argument.
- If no value is specified, the binder uses the default value.

The values for the -extended_names argument and extended_names binder control-file option are default, disabled, version1, and version2.

- If you specify default, the binder determines whether the resulting program module supports extended names, depending on the type of program.
- If you specify disabled, the binder assumes that the resulting program module supports legacy names only, the supports_xfn flag in the program-module header is set to false, and the xfn_version field in the program-module header is set to 0.
- If you specify version1, the binder assumes that the resulting program module supports version 1 extended names, the supports_xfn flag in the program-module header is set to true, and the xfn_version field in the program-module header is set to 1. However, for POSIX applications, the binder may issue warning messages for every object module that it determines may not be able to handle extended names.
- If you specify version2, the binder assumes that the resulting program module supports version 2 extended names, the supports_xfn flag in the program-module header is set to true, and the xfn_version field in the program-module header is set to 2. However, for POSIX applications, the binder may issue warning messages for every object module that it determines may not be able to handle extended names.
A program is considered a POSIX application if it contains a C or C++ function defined with
the name `main` and `posix_object_library` appears in the object library paths before
`c_object_library`. See *OpenVOS POSIX.1 Reference Guide* (R502) for information
about building POSIX applications.

For POSIX applications, the binder normally enables extended-names support if all object
modules are prepared for it (that is, if `_VOS_EXTENDED_NAMES` was defined during
compilation). Otherwise, the binder disables extended-names support when you specify
default. Alternatively, you can explicitly specify `-extended_names version2` to force
the binder to enable version 2 extended-names support and to detect any inconsistencies.
Similarly, you can explicitly specify `-extended_names version1` if you want a version 1
extended-names application to continue behaving in the same manner.

Table 2-4 summarizes the behavior of the `-extended_names/extended_names` values.
Their behavior depends on whether the object modules are POSIX or non-POSIX, and also
whether the object modules were compiled with `_VOS_EXTENDED_NAMES` defined or not.

<table>
<thead>
<tr>
<th>extended_names Value</th>
<th>Non-POSIX Application</th>
<th>POSIX Application</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>All .obj files</td>
<td>Some .obj files</td>
</tr>
<tr>
<td></td>
<td>compiled with</td>
<td>not compiled with</td>
</tr>
<tr>
<td></td>
<td>_VOS_EXTENDED_NAMES</td>
<td>_VOS_EXTENDED_NAMES</td>
</tr>
<tr>
<td>default</td>
<td>Legacy names</td>
<td>Version 2 extended names</td>
</tr>
<tr>
<td>disabled</td>
<td>Legacy names</td>
<td>Legacy names</td>
</tr>
<tr>
<td>version1</td>
<td>Version 1 extended names</td>
<td>Version 1 extended-names with warning</td>
</tr>
<tr>
<td>version2</td>
<td>Version 2 extended names</td>
<td>Version 2 extended-names with warning</td>
</tr>
</tbody>
</table>

You can use the `display_program_module` command to verify whether the resulting
program module supports extended names and, if so, which type. If it does not support them,
you can rebind with `version1` or `version2` specified to force the binder to enable version 1
or version 2 extended-names support.

**Naming Conventions for Shared Libraries**

Typically, a shared library has the `.so` suffix. However, because the binder does not use the
file-name suffix to decide an input file’s type, a shared or archive file name does not require
a particular prefix or suffix. In addition, a file name ending in `.so` does not necessarily
indicate that it is a shared library.
Creating Shared Libraries

The -shared argument directs the binder to create a shared library. It changes binder behavior as follows:

- By default, the output file is named name.so instead of name.pm. However, if the -pm_name argument or the name: binder-control-file directive is specified, the name given is used as-is; the binder does not append the .so suffix to it.

- It sets the default value of load_point to zero.

- It sets -Bdynamic as the default value, which means that the binder selects a shared version of a library, rather than an archive version, if it encounters both.

A shared library is an incomplete executable. This means that:

- By default, undefined symbols are not reported as errors.

- By default, global symbol resolution is deferred to runtime, even if the symbol is defined in the shared library.

- The executable’s load point may be changed at runtime.

- There is no start address.

Aside from these differences, binding a shared library is similar to binding any other program module: input can be OpenVOS object modules, ELF object modules, and/or other shared libraries. You can use a binder control file. You can examine the output with the display_program_module command and other OpenVOS tools.

See Using OpenVOS Dynamic Linking and Shared Libraries (R648) for more information about binding a shared library.

Access Requirements

You need read access to all of the object modules you are binding. You need modify access to the current directory.

Binder Control File

A binder control file is a text file containing control directives to the binder. If you specify the -control argument, the binder takes its instructions from the file control_file_name.bind. In the file, you can specify the content and the order of the program module’s components.

Syntax, Comments, and Empty Lines

The binder interprets a text line in a binder control file by first parsing the line into words. The rules for forming an identifier or name are similar to the rules for forming an OpenVOS path name. An identifier or name can contain any ASCII printable character, which includes the 52 uppercase and lowercase alphabetic characters, the 10 decimal digits, and the following 24 graphic characters.

" # % * $ + - . / < > @ ^ _ ' { | } ~ , :
A number must begin with 0 through 9, +, or -. Subsequent characters in a number can be 0 through 9 and certain letters. For example, identifiers named 2mb and 0AFFx are processed as numbers.

Adjacent words can be separated by space characters, a colon (:), semicolon (;), comma (,), or parentheses ((())). You need not enclose a path name or other word in apostrophes unless the word contains one of these characters. (Note that, of these special characters, only a colon or comma can be used to form a path name.)

In a binder control file, an empty line is not significant. You can use empty lines to improve the readability of the control file. A comment begins with the characters /* and ends with the characters */.

Directives
This section describes all of the directives. Every directive (except end) has a default value or action, given in the descriptions of the directives, which the binder uses when you omit a directive.

- **define**: definition_specifier...;
  Attaches symbolic names to various constant locations in memory. This enables user programs to use these definitions as external variables and have references to them resolved to the proper region of memory. Specifiers must be separated by commas. The definition_specifier has the following form.

```
symbol_name address (number)
```

- **end**;
  Indicates the end of the binder control file. This directive must be the last one in the file. If you specify more than one end directive, the first one in the file effectively ends the binder control file.

- **entry**: identifier;
  Defines the name of the main entry point of the program. The name identifier must be the name of an entry point in one of the object modules being bound. By default, the first non-null entry point the binder finds is used as the main entry point for the program module. If you specify more than one entry directive, the binder disregards all but the last one.

  C programs that contain a function called main should not use the entry directive. Instead, the module containing the main function should always be bound in first.

- **extended_names**: extended_names_string
  Causes the binder to create a program module that enables extended-names support. The supported values are default, version1, version2, and disabled.

  By default, the binder determines whether the program module supports extended names, depending on the type of the application.
The extended_names directive overrides the -extended_names argument. See “Enabling Extended-Names Support for a Program Module” in the Explanation for more information.

- **high_water_mark: address;**
  Specifies the address of the beginning of heap space for a process on an ftServer module. By default, the address of the beginning of heap space for a process is 40000000x. Increase address to connect very large or very numerous shared virtual memory regions; decrease address to maximize stack and heap space. For most applications, however, you do not need to use this directive. For more information on calculating the value for this directive, see Appendix C.

- **load_point: number;**
  Assigns a lowest address for the object module. You can specify any unsigned 32-bit value for number. For example, if the default load point for the kernel is 80000000x, specifying the following directive changes the load point.

  ```
  load_point: 80026000x;
  ```

  The load_point directive overrides the -load_point command line argument. For more information on setting a load point, see the Explanation section.

- **max_heap_size: size;**
  Specifies the maximum byte size to which the heap can grow.

  The max_heap_size directive overrides the -max_heap_size argument. For more information on specifying the maximum heap size, see the Explanation section.

- **max_program_size: number;**
  Specifies, in bytes, the maximum amount of code and data the program can contain, excluding its symbol tables. You can specify any unsigned 32-bit value for number.

  The max_program_size directive overrides the -max_program_size argument. For more information on specifying the maximum size of a program, see the Explanation section.

- **max_stack_size: size;**
  Specifies the total amount of memory, in bytes, that all static tasks’ stacks and fences can occupy. The value of size must be greater than 32,767.

  The max_stack_size directive overrides the -max_stack_size argument. For more information on allocating memory for static tasks, see the Explanation section.

- **modules: module_specifier...;**
  Declares the object modules to be bound. The values for module_specifier identify the path names of the object modules and how the object modules are to be bound. Specifiers must be separated by commas. The module_specifier values have the following form.

  ```
  (module_term)...[module_attribute]...
  ```
A `module_term` is the path name of an object module followed by zero or more `module_attribute` terms separated by spaces. Module terms are separated by commas. You can factor common attributes from a series of object module terms and put them outside the parentheses.

The path name of `module_term` can be a relative path name or a full path name. Include or omit the suffix `.obj`.

Allowed values of `module_attribute` are `compact`, `no_compact`, `table`, `no_table`, and `page_aligned`.

- The `compact` and `no_compact` attributes have no effect on programs running on ftServer V Series or Continuum systems.
- The `table` and `no_table` attributes also have the same effect as previously described, but only for the modules with which they are associated. When specified as part of a binder control file, these attributes override the corresponding command arguments.
- The `page_aligned` attribute tells the binder to put the first word in the code region for this object module on a page boundary. This is useful in connection with shared memory. As an example, see the description of the `s$connect_vm_region` subroutine in the OpenVOS Subroutines manuals.

The `module_attribute` terms override the corresponding attributes set either by command arguments or by the `options` statement.

You can specify more than one `modules` directive in the `section_name` argument of the `section` directive.

- **name**: `program_name;`
  Specifies the name the binder gives to the bound program module. If you do not specify the `name` directive, the binder names the program module based on a series of rules (see the `Explanation` section above).

If you include more than one `name` directive, the binder uses only the last one before the `end` directive.

- **number_of_tasks**: `number_of_tasks;`
  Specifies the number of static tasks the binder is to create in the program module. The number of tasks is limited only by the total size of the program module. Each task has its own stack and its own copy of static storage. By default, the binder creates one task.

  See the description of tasking in any of the OpenVOS Transaction Processing Facility Reference manuals for more information.

- **options**: `options...;`
  Specifies binder options. See Table 2-5 for the possible values for `options`. 
Table 2-5. Values for the **options** Directive of the **bind** Command

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>compact</td>
<td>These values have no effect on programs running on ftServer V Series or Continuum systems.</td>
</tr>
<tr>
<td>no_compact</td>
<td></td>
</tr>
<tr>
<td>dynamic_tasking</td>
<td>Same as the <code>-dynamic_tasking</code> command line argument. The default value is <code>dynamic_tasking</code>.</td>
</tr>
<tr>
<td>no_dynamic_tasking</td>
<td>Same as the <code>-no_dynamic_tasking</code> command line argument.</td>
</tr>
<tr>
<td>kernel</td>
<td>Tells the binder to create a kernel or, when specified with the <code>load_in_kernel</code>, <code>relocatable</code>, and <code>no_library</code> options, a kernel-loadable program. The default value is <code>no_kernel</code>. For more information, see the description of the <code>load_kernel_program</code> command in the manual <em>OpenVOS System Administration: Administering and Customizing a System</em> (R281).</td>
</tr>
<tr>
<td>no_kernel</td>
<td>Tells the binder not to create a kernel or a kernel-loadable program in the kernel.</td>
</tr>
<tr>
<td>library</td>
<td>Tells the binder to try to resolve any unresolved symbol references found while processing the <code>modules</code> directive. To do this, the binder searches for a module with the same name as the unresolved symbol reference. The default value is <code>library</code>.</td>
</tr>
<tr>
<td>no_library</td>
<td>Tells the binder not to resolve any unresolved symbol references.</td>
</tr>
<tr>
<td>load_in_kernel</td>
<td>Same as the <code>-load_in_kernel</code> command line argument. The default value is <code>no_load_in_kernel</code>.</td>
</tr>
<tr>
<td>no_load_in_kernel</td>
<td>Same as the <code>-no_load_in_kernel</code> command line argument.</td>
</tr>
<tr>
<td>mod16</td>
<td>Same as the <code>-align_mod16</code> command line argument. The default value is <code>no_mod16</code>.</td>
</tr>
<tr>
<td>no_mod16</td>
<td>Same as the <code>-no_align_mod16</code> command line argument.</td>
</tr>
</tbody>
</table>
Table 2-5. Values for the options Directive of the bind Command (Continued)

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>private_heap</td>
<td>Same as the -private_heap command line argument. The default value is no_private_heap.</td>
</tr>
<tr>
<td>no_private_heap</td>
<td>Same as the -no_private_heap command line argument.</td>
</tr>
<tr>
<td></td>
<td>These directives are ignored when binding programs compiled for ftServer modules.</td>
</tr>
<tr>
<td>private_stack</td>
<td>Same as the -private_stack command line argument. The default value is no_private_stack.</td>
</tr>
<tr>
<td>no_private_stack</td>
<td>Same as the -no_private_stack command line argument.</td>
</tr>
<tr>
<td></td>
<td>These directives are ignored when binding programs compiled for ftServer modules.</td>
</tr>
<tr>
<td>references_kernel</td>
<td>Same as the -references_kernel command line argument. The default value is no_references_kernel.</td>
</tr>
<tr>
<td>no_references_kernel</td>
<td>Same as the -no_references_kernel command line argument.</td>
</tr>
<tr>
<td>relocatable</td>
<td>Same as the -relocatable command line argument. The default value is no_relocatable.</td>
</tr>
<tr>
<td>no_relocatable</td>
<td>Same as the -no_relocatable command line argument.</td>
</tr>
<tr>
<td>require_external_static_def</td>
<td>Tells the binder to require that external static variables be initialized. The default value is no_require_external_static_def.</td>
</tr>
<tr>
<td>no_require_external_static_def</td>
<td>Tells the binder not to require that external static variables be initialized.</td>
</tr>
<tr>
<td>subroutines_are_functions</td>
<td>Same as the -subroutines_are_functions command line argument. The default value is no_subroutines_are_functions.</td>
</tr>
<tr>
<td>no_subroutines_are_functions</td>
<td>Same as the -no_subroutines_are_functions command line argument.</td>
</tr>
</tbody>
</table>
The options `compact` and `no_compact` have no effect on programs running on ftServer V Series or Continuum systems. The options `table` and `no_table` have the same effects as the command arguments described earlier in the Arguments section.

The `dynamic_tasking` option causes the binder to include relocation information. Such information is needed by a program that may change the number of dynamic tasks while it runs. Specifying `no_dynamic_tasking` decreases the size of the program by removing relocation information. By default, the binder does include relocation information. The `subroutines_are_functions` option suppresses the message that can occur in C programs when a function is being called as a subroutine, or vice versa.

You can insert several `options` directives in a binder control file. These options override the arguments specified in the `bind` command. The binder reads the entire binder control file before acting on any part; if an option is specified more than once, the last value included in the binder control file will be acted upon.

- **processor: processor_string;**
  Controls conditional-preprocessing variable definition, and validates object modules. The following are the possible values for `processor_string`.

  - `default`
  - `pentium4`

  The `default` value specifies the current module’s default system processor. If you do not use the `processor` directive or the `-processor` command line argument, the binder expects object modules that were compiled for the default system processor.

  No more than one `processor` directive can be specified in a binder control file.

  The `processor` directive overrides the `-processor` command line argument. For more information, see the description of the `-processor` argument in the Explanation section of this command description.

- **region_load_point: section_name region_name: location [ , section_name region_name: location ]...;**

  Allows you to place sections and regions at specific addresses. By default, sections are ordered by their first appearance in the binder control file, and regions appear in the following order: code, unshared data, and shared data within a section.
The location can be an absolute address (such as 80000000x), or it can have the following form, which assigns the section or region to a location following another section or region.

\[ \text{after } \text{section\_name } \text{region\_name} \]

The binder places any regions without explicit load points after all regions with explicit load points. The binder then orders the regions without explicit load points according to the default ordering.

Sections and regions with explicit load points should not overlap. Such an overlap causes the binding to fail.

The following example demonstrates one use of region_load_point.

\[
\begin{align*}
\text{region\_load\_point:} \\
\text{mysect } \text{code} &: 00000000x, \\
\text{mysect } \text{static} &: 40000000x, \\
\text{mysect } \text{ext\_static} &: \text{after mysect \_static};
\end{align*}
\]

The preceding example assigns three regions in the user-named section mysect. The code and static regions are assigned absolute addresses, while the ext\_static region is directed to follow the static region. Since no load point was specified for the symtab region, the binder will place it after the other three regions. The binder will place the maps section after the symtab region, by default.

**Note:** The region_load_point directive is used primarily for Stratus internal development. Most users should not use this directive.

**retain**

\[
\text{retain} \quad \left[ : \text{entry\_name} \{ \text{as new\_name} \} , \text{entry\_name} \{ \text{as new\_name} \} \right] \ldots;
\]

Specifies the external entry names for the binder to place in the program module's entry map. This map contains the entry value for each name. You can specify the names of more than one entry point in a retain directive if you separate the names with commas. If no names are specified, the binder places all entry point names in the map. If no names are specified, omit the colon (:) after retain.

The as new\_name option allows you to specify an alternate name under which the entry point is retained.

**search**

\[
\text{search: } \text{directory\_name}..;
\]

Specifies the directory or directories the binder is to search when looking for object modules. (The binder looks for object modules that are referred to by object modules being bound.) The specified directories are added to the list of directories specified in the -search argument of this command. You must separate multiple directory\_name values with commas.

You can include more than one search directive in a binder control file. Each directive adds more directories to the search list.
The binder allows as many command line and/or binder control file search directories, whether specified or default, as memory permits.

See Using OpenVOS Dynamic Linking and Shared Libraries (R648) for more information about the search rules for binding an object module or shared library.

section: section_name;

Specifies the section of the address space in which the binder is to locate object modules or variables. You can specify more than one section, modules, or variables directive. The modules and variables directives are subject to the most recent section directive. By default, all modules and variables for a user program will be placed in the paged section; all modules and variables for a kernel program will be placed in the wired section.

Possible values for section_name are wired, initialization, paged, maps, and user-defined names.

For kernel programs, you can specify any section name except maps, as long as the name follows the naming conventions described in “Syntax, Comments, and Empty Lines,” earlier in this command description. The first four sections of the address space are always the wired, initialization, paged, and maps sections, in that order. Any additional section names that you specify will follow these sections in the section map. You can specify up to 28 more sections in a binder control file. The load_kernel_program command deletes the initialization section after calling each retained entry point in the initialization section. (See OpenVOS System Administration: Administering and Customizing a System (R281) for more information on load_kernel_program.)

For user programs, you can specify the wired and paged section names. Any other sections are ignored and are not loaded when the program executes.

Data or code placed in a wired or initialization section is not subject to page faults. Data or code placed in the paged section is subject to page faults. All wired and paged memory is released when the program terminates.

size: size;

Specifies the size of the address space for which the binder is to bind the object modules. You can specify any numeric value for the size directive, as well as the values small (to specify a 2-megabyte address space) and large (to specify an 8-megabyte address space). This directive has the same effect as the bind command’s -size argument.

If you include more than one size directive, the binder disregards all but the last one. A size specified in a binder control file overrides a size given as a command line argument.

In the following example, the size directive specifies a large address space of eight megabytes.

size: large;
See the Explanation section for more information on process address space.

- **stack_fence_size**: stack_fence_size;
  Specifies the size, in bytes, of the fence to be placed after each static task’s stack. A fence is an unmapped area of memory; its purpose is to prevent runaway stacks from overwriting other data. The default value is default, which corresponds to a 4096 byte stack fence. If you do not want a stack fence, specify a value of 0.

- **stack_size**: stack_size;
  Specifies the number of bytes of storage to reserve for the stack. On ftServer modules, the value of stack_size must be divisible by 16. This directive interacts with the number_of_tasks directive as follows:
  - If number_of_tasks is one, the stack size is the minimum stack size.
  - If number_of_tasks is greater than one, the stack size is the maximum size of the stack for each task.

By default, the binder allocates 65,536 bytes for each static task.

- **synonym**: synonym_specifier...;
  Specifies an entry name to which one or more names are resolved. Each value for synonym_specifier is of the form:
  
  old_name[\*] for entry_name

  All external references matching old_name are resolved to entry_name. The old_name term can have an asterisk as its last character, representing any sequence of zero or more valid identifier characters. Generally, old_name appears in a variable_arg_count directive and old_name defines a set of declarations with different numbers or types of arguments.

- **variable_arg_count**: identifier...;
  Indicates that the program (entry point) named identifier can be called with an indefinite number of arguments. Designating the program identifier suppresses the warning message the binder normally writes on your terminal when you call a program with the wrong number of arguments.

  You can include more than one variable_arg_count directive in a control file. Each directive adds more program names to the list of entry points that accept an indefinite number of arguments.

- **variables**: variable_specifier...;
  Modifies the attributes of an external variable, and, in some cases, can also be used to define an external variable. An external variable has external scope and static storage duration.

  The variable_specifier argument tells the binder the name of an external variable. It can also specify the number of bytes to allocate for the variable and an initial value to give to the variable. In addition, one or more of the following attributes can be associated with the variable: shared or unshared, page_aligned, and flexible_length.
The name value must be the name of an external static variable in at least one of the object modules being bound. The size value term must be an unsigned integer indicating the number of bytes allocated for the variable. The initial_value specifies an initial value for the variable. (You can shorten the word initial to init.) The following is an example of a variables directive.

    variables: num_records (4) init (100000);

This variables directive declares that the external static variable num_records has been allocated 4 bytes, and that the binder is to initialize the variable to 100000.

The term initial_value can be either a character string or an arithmetic constant. The binder initializes a character string as a nonvarying character string having the length size. It initializes an arithmetic variable as a 16-bit or 32-bit signed integer. The size value for an arithmetic variable must be either 2 or 4. An initial value that you assign a variable in a binder control file overrides an initial value specified in a program.

You can define external static variables in a tasking environment as shared variables. The binder allocates external static variables in the program’s static region. Shared variables are allocated in the shared static region, and unshared variables are allocated in the unshared static region. If you define an external static variable with the same name in different compilation units, compile those units into object modules, and then bind the object modules into one program module, the operating system treats all references to the variable in the object modules as a single variable. The binder allocates storage for an external variable and puts the storage address in all references to the variable in the object modules.

The qualifiers shared and unshared cannot appear in the same variable specifier. Shared and unshared variables are allocated in the shared and unshared static regions of the program module, respectively, regardless of whether the program is a tasking program.

You can share a variable that is defined as external static data among tasks in a tasking environment if you specify in the bind file that the variable is shared. When you tell the binder that an external static variable is shared, the binder allocates storage for the variable only once, instead of allocating it for each task.

For example, the following binder control directive makes variables shared variables.

    variables: variable_name shared...;

When you specify more than one variable_name shared term, you must separate them by commas.

When you tell the binder that an external static variable is unshared, the binder allocates storage for the variable for each task.
It is possible to define a variable as `page_aligned`. The `page_aligned` attribute tells the binder to allocate storage for a variable on a page boundary.

The `flexible_length` attribute suppresses warnings that two modules have declared different lengths for an external variable. In the following example, assume that `prog1` and `prog2` have assigned different lengths to `var_1`.

```plaintext
variables: var_1 flexible_length;
modules: prog1, prog2;
```

In the example, the `flexible_length` attribute allows the binder to bind the modules without issuing warnings, using the maximum length of `var_1`.

The length given for a variable in a binder control file overrides the length given in a program.

If an external variable is declared as `shared` in one but not all modules that share it, the binder issues a warning.

An attribute of `shared` (or `unshared`) assigned to a variable in a binder control file overrides any object module attribute.

You can specify more than one `variables` directive in the `section_name` argument of the `section` directive.

```
visibility: visibility-specifier [, visibility-specifier...];
```

Specifies the symbol visibility for shared libraries.

The `visibility-specifier` argument has the following form:

```
symbol-name[*] { default global protected symbolic hidden internal }
```

The `default` and `global` values make a symbol visible to all shared libraries, and they also defer resolution of that symbol until runtime. These two values are interchangeable.

The `protected` and `symbolic` values are identical to the `default` and `global` values, except that references to the symbol from within the defining shared library are resolved at bind time. These two values are interchangeable.

The `hidden` and `internal` values make a symbol local to the defining shared library. All references to it are resolved at bind time. These two values are interchangeable.

As with other binder control-file directives, `symbol-name` can have a final asterisk (*), making it a star name that affects all symbols matching that star name. If more than one `visibility-specifier` argument applies to a single symbol, the last specifier given
bind

applies and overrides any prior directives. This allows a whole class of symbols to be
given one visibility (for example, $c_*$ hidden), and then certain exceptions can be
applied (for example, $c_set_errno symbolic). Symbol visibility specified in the
binder control file overrides any symbol visibility specified in the source file.

Examples

The following binder control file is named make_reports.bind. It binds the program
make_reports.

```plaintext
name: make_reports;
entry: get_report_files;
size: small;
modules: get_report_files, update_reports_files, process_reports,
        put_reports;
variables: version init ('2.1');
end;
```
**break_process**

**Purpose**
This command causes a process or set of processes to go to break level.

**Display Form**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>process_name:</td>
<td>process_name</td>
</tr>
<tr>
<td>-user:</td>
<td>user_name</td>
</tr>
<tr>
<td>-module:</td>
<td>module_name</td>
</tr>
<tr>
<td>-ask:</td>
<td>yes</td>
</tr>
</tbody>
</table>

**Command Line Form**

```
break_process [process_name]
[ -user user_name ]
[ -module module_name ]
[ -no_ask ]
```

**Arguments**

- **process_name**
The name or star name of the process to send to break level. The command signals the break condition in any process whose name matches `process_name`, except for the process issuing the command.

- **-user user_name**
  Specifies a user name or star name whose processes are to be interrupted. Selecting `-user` allows you to break only the processes named `process_name` that were started by the specified user. By default, the command uses your user name. The command does not break the process from which you issue the command. Your process must be privileged to break another user’s process.

- **-module module_name**
  Specifies the module on which the specified process is running. By default, the command looks for the process on the module that is executing your login process.

- **-no_ask**
  Suppresses the prompt, issued when a specified process is a star name, asking whether to break a process with a matching name. By default, the command prompts you to verify the breaking of each process.
**Explanation**

The `break_process` command sends to break level any process whose name matches `process_name` except for the process from which you issue the command. If `process_name` is a star name and you do not use `-no_ask`, the command prompts you before breaking a process whose name matches `process_name`.

The `break_process` command directed to an interactive process has the same effect as pressing the [CTRL]/[BREAK] keys at the terminal to which that process is attached. You can use the `break_process` command even if the process has disabled the [CTRL]/[BREAK] keys.

The `break_process` command directed to a noninteractive process logs that process out. The operating system generates a keep file that can be examined with the debugger later. A keep file is a copy of the interrupted executable image. The file is created in the current directory of the process to which the `break_process` command is directed, with the same name as the interrupted program module except that the suffix is changed from `.pm` to `.process_id.kp` (`process_id` is the program’s process ID, or PID).

**Access Requirements**

You must be privileged to break another user’s process.

**Examples**

If you have more than one process and you issue the `break_process` command using a star name for `process_name`, the system issues the following prompt.

```
Verify processes to be broken.
Smith.Sales(login)? (yes,no,info)
```

If you type `yes` at the prompt, the process goes to break level; if you type `no`, the process continues uninterrupted. If you specify `info`, the system displays information about the subprocess level, program name, PID (that is, the process identifier of the process), and login time of the process. If the process is interactive, the system returns the terminal name from which the process was started. The system does not return a terminal name if the process is not interactive or if the process is logged in remotely from a module that is not running a current version of the operating system.

The system then issues the prompt again.

```
Logged in at 90-02-19 07:33:26 EDT, sub-process level 0.
Running emacs.pm on %s1#t1.6, PID 011D88DDx.
Smith.Sales(login)? (yes,no,info)
```

If your process is at command level, the system returns the following information.

```
Logged in at 90-02-19 07:33:26 EDT, sub-process level 0.
Running on %s1#t1.6, PID 011D88DDx.
Smith.Sales(login)? (yes,no,info)
```

**Related Information**

See the descriptions of the `start_process` and `logout` commands.
bundle

Purpose
This command packages a group of files for transfer to another location.

Display Form

<table>
<thead>
<tr>
<th>Argument</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>source_star_name_or_dir</td>
<td>Required</td>
</tr>
<tr>
<td>destination_file</td>
<td></td>
</tr>
<tr>
<td>-exclude</td>
<td>yes</td>
</tr>
<tr>
<td>-save_subdirs</td>
<td>yes</td>
</tr>
<tr>
<td>-combine_using</td>
<td>save</td>
</tr>
<tr>
<td>-compress_using</td>
<td>gzip</td>
</tr>
<tr>
<td>-output_format</td>
<td>for_ftp</td>
</tr>
<tr>
<td>-short_suffix</td>
<td>no</td>
</tr>
<tr>
<td>-brief</td>
<td>no</td>
</tr>
</tbody>
</table>

Command Line Form

```
bundle source_star_name_or_dir [destination_file]
    [-exclude star_name...]
    [-no_save_subdirs]
    [-combine_using save_type]
    [-compress_using compression_type]
    [-output_format format]
    [-short_suffix]
    [-brief]
```

Arguments

- `source_star_name_or_dir` **Required**
  Either a star name that represents the files you want to send, or a directory that contains the files you want to send. If you specify a star name, the files do not need to be in the current directory. If you do not want to send all of the files that match your star name, or if you do not want to send all of the files in the directory, use the `--exclude` argument to exclude some of the files.

- `destination_file`
  The path name of a file. The file does not need to be in the current directory. All of the files specified in `source_star_name_or_dir` are bundled together into a single destination file. The `unbundle` command expands the combined file back into individual files. See the Explanation for more information.
**-exclude star_name**

Specifies the list of star names that you do not want included in the final package. They must be object names, but you cannot specify a directory. If you specify more than one `star_name` on the command line, separate the names with spaces and enclose the whole list in quotes. (You do not need quotes when you enter the names into the display form.) For example:

```
bundle mydir>special>* mydir>sendfile -exclude '*.backup *.test'
```

**-no_save_subdirs**

Specifies which files, links, and subdirectories to save.

By default (the value yes), the command saves the entire source directory tree, including any subdirectories that may be present. If you enter a star name for the source, the command includes both files and directories that match the star name. The command also includes links that are in the source directory or that match the source star name.

If you specify the value no, the command saves only the files in the source directory, or only files that match the source star name. The command does not include subdirectories in the source directory, nor does it include directories that match the source star name. The command does not include links either in the source directory or that match the source star name.

**-combine_using save_type**

Specifies whether to create a .save file before compression occurs. The values are save and none.

If you specify save (the default), the command creates a .save file before compression is applied. The command saves all objects, including subdirectories and links, in the source directory. Specify the -no_save_subdirs argument if you want to save only files (not links or subdirectories) in the source directory.

If you specify none, the command does not save anything. You can specify this value only when you send a single file. The file must be a non-extent OpenVOS file with the sequential, stream (but not 64-bit stream), relative, or fixed file format so that it can be transported through non-OpenVOS systems without loss of the OpenVOS file-format information. If you specify an extent file, it becomes a non-extent file after it is bundled and therefore might not be able to grow large enough to hold the contents of the original extent file. In addition, any attributes associated with the original file (for example, open options, implicit locking, and so on) are not present on the bundled file.

**-compress_using compression_type**

Specifies which utility, if any, to use to compress the data. The values are gzip, bzip2, and none.

If you specify gzip (the default), the command uses the gzip utility to compress the data. This utility can achieve up to 90% compression (that is, a 100-block file would be compressed to 10 blocks). Compression ratios of 60-70% are common.
If you specify `bzip2`, the command uses the `bzip2` utility to compress the data. This utility often compresses more efficiently than the `gzip` utility.

If you specify `none`, the command does not perform any compression.

- **-output_format format** [CYCLE]
  Specifies the output format for the destination file. Different methods of transporting the file can cause certain kinds of corruption in the file; these output formats help protect against file corruption. The values are `for_ftp`, `for_rsn`, `uuencode`, `uuencode_stream`, `base64`, and `base64_stream`. See the Explanation for more information.

- **-short_suffix** [CYCLE]
  Specifies whether the destination path has long or short suffixes.

  By default (`no`), the destination path has a series of suffixes that show which processes were used to produce that file (for example, `sendfile.save.evf.gz.uue`).

  If you specify `yes`, all output formats other than `for_ftp` rename the destination path so that only the final suffix is retained (for example, `sendfile.uue`). If you specify the `for_ftp` output format, the command performs an additional encoding step and then renames the suffix to `.ftp` (for example, `sendfile.save.evf.gz` becomes `sendfile.ftp`).

- **-brief** [CYCLE]
  Announces each step in the bundling process on the screen as it occurs. By default (`no`), all output to the screen is suppressed except for error messages and a few messages from the `save` command that cannot be turned off.

**Explanation**

The `bundle` command packages a group of files for transfer to another location. You can perform optional compression with the `bzip2` or `gzip` utility.

If the bundling operation results in a `destination_file` that is larger than 2 GB, the command generates multiple compressed save files as well as a `.toc` file. Be sure to transfer this `.toc` file to the target system along with the save files, or the `unbundle` command will fail.

**Destination-File Suffixes**

The `bundle` command appends a specific suffix to `destination_file`, depending on what type of processing occurs. If you specify `-short_suffix`, the command removes all but the last of these suffixes, to simplify the handling of the file names.

For example, if you specify the value `xxx` for `destination_file`, Table 2-6 shows some (but not all) of the possible destination names after processing occurs.
Table 2-6. Possible Destination File Names After Bundling

<table>
<thead>
<tr>
<th>Value of -output_format</th>
<th>Is -short_suffix Specified?</th>
<th>Description</th>
<th>Destination File Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>for_ftp (default)</td>
<td>No</td>
<td>Binary stream</td>
<td>xxx.save.evf.gz</td>
</tr>
<tr>
<td>for_rsn</td>
<td>No</td>
<td>Binary sequential</td>
<td>xxx.save.evf.gz.rsn</td>
</tr>
<tr>
<td>uuencode</td>
<td>No</td>
<td>ASCII sequential</td>
<td>xxx.save.evf.gz.uue</td>
</tr>
<tr>
<td>uuencode_stream</td>
<td>No</td>
<td>ASCII stream</td>
<td>xxx.save.evf.gz.uu</td>
</tr>
<tr>
<td>base64</td>
<td>No</td>
<td>Alphanumeric</td>
<td>xxx.save.evf.gz.b64</td>
</tr>
<tr>
<td>base64_stream</td>
<td>No</td>
<td>Alphanumeric</td>
<td>xxx.save.evf.gz.b</td>
</tr>
<tr>
<td>for_ftp</td>
<td>Yes</td>
<td>Binary stream</td>
<td>xxx.ftp</td>
</tr>
<tr>
<td>for_rsn</td>
<td>Yes</td>
<td>Binary sequential</td>
<td>xxx.rsn</td>
</tr>
<tr>
<td>uuencode</td>
<td>Yes</td>
<td>ASCII sequential</td>
<td>xxx.uue</td>
</tr>
<tr>
<td>uuencode_stream</td>
<td>Yes</td>
<td>ASCII stream</td>
<td>xxx.uu</td>
</tr>
<tr>
<td>base64</td>
<td>Yes</td>
<td>Alphanumeric</td>
<td>xxx.b64</td>
</tr>
<tr>
<td>base64_stream</td>
<td>Yes</td>
<td>Alphanumeric</td>
<td>xxx.b</td>
</tr>
</tbody>
</table>

Output Formats
The `bundle` command allows you to specify different output formats for the destination file:

- If you specify `for_ftp` (the default), the destination file is the actual file that the `gzip` utility outputs; it has a suffix of `.gz`. This is a binary file; therefore, you must specify the binary file-transfer mode in `ftp` before getting or putting this file. If you also specify the `-short_suffix` argument, the `gzip` output is re-encoded (which adds an additional `.evf` suffix), and all of the suffixes are renamed to a single `.ftp` suffix.

- If you specify `for_rsn`, the output format is a sequential file composed of records, each of which contains 61 bytes of binary data. This allows “problem” modems to handle the file, even though they would not handle files with long records, such as program modules (.pm files). This conversion is necessary because the output from `gzip` is a “raw” stream file, without record separators. The `gzip` utility’s format is not compatible with `remote_request` for transmission over the RSN. The destination file has a suffix of `.rsn` when you specify `for_rsn`.

- If you specify `uuencode`, the output file is encoded in the UNIX `uuencode` format, and the file has a suffix of `.uue`. This is useful when a binary file must be transferred using a method that does not allow transmission of binary data. The output format is a sequential file composed of records, each of which contains 61 printable characters. Some Internet mail servers may still corrupt `uuencoded` files, because the character set, although printable, includes punctuation characters that mail servers rarely filter.
Because of this, the base64 encoding scheme is recommended for use with Internet mail applications. Using the uuencode format increases the size of the output file by 38%, which is more than base64 (33%).

- If you specify uuencode_stream, the output file is exactly like the one created by the uuencode value, except that the file is a stream file instead of a sequential file, and the file has a suffix of .evf. Many applications accept both sequential and stream files. However, the remote_request command (put_file, get_file) does not accept stream files. Some UNIX-oriented applications may require the stream file format, although ftp appears to accept either (the file on the receiving end of ftp is a stream file, but the transmitted file can be either stream or sequential).

- If you specify base64, the output file is encoded in the MIME base64 format, and the file has a suffix of .b64. MIME is a standard developed for Internet mail transmission. Base64 is the MIME standard for transmission of binary data in mail messages. The output format is a sequential file composed of records, each of which contains 72 alphanumeric characters (plus the slash (/), plus-sign (+), and equals-sign (=) characters). These characters were chosen because they are not corrupted by Internet mail servers, even those that perform ASCII-to-EBCDIC conversion. Base64 encoding increases the size of the output file by 33%, which is less than a uuencoded file (38%).

MIME base64 encoding is the most reliable format, because you can send files by nearly any method: remote_request (put_file, get_file), ftp, rsn_transfer, or email. When sending the file by ftp, you do not have to specify the binary file-transfer mode. Likewise, you can specify rsn_transfer without the -binary argument.

To email the file, pull it into the message; you may surround the file with regular message text, if desired. To restore the file, edit it out of the mail message; the file begins with a line containing OpenVOS and ends with a line containing EVF.

- If you specify base64_stream, the output file is exactly like the one created by the base64 value, except that the file is a stream file instead of a sequential file, and the file has a suffix of .evf. Many applications accept both sequential and stream files. However, the remote_request command (put_file, get_file) does not accept stream files. Some UNIX-oriented applications may require the stream file format, although ftp appears to accept either (the file on the receiving end of ftp is a stream file, but the transmitted file can be either stream or sequential).

**Related Information**

See the description of the unbundle command.
**Purpose**

This command compiles an OpenVOS C source module.

**Display Form**

```
<table>
<thead>
<tr>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>source_file_name:</td>
<td></td>
</tr>
<tr>
<td>definition_file_names:</td>
<td></td>
</tr>
<tr>
<td>-mapping_rules:</td>
<td>default</td>
</tr>
<tr>
<td>-processor:</td>
<td>default</td>
</tr>
<tr>
<td>-suppress_diag:</td>
<td>no</td>
</tr>
<tr>
<td>-list:</td>
<td>no</td>
</tr>
<tr>
<td>-table:</td>
<td>no</td>
</tr>
<tr>
<td>-optimize:</td>
<td>yes</td>
</tr>
<tr>
<td>-profile:</td>
<td>no</td>
</tr>
<tr>
<td>-silent:</td>
<td>no</td>
</tr>
<tr>
<td>-full:</td>
<td>no</td>
</tr>
<tr>
<td>-system_programming:</td>
<td>no</td>
</tr>
<tr>
<td>-fixedoverflow:</td>
<td>no</td>
</tr>
<tr>
<td>-include_files:</td>
<td>yes</td>
</tr>
<tr>
<td>-optimization_level:</td>
<td>3</td>
</tr>
<tr>
<td>-registers:</td>
<td>yes</td>
</tr>
<tr>
<td>-check_ansi:</td>
<td>no</td>
</tr>
<tr>
<td>-check_arguments:</td>
<td>no</td>
</tr>
<tr>
<td>-extension_checking:</td>
<td>none</td>
</tr>
</tbody>
</table>
```

- xref: no
- production_table: no
- mapcase: no
- cpu_profile: no
- statistics: no
- nesting: no
- default_char: unsigned
- show_macros: unexpanded
- linted: no
- check_uninitialized: no
- ansi_rules: no
- store_args: no
- type_checking: no
- check Enumeration: no
Command Line Form

```
c source_file_name
   [definition_file_name...]
   [-mapping_rules mapping_string]
   [-processor processor_string]
   [-suppress_diag number]
   [-list]
   [-xref]
   [-table]
   [-production_table]
   [-no_optimize]
   [-mapcase]
   [-profile]
   [-cpu_profile]
   [-silent]
   [-statistics]
   [-full]
   [-nesting]
   [-system_programming]
   [-default_char_string]
   [-fixedoverflow]
   [-show_macros string]
   [-no_include_files]
   [-linted]
   [-optimization_level number]
   [-check_uninitialized]
   [-no_registers]
   [-ansi_rules]
   [-check_ansi]
   [-store_args]
   [-check_arguments]
   [-type_checking level]
   [-extension_checking level]
   [-check Enumeration]
```

Arguments

- **source_file_name**  
  Required
  The path name of an OpenVOS C source module with the suffix .c or .ex.c. You can either supply or omit the .c suffix when you give source_file_name.

- **definition_file_name...**
  One or more path names of definition files, each of which contains a set of #define macros to be established before compilation begins.
-mapping_rules mapping_string

Specifies one of the following data alignment values for a given compilation.

- default
- default/check
- shortmap
- shortmap/check
- longmap
- longmap/check

The default value indicates the system-wide default. The default alignment method is site-settable. To determine the default value, issue the display_error m$default_mapping command. By default, the compiler uses the data alignment rules specified by default. (See the Explanation section of this command description for details.)

-processor processor_string

Specifies the processor on which the program module (.pm) is to run. The display form for the -processor argument restricts the values that you can choose to values for the processor family of the current module.

If the current module uses a processor from the IA-32 family, or if you specify, on the command line, the -processor argument with the pentium4 value, the allowed processor_string values are as follows:

- default
- pentium4

The default value indicates the system-wide default. Unless your system administrator has reset this value, default is pentium4 for modules using IA-32 processors. To determine the default value, issue the display_error m$default_processor command. By default, the compiler produces code intended for the processor specified by default.

-suppress_diag number

Suppresses any diagnostic having a number specified in number. Diagnostics having a severity of 3 or greater are not suppressed. If you want multiple diagnostics to be suppressed, separate each number in the list by a space. The default is to display diagnostics.

-list

Creates a compilation listing. A compilation listing shows all source statements from the source module and include files, as well as a summary of all data definitions and the path names of include files used. You need not select -list when you specify -full, -nesting, or -xref since those arguments create a compilation listing in addition to other listings. By default, the compiler does not generate a compilation listing.
-xref

    Creates a compilation listing and an alphabetized cross-reference listing of all data actually referenced in the program. By default, the compiler does not generate a cross-reference listing.

-table

    Creates a symbol table in the object module, for use by the OpenVOS Symbolic Debugger. The compiler also performs some related operations. (See the Explanation section of this command description for details.) In addition, it suppresses interstatement code optimization, which results in code that is slower than normal. Specifying -table sets the maximum optimization level to 1, unless it has been set to 0 with the -optimization_level argument. By default, the compiler does not create a symbol table, suppress interstatement code optimization, or perform any related operations.

    **Note:** A symbol table greatly increases the size of an object module.

If you specify both -production_table and -table, the compiler produces only a production table and sets the maximum optimization level to 3, unless you explicitly specify some other value.

-production_table

    Creates a symbol table in the object module, for use by the OpenVOS Symbolic Debugger in a production environment. Only variables actually referenced in the source module are placed in the symbol table. The compiler also performs some related operations. (See the Explanation section of this command description for details.) Unlike -table, -production_table does not suppress interstatement code optimization; specifying -production_table sets the optimization level to 3, unless you explicitly specify some other value. As a result, using the set and continue requests of the debug command can lead to unpredictable results. Also, the contents of variables in registers cannot be accurately displayed with the display request of the debug command. In addition, if the optimization level is greater than 2, the contents of any variables may not be accurately displayed with the display request of the debug command. By default, the compiler does not create a symbol table, suppress interstatement code optimization, or perform any related operations.

    **Note:** A symbol table greatly increases the size of an object module.

If you specify both -production_table and -table, the compiler produces only a production table and sets the maximum optimization level to 3, unless you explicitly specify some other value.

-no_optimize

    Generates the object code without optimizing it. Optimization produces more compact object code by removing unnecessary or redundant computations. Specifying -no_optimize sets the optimization level to 0. This overrides any other specification of the optimization level. By default, the compiler optimizes the object code.

-mapcase

    Interprets all uppercase letters except those in character string constants as lowercase letters. If you specify -mapcase, and the source module contains an external variable
name or entry name, you may not be able to bind the resulting object module. (See the Explanation section of this command description.) By default, the compiler distinguishes between uppercase and lowercase letters, and keywords must be in lowercase.

► -profile <CYCLE>
Inserts code in the compiled program that counts the number of times each source statement is executed when the program runs. (See the description of the profile and add_profile commands.) By default, the compiler omits the counting code. You cannot specify both -profile and -cpu_profile in the same command.

► -cpu_profile <CYCLE>
Inserts code in the compiled program that counts the number of times each source statement is executed, the amount of CPU time (in milliseconds) spent executing each statement, and the number of page faults taken executing each statement when the program runs. (See the description of the profile and add_profile commands.) By default, the compiler omits the counting code. The code inserted by this argument uses much more CPU time, but provides more useful information, than the code inserted by -profile. You cannot specify both -cpu_profile and -profile in the same command.

► -silent <CYCLE>
Suppresses the warning messages of severity-1 or severity-0 errors on your terminal during compilation. The compiler, nevertheless, puts the messages in an error file and in any listing it produces. By default, the compiler writes all error messages on your terminal.

► -statistics <CYCLE>
Displays statistics about the compilation as it proceeds. The compiler displays the version number of the compiler as well as the following statistics for each phase:

- disk I/O information
- elapsed real time
- amount of storage used
- number of page faults taken
- elapsed CPU time
- time when the compiler completed the phase

The compiler also displays statistical information for the entire compilation, such as the number of source lines and the symbol table size.

You can specify -statistics to see the progress of the compilation and to determine the phase in which an error occurs. If the compiler produces a listing, it puts the statistics in the listing. By default, the compiler does not display compilation statistics.

► -full <CYCLE>
Creates, from the compiled object code, an assembly language listing, with added comments, in addition to a compilation listing. The compiler uses a disassembler to produce the listing. By default, the compiler does not produce an assembly language listing.
-nesting [CYCLE]
Prints, immediately before each line, the nesting level of structure definitions and compound statements in a listing. The top level is 1, the next level is 2, and so forth. The compiler produces a listing if you specify this argument. By default, the compiler does not put the nesting level on source statements in any listing it produces.

-system_programming [CYCLE]
Checks for the presence of alignment padding within structures that are allocated using the longmap alignment rules. In addition, the compiler diagnoses any function that is declared or defined without a prototype if the function is referenced in the source module. The -system_programming argument also diagnoses all occurrences of unrecognized #pragma directives.

The -system_programming argument also returns a warning message when a slash-asterisk (/* or */ ) style of comment contains a slash-asterisk comment start sequence with no corresponding slash-asterisk comment end sequence. For example, the following comment does not return a warning:

/* This comment is fine. */

However, the following comment does return a warning:

/* This comment generates a warning because
  */ it contains an extra slash-asterisk. */

See the VOS C Language Manual (R040) for more information about -system_programming.

default_char [CYCLE]
Specifies whether char variables that are not explicitly defined as signed or unsigned will default to unsigned or signed. By default, char variables default to unsigned.

-fixedoverflow [CYCLE]
Generates code to check for fixed-point overflow in arithmetic operations when the program is run and to signal the fixedoverflow condition when it occurs. By default, the OpenVOS C compiler ensures that fixed overflow exceptions are never detected in arithmetic operations. In this case, if a fixedoverflow occurs, the high-order bits that caused the overflow are lost, and the remaining bits appear as they normally would in the result.

-show_macros [CYCLE]
Specifies the form in which the source of the macros is to be shown in the list file. Possible values for string are expanded and unexpanded. By default, the compiler does not expand macros in the list file.

-no_include_files [CYCLE]
Suppresses, in the compilation listing, source text from files included into the source module with an #include preprocessor directive and text from files specified in the definition_file_names argument. By default, the compiler incorporates such text in the compilation listing.
-linted
Specifies that the source has been verified by the UNIX® lint command. The lint command can be used to process a number of OpenVOS C source modules and diagnose various errors and misuses of the C language. If this argument is specified, better code is generated for return statements. For example, the lint command reports an error if one source module defines a function to return a particular data type, and another source module calls that function assuming a different data type.

-optimization_level number
Specifies the degree of optimization. The possible values are 0, 1, 2, 3, and 4. See the Explanation section of this command description for details. By default, the compiler uses optimization level 3.

-check_uninitialized
Checks all variables for initialization and issues diagnostics for those that are uninitialized if you also specify an optimization level of 3 or 4. This argument is useful when verifying new code or checking for possible bugs, but it can return misleading diagnostics, as in the case of variables that are initialized within a conditional statement. The categories of uninitialized variables diagnosed by the compiler vary depending on whether you choose both -check_uninitialized and an optimization level of at least 3, or choose only an optimization level of at least 3.

-no_registers
Suppresses assignment of items defined with the register storage class to machine registers. Specifying -no_registers allows you to avoid using registers for register data items without having to alter their source declarations. By default, the compiler attempts to allocate machine-register space to items defined with the register storage class. See the VOS C Language Manual (R040) for information on the register storage class.

-ansi_rules
Interprets certain constructs according to ANSI rules. Specifically, the compiler suppresses macro expansion inside quoted strings that appear on macro definition lines. By default, the compiler does not interpret these constructs according to ANSI rules.

-checkansi
Checks whether the source module contains elements that are interpreted differently if -ansi_rules is specified. By default, the compiler does not check.

-store_args
This argument has no effect on programs compiled for ftServer modules but has been retained for compatibility with existing software build scripts.

-check_arguments
Issues a warning message when the value, rather than the address of the value, of a char_varying string or struct longer than eight bytes is passed to a function with no prototype or variable argument list. By default, argument checking does not occur.
-type_checking level
Checks for occurrences of implicit or unintended data-type conversions and for other programming constructs that can cause error conditions to occur. There are four levels of data-type checking.

- none
- minimum
- normal
- maximum

By default, the compiler uses the minimum level of checking. See the Explanation section of this command description for descriptions of the four levels.

-extension_checking level
Checks for use of OpenVOS C language extensions that can affect program transportability. OpenVOS C allows certain programming practices that are not allowed in ANSI C. There are three levels of extension checking.

- none
- minor
- all

By default, the command uses a level of none.

-check Enumeration
Checks operations on enumeration data. The compiler ensures that an enumeration item is only assigned or compared against another item of the same enumeration type or one of its defined enumerators. Other usages are allowed, but the compiler issues a warning message.

To make the compiler diagnose implicit conversions involving an enumeration item, you must specify at least minimum for -type checking. If you specify -type checking none, you disable the type checking performed as a result of -check Enumeration.

Explanation
The c command compiles an OpenVOS C source module into an object module.

The name of the source module must have one of the suffixes .c or .ex.c; you can either supply or omit the .c suffix when you give source_file_name. A source module with the suffix .ex.c is assumed to be the output of the preprocessor and thus fully expanded. The compiler generates an object module, puts it in your current directory, and names it. The name of the object module is the name of the source file with the suffix changed from either .ex.c or .c to .obj.

In general, values specified in a source module (using the #pragma preprocessor directive) take precedence over values specified on the command line. The arguments -mapping_rules, -type_checking, -extension_checking, -mapcase, -processor, -default_char, -fixedoverflow, -show_macros, -no_registers, -ansi_rules, and -check Enumeration have corresponding options that can be entered as #pragma directives in the source code.
When you are compiling for an ftServer module at all optimization levels, the module on which you are compiling must have at least 30,000 pages of paging partition available to avoid running out of virtual memory. In addition, the module on which you are compiling should have 64MB of physical memory available to achieve optimal compiler performance.

Using the -mapping_rules Argument
The -mapping_rules argument allows you to specify the default alignment rules for a given compilation. The value default indicates the system-wide default. The default is site-settable. The value shortmap specifies that the shortmap alignment rules are to be used for the source module. The value longmap specifies that the longmap alignment rules are to be used for the source module. The values default/check, shortmap/check, and longmap/check are equivalent to default, shortmap, and longmap, respectively, except that they also diagnose alignment padding within structures. For example, if you specify default/check, the compiler displays a diagnostic message stating how many bytes of padding exist within a structure. A #pragma preprocessor control line indicating a data alignment method overrides the alignment method specified in -mapping_rules, but alignment padding within structures is still diagnosed if you specify one of the checking values.

For more information on data alignment rules, see the VOS C Language Manual (R040).

Using the -processor Argument
The -processor argument allows you to specify the processor on which the program is to run. The -processor argument also allows you to perform cross-compilation on a source module if the C cross compiler is available on your system. Cross-compilation occurs when a compiler running on one processor family translates a source module into object code for another processor family. The IA-32 cross compiler generates code to run on ftServer modules. Specify the value pentium4 for the -processor argument to target an ftServer module.

Depending on the value specified in the -processor argument or the corresponding #pragma option, the compiler automatically defines one preprocessor variable for the processor family and one or more preprocessor variables corresponding to the processor type(s), as shown in Table 2-7.

Table 2-7. Predefined Preprocessor Variables

<table>
<thead>
<tr>
<th>Processor Value</th>
<th>Preprocessor Variable</th>
</tr>
</thead>
<tbody>
<tr>
<td>default</td>
<td>Varies, depending on the default system processor</td>
</tr>
<tr>
<td>pentium4</td>
<td><strong>PENTIUM4</strong>, <strong>IA32</strong>, and __i386</td>
</tr>
</tbody>
</table>

If you specify -processor pentium4 on the command line, the preprocessor variables __PENTIUM4__, __IA32__, and __i386 are defined.

If the value specified in the -processor argument indicates the IA-32 processor, the maximum number of bytes available for each function’s initial stack frame is 2,147,483,584 bytes.
The amount of automatic storage you can actually declare is somewhat less than these limits because temporary variables generated by the compiler also count towards the limit. Note that although the OpenVOS C compiler supports extremely large values (such as 2,147,483,646), the operating system does not support them.

**Using the -type_checking Argument**
The -type_checking argument allows you to specify which level of data-type consistency you would like the compiler to check and produce warnings for. You can specify one of the following levels of data-type checking: none, minimum, normal, and maximum.

If you specify a type checking value of none, the compiler does no checking for data-type consistency.

If you specify a type checking value of minimum, the compiler produces warnings for the following occurrences.

- the following implicit data-type conversions involving char_varying data and pointers to different objects:
  - pointer to pointer with an object of a different data type
  - non-char_varying to char_varying strings
  - char_varying to non-char_varying strings

- other violations that affect program execution:
  - omitting a level-one element in a structure or union reference
  - returning a value in a void function
  - use of a pointer to a function where a function is required
  - use of an address expression to qualify a structure member that does not locate a structure containing such a member
  - inconsistent extern declaration and definitions
  - an octal constant containing decimal digits
  - omission of a required semicolon
  - omission of an equals sign (=) before an initializer list

If you specify a type checking value of normal, the compiler performs all minimum type checking as well as warnings for the following occurrences.

- all cases specified under minimum

- other implicit data-type conversions, such as constant conversion where the precision is lost (for example, floating-point-to-integer conversion)
other violations that affect program execution:

- use of expressions that do not produce code (for example, `a == 0;`)
- omission of braces surrounding an initializer list
- use of unknown `#pragma` directives
- implicit declarations of functions

If you specify a type checking value of `maximum`, the compiler performs all minimum type checking as well as warnings for the following occurrences. You might find `maximum` type checking too strict; for example, the compiler will issue a warning about the statement `a = a + 1` if `a` is of type `char` or `short`. However, you might find `maximum` type checking useful if you are compiling a program that you plan to debug in order to locate all potential type conversions.

- all cases specified under `minimum` and `normal`
- other implicit data-type conversions:
  - signed to unsigned
  - long to short or char
  - short to char
  - double to float
  - non-float to float
- other violations that affect program execution, use of an obsolete compound assignment operator (for example, use of a minus sign (`-`), asterisk (`*`), or ampersand (`&`) immediately following an equals sign (`=`), without an intervening space).

**Using the `-extension_checking` Argument**

If you specify the `-extension_checking` argument, the compiler checks for use of OpenVOS C language extensions. The three levels of extension checking are `none`, `minor`, and `all`.

If you specify a type checking value of `none`, the compiler does no checking for OpenVOS C language extensions.

If you specify a type checking value of `minor`, the compiler produces warnings for the following OpenVOS C language extensions.

- declaration of an anonymous data item (for example, a `struct` or `union` with no name)
- a reference to a structure member in which all elements of the member’s name are not explicitly stated
- two-byte character constants
- the address-of operator (`&`) with a non-lvalue in an argument list
- an external array definition for which the number of elements is not specified (thereby implying an `extern` declaration)
• an expression yielding a non-address used in a context where an address is required
• an undeclared identifier implicitly defined as int

If you specify a type checking value of all, the compiler performs all minor extension checking and produces warnings for the following OpenVOS C language extensions.

• use of a built-in function
• the Forms Management System accept or screen statement
• definition of char_varying data items
• reference to generic string functions

Using the -list, -full, -xref, or -nesting Argument
If you specify the -list, -full, -xref, or -nesting argument, the compiler creates a compilation listing file and puts it in your current directory. The name of the compilation listing is the name of the source file with the suffix changed from either .ex.c or .c to .list. Any error messages produced or statistics requested are appended to the list file. The -full argument creates an assembly language listing in addition to a program listing. The -nesting argument adds numbers showing the nesting depth of each source statement in a program listing. The -xref argument creates a list of cross-references in addition to a program listing.

Optimizations for ftServer Modules
The -optimization_level argument allows you to optimize programs at different levels. When you are compiling a source module to run on ftServer modules, the levels of optimization are 0, 1, 2, 3, and 4. Specifying optimization level 3 or 4 causes the compiler to perform level 3 optimizations.

If you specify optimization level 0, the compiler performs the following local optimizations.

• local register allocation
• elimination of unreachable code

If you specify optimization level 1, the compiler performs all level 0 optimizations plus the following local optimizations.

• local pattern replacement
• short-circuit evaluation of Boolean expressions
• recognition of algebraic identities
• constant folding
• local combination of common subexpressions within a statement
• peephole optimizations within a single statement
• result incorporation

If you specify optimization level 2, the compiler performs all level 1 optimizations plus the following global optimizations.

• branch retargeting
• global combination of common subexpressions
• removal of invariant expressions from loops
• subsumption
• peephole optimizations across statement boundaries
• global register allocation

If you specify optimization level 3, the compiler performs all level 2 optimizations plus the following global optimizations.

- constant propagation
- removal of invariant assignments from loops
- strength reduction
- linear test replacement
- elimination of dead assignments
- elimination of useless loops
- detection of uninitialized variables
- elimination of dead code and dead stores
- inline expansion
- instruction scheduling
- no allocation of stack space by automatic variables whose values are kept in registers

As stated above, unreachable code is eliminated at all optimization levels on ftServer modules. Sometimes, however, you might want your program to contain some code that will be executed only during a debugging session, not during normal program execution. To prevent the compiler from eliminating such unreachable code, you might consider changing your program as follows.

```c
volatile static int always_zero=0;

if (always_zero != 0) {
    /* Code that should not be eliminated goes here */
```

If you delete the `volatile` attribute from the preceding declaration, the compiler will eliminate the unreachable code. See the VOS C User's Guide (R141) for more information on `volatile`.

**Specifying the Optimization Level**

The arguments `-no_optimize`, `-table`, and `-optimization_level` determine the optimization level. By default, the level is 3.

Table 2-8 describes how each of these compiler arguments affects the optimization level for a source module.

**Table 2-8. Arguments Affecting Optimization Level**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Optimization Level for a Source Module</th>
</tr>
</thead>
<tbody>
<tr>
<td>-optimization_level</td>
<td>Specifies the maximum level of optimization that the compiler uses. Allowed values are 0, 1, 2, 3, and 4. The default level is 3.</td>
</tr>
<tr>
<td>-no_optimize</td>
<td>Specifies a maximum optimization level of 0.</td>
</tr>
<tr>
<td>-table</td>
<td>Specifies a maximum optimization level of 1.</td>
</tr>
</tbody>
</table>
Note: If you compile a program with either the -profile or -cpu_profile argument, you must specify an optimization level lower than 3. Otherwise, -profile or -cpu_profile might not return accurate information, since high optimization levels can cause code to be moved from one statement to another.

Using the -check_uninitialized Argument
The optimization level for a source module also affects the functionality of the -check_uninitialized argument.

- If you select the -check_uninitialized argument and an optimization level of at least 3, the compiler diagnoses all instances of uninitialized variables within the source module. In this case, the compiler diagnoses variables that are initialized as part of code executed conditionally.

- If you do not select the -check_uninitialized argument but do select an optimization level of at least 3, the compiler diagnoses instances of variables within the source module that it knows are uninitialized. In this case, the compiler does not diagnose variables that are initialized as part of code executed conditionally.

- If you select an optimization level of less than 3, the compiler does not diagnose uninitialized variables within the source module even if you select -check_uninitialized.

Using the -table and -production_table Argument
If you specify the -table argument, the compiler creates a symbol table, and allocates storage and generates addresses for all external references, including any that are not used. Symbol-table capacity is 2,147,483,647 nodes. The compiler suppresses interstatement code optimization. The compiler also assures that the generated code never uses a value in one statement from a register that has been loaded in another. That is, all statements are completely self-contained; identifiers can be “set” to any value before executing a statement, and a continue request to branch to any statement will work as expected. Variables defined with the register storage class are allocated and kept in memory locations. Code produced with the -table argument executes more slowly than code produced with the -production_table argument.

If you select the -production_table argument, the compiler performs all of the same operations as it does with -table, except that the compiler does not suppress interstatement code optimization or always keep register variables in memory, and places only variables actually referenced in the symbol table (most unreferenced variables are from include files). Code produced with -production_table may yield unpredictable results if you invoke the OpenVOS Symbolic Debugger set and continue requests.

Using the -linted Argument
If you specify the -linted argument, the compiler assumes that source modules have been “linted” — processed by the UNIX lint command — and that function declarations and definitions are consistent. If you omit the -linted argument, the compiler generates extra code for the return statement to load both the address and data registers, disregarding the definition of the function containing the statement. You can use the -linted argument to tell the compiler that the source module being compiled has been linted and any such additional code may be eliminated (because all such mismatches have been “linted” out).
Using the -mapcase Argument

When you compile a source module using the -mapcase argument, and the module contains an external variable name or entry name with one or more uppercase letters, you may not be able to bind the resulting object module. If the binder encounters a reference to the original name (for example, in a binder control file), it will not recognize the original name and its lowercase version as the same name.

Interpreting Compiler Diagnostics

If the compiler discovers any errors in your source module, it displays an error message on your terminal. Severity-1 and severity-0 messages are not displayed on your terminal when you specify the -silent argument. The compiler also creates an error file in the current directory and writes the error messages to the file. The name of the error file is the name of the source file with the suffix changed from either .ex.c or .c to .error. The compiler also appends error messages to a compilation listing if it produces one. Any .error file is deleted by the system if a subsequent compilation of the same source module is successful (contains no errors).

The OpenVOS C compiler diagnoses five types of errors.

<table>
<thead>
<tr>
<th>SEVERITY</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Advice</td>
</tr>
<tr>
<td>1</td>
<td>Warning</td>
</tr>
<tr>
<td>2</td>
<td>Correctable error</td>
</tr>
<tr>
<td>3</td>
<td>Uncorrectable error: translation can continue</td>
</tr>
<tr>
<td>4</td>
<td>Uncorrectable error: translation cannot continue</td>
</tr>
</tbody>
</table>

The text of the error message explains the cause of the error.

A severity-0 error, although valid C, indicates that improvement is possible, usually in the area of performance. The source module is syntactically correct, so the compiled object module can be bound and executed, but probably with less than optimum efficiency.

A severity-1 error, although valid C, is probably a programming error. Since the source module is syntactically correct at the point of a severity-1 error, however, the compiler continues to compile the source. The compiled object module can be bound and executed, but the program probably will not perform as expected.

A severity-2 error is invalid C, but the compiler can reinterpret the source in such a way that it can continue to compile the program. The compiler proceeds as if the faulty code were replaced with the most likely syntactically correct code. The compiled object module can be bound and executed, but it probably will not perform as expected.

A severity-3 error is invalid C, and the compiler cannot reinterpret the source in such a way that it can continue to compile the program into a usable object module. Nevertheless, the compiler continues to process the source module to detect additional errors. However, the object module is not created.

A severity-4 error is invalid C, and the compiler cannot reinterpret the source in such a way that it can continue to process the source module from the point of the severity-4 error. The object module is not created.
Note: If the compilation results in more than 100 errors, in any combination (excluding severity-0 errors), compilation terminates.

The compiler always overwrites an existing object module having the same name as the object module it produces.

Access Requirements
You need read access to the source module to compile it. You need modify access to the directory from which you are issuing the compile command, in which the .obj file will be created.

Related Information
See the VOS C Language Manual (R040) for a complete description of the OpenVOS C language. See the VOS C User’s Guide (R141) for information on using the OpenVOS C command and its arguments.
c_preprocess

Purpose
This command produces a fully expanded OpenVOS C source module.

Display Form

```
-------------------------- c_preprocess --------------------------
source_file_name:          
output_file_name:          
-definition_files: 
-processor:    default
-list:                     no
-statistics:              no
-silent:                  no
-ansi_rules:              no
-check_ansi:              no
```

Command Line Form

```
c_preprocess source_file_name
   [output_file_name]
   [-definition_files path_name...]  
   [-processor processor_string]
   [-list]
   [-statistics]
   [-silent]
   [-ansi_rules]
   [-check_ansi]
```

Arguments

- `source_file_name`  **Required**
  The path name of an OpenVOS C source module with the suffix `.c`; you can omit the suffix when you give `source_file_name`.

- `output_file_name`
  The path name of an output file. By default, the command names the output file `source_file_name.ex.c`.

- `-definition_files path_name`
  Specifies the path name of one or more definition files, each of which contains a set of `#define` macros to be established before the start of normal preprocessing.
-processor processor_string

Controls conditional-preprocessing symbol definition. The following are values of processor_string.

- default
- pentium4

The default value indicates the system-wide default. Unless your system administrator has reset this value, default is pentium4 for modules using IA-32 processors. By default, processor_string is the processor family of the current module. See the Explanation section of this command description for more information.

-list

Creates a list file named source_file.ex.list. A listing shows all source statements from the source file and include files as well as the path names of include files used.

-statistics

Displays the following statistics on processing as it proceeds.

- version number of the compiler
- elapsed CPU time
- elapsed real time
- number of page faults
- amount of storage used

You can use this argument to observe the progress of processing and to determine the phase in which an error occurs. If the preprocessor produces a listing, it puts the statistics in the listing as well. By default, the preprocessor does not display statistics.

-silent

Suppresses the warning messages of severity-1 errors on your terminal during preprocessing. The preprocessor, nevertheless, puts the messages in an error file and in any listing it produces. By default, the preprocessor writes all error messages on your terminal.

-ansi_rules

Interprets certain constructs according to ANSI rules. Specifically, the preprocessor suppresses macro expansion inside quoted strings that appear on macro definition lines. By default, the preprocessor does not interpret these constructs according to ANSI rules.

-check_ansi

Checks the source module’s conformance to the ANSI standard.

Explanation

The c_preprocess command expands a C source file named source_file_name.c into a source file named output_file_name.ex.c.
If you select the `list` argument, the command creates a file named
`source_file_name.ex.list`, which contains a line-numbered listing of the expanded
source with statistics (if requested) and any diagnostics appended. Diagnostics are also
written to the file `source_file_name.ex.error`, which is produced only if errors occur.

The `-processor` argument controls conditional-preprocessing variable definition. When
you specify `-processor`, preprocessor variables corresponding to the value specified in
`processor_string` are automatically defined. If you specify `-processor pentium4` on
the command line, the preprocessor variables `__PENTIUM4__`, `__IA32__`, and `__i386` are
defined.

**Related Information**

See also the description of the `c` command and the `preprocess_file` command.
call_thru

Purpose
This command connects your login terminal to a remote host as a login terminal or as a slave terminal.

Display Form

```
--- call_thru ---
extension: 255
-gateway:
-address:
-system:
-slave_id:
-rev_charge: no
-line_speed_input: no
```

Command Line Form

```
call_thru [extension]
  [-gateway network]
  [-address system_address]
  [-system name]
  [-slave_id string]
  [-rev_charge]
  [-line_speed_input]
```

Arguments

- **extension**
  The extension number for which the x25_exchange command is configured to receive call_thru connections at the target system. The default extension is 255.

- **-gateway network**
  Specifies an X.25 gateway configured on the local system. The network is the network to which both the local and target systems are connected. If you specify -gateway, you must also specify -address.

- **-address system_address**
  Specifies the address of the target system within the chosen network. The value of system_address is an integer up to 15 digits long. If you specify -address, you must also specify -gateway.
call_thru

- **system name**
  Specifies the system to which you want to connect. The StrataNET networking facility automatically opens a connection with the X.25 gateway module on the target system. If you wish to log in to a module other than the gateway module, use the -module argument of the login command after calling through to the target system.

- **slave_id string**
  Specifies a slave ID from the list of those assigned to one or more slave virtual terminals on the target system.

- **rev_charge**
  Reverses the charge for the call.

- **line_speed_input**
  Increases the asynchronous input/output buffer to its maximum size.

**Explanation**

The call_thru command connects your login terminal to a target system as either a login terminal or a slave terminal.

Remote logins and slave attachments are available through the StrataNET networking facility without requiring a special command. However, the call_thru command is intended specifically for terminal-to-host connection, avoiding the overhead of more general host-to-host communication used by the network.

You must specify either both or neither of the -gateway and -address arguments. If you specify one of these arguments but not the other, the call_thru command displays the following error message, and returns you to command level.

**Missing gateway name or address.**

If you select the -slave_id argument, the string must omit the prefix of the ID on the target site list of IDs. If you omit the prefix, the operating system assumes a value of id=.

If you select the -line_speed_input argument, you increase the asynchronous I/O buffer to its maximum size of 2K bytes. Once you have logged off the other system and the call_thru program is completed, the buffer size returns to its default size.

The operation of the call_thru command is compatible with the X.29/X.25 protocol used to support virtual terminals. Therefore, a target system must be properly configured to accept virtual terminal connections before you can connect to it using the call_thru command. To configure a target system, the x25_exchange command must be used by a system administrator or network administrator. The choice of extension 255 as a default if you do not give the extension argument eliminates the need to remember an extension number when connecting to another system. You cannot omit the extension argument unless the target system is configured with the x25_exchange command with the argument -call_thru_ext 255.

The call_thru command is typically used for a login. Use of the command to log in a slave terminal usually requires an application program designed to control slave terminals on the target system.
After connecting to the target system using the `call_thru` command, log in normally. However, because the terminal type is determined by the device table entry for the virtual terminal, you may need to change your terminal type. Use the `set_terminal_parameters` command to set the terminal type to the correct value for the terminal you are using.

The `call_thru` command passes all input from your keyboard to the target system, and passes all output from the target system to your terminal. Input editing is provided by the terminal handler at your local system. Since `call_thru` is a single command on your local system, the local terminal handler does not recognize any of the terminal input as command lines. As a result, you can retrieve the last input line typed on the target system, but not the last command line.

The `call_thru` command allows connections to non-Stratus systems across an X.25 network. The `call_thru` command emulates a packet assembler/disassembler (PAD) by implementing the PAD side of X.29. To connect to any system on an X.25 network that supports a host X.29, you must give the `-gateway` and `-address` arguments.

The `call_thru` command does not implement all PAD features. In particular, certain X.3 parameters have no effect on `call_thru` operation. Also, the idle timer (parameter 4) and editing (parameter 15) are handled in a special way. There are limitations that affect the `call_thru` command’s emulation of the PAD side of X.29. See *VOS Communications Software: X.25 and X.29 Programming* (R028) for information about the Virtual Terminal Facility.

**Related Information**

See also the command descriptions of `list_gateways` and `set_terminal_parameters`. For additional information about StrataNET and X.25 networks, see *VOS Communications Software: X.25 and StrataNET Administration* (R091).
cancel_batch_requests

Purpose
This command removes an unstarted batch process or processes from a batch queue and stops a batch process if it is currently executing.

Display Form

```
------------------------ cancel_batch_requests ------------------------
process_names: ...
-user: ...
-queue: normal
-module: ...
-ask: yes
```

Command Line Form

```
cancel_batch_requests process_names ...
   [-user user_name]
   [-queue queue_name]
   [-module module_name]
   [-no_ask]
```

Arguments

- **process_names** Required
  One or more names or star names of batch processes to be canceled. The command cancels all processes with matching names.

- **-user user_name**
  Specifies a user name or star name whose batch process you want to cancel. If you specify a `user_name` value, the command cancels only the batch processes matching the `process_names` value that the specified user has submitted. By default, the command uses your user name and thus cancels only your batch requests. You can give a user name other than your own only if you have write access to the queue file of the batch queue.

- **-queue queue_name**
  Specifies the batch queue holding the batch processes to be canceled. By default, the command cancels the processes in the default batch queue, either on the module specified in -module or on the current module.
**cancel_batch_requests**

- **-module module_name**
  Specifies the module containing `queue_name`. By default, the command uses your current module.

- **-no_ask**
  Suppressed the prompt, when a process name is a star name, that asks before you cancel any batch processes with a matching name. By default, the command asks you before canceling any processes when the command is invoked with a star name.

**Explanation**

The `cancel_batch_requests` command removes any matching process from the batch queue. If a process is executing when you cancel it, the batch processor stops execution of the process.

You can specify the batch processes to be canceled by their process names and, if you have write access to the queue file of the batch queue, by the name of the user who submitted the processes.

When `process_names` is a star name, the command prompts you before canceling a batch process with a matching name. The batch processor does not cancel any request in the set until it has prompted you for all of them. If you cancel the command before answering prompts about all requests in the set, none of the requests in the set are canceled. After you have answered prompts for all the names that match one `process_names` term, however, the batch processor cancels the requests in that set before the command asks you about the next set of names. To suppress the prompts and cancel the batch requests without user intervention, issue the `-no_ask` argument.

**Access Requirements**

You can cancel a batch request by the name of the user who submitted it if you have write access to the queue file of the batch queue.

**Examples**

The following command cancels the batch process named `accts_recv` in the batch queue `io_bound`.

```
cancel_batch_requests accts_recv -queue io_bound
```

**Related Information**

For a detailed discussion of batch processing, see the *OpenVOS Commands User’s Guide* (R089). To see the names and queue sequence numbers of the batch processes in a queue, specify the `list_batch_requests` command. See also the command descriptions of `batch`, `display_batch_status`, `update_batch_requests`, `reserve_device`, `move_device_reservation`, and `cancel_device_reservation`. 
cancel_device_reservation

Purpose
This command frees a device for use by other processes.

Display Form

```
-------------------------- cancel_device_reservation --------------------------
device_path_name:          
-force:                   no
```

Command Line Form

```
cancel_device_reservation device_path_name
[-force]
```

Arguments

- **device_path_name**
  Required
  The device for which the reservation is to be canceled.

- **-force**
  Forces the device reservation to be canceled if the device is reserved for some process other than your current one. To use -force you must be privileged. By default, this command cancels reservations only for your current process.

Explanation

The `cancel_device_reservation` command frees a reserved device so that it can be used by other processes. (Devices are usually reserved for the execution of a batch process.)

Devices can be reserved by the `reserve_device` command. It is not necessary to use the -force argument to cancel the reservation of a device reserved for the current process. If some other process has reserved a device with the `reserve_device` command, you can cancel it using the -force argument if your process is privileged.

If another person attaches a port to a device to reserve it, you cannot release the device.

Related Information

For information about attaching and detaching a port, see the descriptions of the `attach_port` and `detach_port` commands. See also the command descriptions of `batch`, `cancel_batch_requests`, `display_batch_status`, `list_batch_requests`, `reserve_device`, and `move_device_reservation`. 

cancel_print_requests

Purpose
This command cancels one or more previously queued print requests.

Display Form
```plaintext
----------------------------- cancel_print_requests -----------------------------

  file_names: ...
  -user: ...
  -queue: standard
  -module: ...
  -ask: yes
```

Command Line Form
```
cancel_print_requests file_names ...
  [-user user_name]
  [-queue queue_name]
  [-module module_name]
  [-no_ask]
```

Arguments
- **file_names**  
  One or more names or star names of files whose printing is to be canceled. The command cancels print requests for all files in the print queue that have matching names and were submitted by the specified user.

- **-user user_name**  
  Specifies a user name or star name, whose print requests are to be canceled. The command cancels only the print requests matching `file_names` that the specified users have submitted. By default, the command uses your user name and thus cancels only your print requests. You can give a user name other than your own only if you have write access to the queue file of the print queue.

- **-queue queue_name**  
  Specifies the print queue containing the print requests to be canceled. By default, the command cancels the print requests in the default print queue, either on the module specified in `-module` or on the current module.

- **-module module_name**  
  Specifies the module containing the specified queue. By default, the command looks for the queue on your current module.
cancel_print_requests

- no_ask

Suppresses the prompt, when there is more than one print request in the print queue that matches a specified file name, that asks whether to cancel a request with a matching name. By default, when more than one request in the queue matches a specified file name, the command asks you before canceling each request with a matching name.

Explanation

The cancel_print_requests command cancels previously entered print requests. If a print request is in the print queue when you cancel it, the command removes the request from the queue. If the operating system is printing the file when you cancel the print request, printing stops.

Specify the print requests to cancel by giving the names of the files in the print queue. If more than one print request in the print queue matches a specific file_names term, the command asks you before canceling each matching request.

When file_names matches more than one print request in the print queue, unless you specify -no_ask the command prompts you before canceling a request with a matching name. The cancel_print_requests command does not cancel any request in the set until it has prompted you about all of them. If you cancel the command before answering prompts about all requests in the set, no requests are canceled. After you have answered prompts about all requests in the current set, however, the command cancels the requests in that set before asking you questions about any other set.

Access Requirements

You can cancel a print request by the name of the user who submitted it if you have write access to the queue file of the batch queue.

Examples

The following command cancels the printing of the file old_memos in the sales_printer print queue.

`cancel_print_requests old_memos -queue sales_printer`

Related Information

See the description of the print command for general information about printing a file. To see the queue of files submitted for printing, use the list_print_requests command. See also the display_print_status command.
### Purpose

This command compiles an OpenVOS Standard C source module.

### Display Form

```plaintext
source_file_name: 
->option_help:  cycle for available options...
option_selection: 
- suppress_diag: 
- include: 
  - processor:  default
  - mapping_rules:  default
  - type_checking:  default
  - extension_checking:  minor
  - check_uninitialized:  no
    - default_char:  unsigned
  - check Enumeration:  no
    - truncate_to:  default
  - check incompatible:  no
    - mapcase:  no
  - check:  no
    - nesting:  no
  - system_programming:  no
    - fixedoverflow:  no
  - table:  no
    - compress:  no
  - list:  no
    - xref:  none
  - statistics:  no
    - full:  no
  - show include:  local
    - show Macros:  unexpanded
  - store Args:  no
    - check Arguments:  no
```
Command Line Form

```
cc source_file_name
   [-option_help string]
   option_selection...
   -suppress_diag number...
   -include include_file_name...
   -processor processor_string
   -mapping_rules mapping_string
   -type_checking level
   -extension_checking level
   -check_uninitialized
   -default_char string
   -checkEnumeration
   -truncate_to string
   -check_incompatible
   -mapcase
   -check
   -nesting
   -system_programming
   -fixedoverflow
   -table
   -compress
   -list
   -xref string
   -statistics
   -full
   -show_include string
   -show_macros string
   -store_args
   [-check_arguments]
```

Arguments

- **source_file_name**  
  Required  
  The path name of an OpenVOS Standard C source module with the suffix .c. You can either supply or omit the .c suffix when you give source_file_name.

- **-option_help**  
  CYCLE  
  Provides help for the short options used with this command. (See the description of the option_selection argument.) Use the arrow keys to cycle through brief descriptions of the short options.
Specifies one or more of the short options. Short options are similar to those often found on the C compilers of other operating systems. These options are preceded by a hyphen (-) and are composed of one character that defines the option and, in some cases, one or more other characters that further specify the option. Table 2-9 briefly summarizes each of the short options.

See the Explanation section of this command description for more information on short-option syntax and the -O, -g, -Xa, -Xc, and -Xt short options. For a complete explanation of each short option, see the OpenVOS Standard C User’s Guide (R364).

### Table 2-9. *cc* Command: Short Options

<table>
<thead>
<tr>
<th>Short Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-A</td>
<td>Undefines the <strong>STDC</strong> macro as well as all predefined macros, such as <strong>VOS</strong>, that are OpenVOS Standard C extensions.</td>
</tr>
<tr>
<td>-Dname[=def]</td>
<td>Predefines a macro from the command line.</td>
</tr>
<tr>
<td>-E</td>
<td>Preprocesses the file only, without compiling.</td>
</tr>
<tr>
<td>-g</td>
<td>Creates a production symbol table for use in debugging.</td>
</tr>
<tr>
<td>-I dir</td>
<td>Specifies a directory path name in which to search for include (header) files. The directory specified in dir is searched before the directories in the process’s include library paths list. You can specify more than one directory to search by using the -I option more than once. The directories are searched in order, beginning with the one specified in the leftmost -I option.</td>
</tr>
<tr>
<td>-M</td>
<td>Compiles without preprocessing the file. If- M is specified, the compiler ignores all preprocessing-related options.</td>
</tr>
<tr>
<td>-O[n]</td>
<td>Specifies an optimization level. Valid values for the optimization level given in n are 0 through 4. See the Explanation section of this command description for information on the -O option and optimization.</td>
</tr>
<tr>
<td>-opath</td>
<td>Changes the file name of the output file (object module or permanent preprocessor output file, if any, or both) to the file name given in path. Or, causes the compiler to write all files it generates to the directory specified by the path name given in path.</td>
</tr>
<tr>
<td>-P[sf]</td>
<td>If you specify the -E option or -compress argument, produces a permanent file having the suffix given in sf and containing preprocessed output. The source file name must be in the form file_name.c, and preprocessed output will be in the form file_name.sf.c. If you omit sf when specifying -P, the default suffix of .ex is used. If you give a leading or trailing period in sf, it is ignored.</td>
</tr>
<tr>
<td>-qc</td>
<td>Adds code to track CPU time per statement if you specify -qc, or to track statement coverage if you specify -q1. For more information, see the descriptions of the profile and add_profile commands.</td>
</tr>
<tr>
<td>-U name</td>
<td>Undefines a specified predefined compiler macro. The compiler does not issue a diagnostic if the given name is not a predefined macro.</td>
</tr>
</tbody>
</table>
Table 2-9. cc Command: Short Options (Continued)

<table>
<thead>
<tr>
<th>Short Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>-u</code></td>
<td>Converts UNIX-style path names for header file names (specified in <code>#include</code> directives) into OpenVOS-style path names. That is, the compiler converts <code>/</code> to <code>&gt;</code>, and converts a leading <code>.(/</code> to <code>(current_dir)&gt;</code>. In addition, the <code>-u</code> option changes the order in which directories are searched to match typical UNIX behavior: for header files specified with quotation marks, the referencing directory (the directory containing the source module or header file that holds the current <code>#include</code> directive) is the starting place for the search.</td>
</tr>
<tr>
<td><code>-W</code></td>
<td>Causes the compiler to display verbose diagnostic messages on the terminal. Verbose messages are always displayed in the <code>.error</code> file and the <code>.list</code> file. The <code>-W</code> option affects only the format of diagnostics displayed on the terminal’s screen.</td>
</tr>
<tr>
<td><code>-w[n]</code></td>
<td>Suppresses diagnostics of severity level <code>n</code> and less. Valid values for the severity level given in <code>n</code> are 1 and 2. If you omit <code>n</code>, the default level is 1. The compiler never suppresses diagnostics of severity-3 or severity-4.</td>
</tr>
<tr>
<td><code>-xa</code></td>
<td>Specifies the degree of ANSI-C conformance that the compiler will use. By default or if you specify <code>-xa</code>, the compiler uses the default conformance mode. In this mode, the compiler is ANSI-C compliant with two exceptions: it does not recognize trigraphs and it defines several keywords not defined by the ANSI C Standard.</td>
</tr>
<tr>
<td><code>-xc</code></td>
<td>If you specify <code>-xc</code>, the compiler uses the default conformance mode but also allows C++-style comments.</td>
</tr>
<tr>
<td><code>-xt</code></td>
<td>If you specify <code>-xt</code>, the compiler uses a transitional conformance mode, allowing certain usages and programming constructs that were common in some older, “traditional” (pre-ANSI) C compilers.</td>
</tr>
</tbody>
</table>

- **-suppress_diag number**
  Supresses any diagnostic having the severity level specified in `number`. Severity levels range from 0 through 4, but the compiler never suppresses diagnostics having a severity of 3 or greater. If you want multiple diagnostics to be suppressed, separate each number in the list by a space. The default is to display all diagnostics.

- **-include include_file_name**
  Specifies one or more path names for include (header) files. The compiler treats the files specified with the `-include` argument as the first include files. If the compiler does not find the file, it displays an error message.
-processor processor_string

Specifies the processor on which the program module (.pm) is to run. The display form for the -processor argument restricts the values that you can choose to values for the processor family of the current module.

If the current module uses a processor from the IA-32 family, or if you specify, on the command line, the -processor argument with the pentium4 value, the allowed processor_string values are as follows:

- default
- pentium4

The default value indicates the system-wide default. Unless your system administrator has reset this value, default is pentium4 for modules using IA-32 processors. To determine the default value, issue the display_error m$default_processor command. By default, the compiler produces code intended for the processor specified by default.

-mapping_rules mapping_string

Specifies the data alignment values for a given compilation. The allowed values for mapping_string are as follows:

- default
- default/check
- shortmap
- shortmap/check
- longmap
- longmap/check

The default value indicates the system-wide default. The default alignment method is site-settable. To determine the default value, issue the display_error m$default_mapping command. By default, the compiler uses the data alignment rules specified by default. See the Explanation section of this command description for more information on the -mapping_rules argument.

type_checking level

Specifies the level of type checking that the compiler uses to diagnose occurrences of implicit data-type conversions and other programming constructs that can cause error conditions to occur. You can specify one of five levels of data-type checking in level.

- default
- none
- minimum
- normal
- pedantic

Note: If you use the -type_checking argument and also use the -Xc option, you should specify minimum as the level of type checking so that the -Xc option will work correctly.
The default value causes the compiler to use the normal level of checking unless you specify the -Xc option. If you specify -Xc, the compiler uses the minimum level. See the Explanation section of this command description for information on the five levels.

- extension_checking level
  Checks for the use of OpenVOS Standard C language extensions that can affect program transportability. OpenVOS Standard C allows certain programming practices that are not allowed by the ANSI C Standard. You can specify one of three levels of extension checking in level.

  - none
  - minor
  - all

  By default, the compiler uses a level of minor. See the Explanation section of this command description for information on the three levels.

- check_uninitialized
  Checks for uninitialized variables if you also specify the value of -O (optimization level) as 3 or 4. If you specify this argument and a value for -O that is less than 3, the compiler issues an error. The -check_uninitialized argument and the -O option determine how the compiler checks for uninitialized variables. The categories of uninitialized variables diagnosed by the compiler vary depending on whether you choose both -check_uninitialized and an optimization level of at least 3, or choose only an optimization level of at least 3. See the Explanation section of this command description for information on checking for uninitialized variables.

- default_char string
  Specifies whether char data items that are declared without an explicit signed or unsigned keyword are signed or unsigned. The allowed values for string are signed and unsigned. By default, char data items that are declared without an explicit signed or unsigned keyword are unsigned.

- check Enumeration
  Checks operations on enumeration data. The compiler ensures that an enumeration item is only assigned or compared against another item of the same enumeration type or one of its defined enumerators. Other usages are allowed, but the compiler issues a warning message.

  By default, the compiler does not check operations on enumeration data. To make the compiler diagnose implicit conversions involving an enumeration item, you must specify at least the minimum value for -type_checking. If you specify -type_checking none, you disable the type checking performed by the -check enumeration argument.
-truncatemetostring

Causes the compiler to truncate externally visible objects. The allowed values for string follow.

- default
- 28/warn
- 28
- 32/warn
- 32

By default, the compiler uses the default value, which maintains all names internally as specified. See the Explanation section of this command description for information on the -truncatemeto argument.

-checkincompatible

Checks for programming constructs that the OpenVOS Standard C compiler (the cc command) treats differently from and the OpenVOS C compiler (the c command). The command checks for the following incompatibilities.

- appearance of trigraphs (if you also specify -Xc)
- use of signed bit fields
- declarations with unnamed bit fields in a structure having an initializer list
- hexadecimal escape sequences whose value would be interpreted differently by the two compilers
- use of macro definitions in which parameter names appear within character constants or character-string literals in macro-definition lines

By default, the compiler does not check for these programming constructs.

-mapcase

Interprets all uppercase letters except those in character constants and character-string literals as lowercase letters. If you specify -mapcase and the source module contains an external variable name or entry name, you may not be able to bind the resulting object module.

By default, the compiler distinguishes between uppercase and lowercase letters, and keywords must be in lowercase. See the Explanation section of this command description for more information on the -mapcase argument.

-check

Checks for out-of-bounds array subscript errors. If the error is caused by an array subscript that is a constant value, the compiler may detect the error at compile time. Otherwise, the error is detected at run time. By default, the compiler does not check for out-of-bounds array subscript errors.

-nesting

Creates a compilation listing and prints, immediately before each line, the nesting level of structure definitions and compound statements. The top level is 1, the next level is 2,
and so forth. By default, the compiler does not put the nesting level on source statements in any listing it produces.

- **-system_programming**

  Performs checks that are useful both in system programming and application programming. The compiler issues a diagnostic when it detects the following:

  - alignment padding within structures that are allocated using the longmap alignment rules
  - a function that is declared or defined without a prototype if the function is referenced in the source module
  - an unrecognized #pragma directive

  The **-system_programming** argument also returns a warning message when a slash-asterisk (/\* or \*/ ) style of comment contains a slash-asterisk comment start sequence with no corresponding slash-asterisk comment end sequence. For example, the following comment does **not** return a warning:

  ```
  /* This comment is fine. */
  ```

  However, the following comment does return a warning:

  ```
  /* This comment generates a warning because
   */ it contains an extra slash-asterisk. */
  ```

  By default, the compiler does not perform the system programming checks. See the *OpenVOS Standard C Reference Manual* (R363) for information on #pragma directives.

- **-fixedoverflow**

  Generates code to check for fixed-point overflow in arithmetic operations when the program is run, and to signal the fixedoverflow condition when it occurs. By default, the OpenVOS Standard C compiler ensures that fixed overflow exceptions are never detected in arithmetic operations. When you do not select the **-fixedoverflow** argument and fixed-point overflow occurs, the high-order bits that caused the overflow are lost, and the remaining bits appear as they normally would in the result.

- **-table**

  Creates a symbol table in the object module, for use by the OpenVOS Symbolic Debugger. The compiler also performs some related operations. (See the Explanation section of this command description for details.) In addition, it suppresses interstatement code optimization, resulting in code that is slower than normal. Specifying **-table** sets the maximum optimization level to 1 unless it has been set to 0 with the **-O** option. By default, the compiler does not create a symbol table.

  **Note:** A symbol table greatly increases the size of an object module.

- **-compress**

  Enables the preprocessing and compiling of a source module containing more than the compiler limit of 2,147,483,647 lines of input (in the source module itself as well as
lines in any include files). By default, a source module can contain no more than 2,147,483,647 lines of input.

The preprocessor always outputs only nonblank lines. That is, the preprocessor does not output blank lines, lines with comments, lines containing preprocessor directives (other than #pragma directives and unknown directives), and lines not incorporated as a result of conditional inclusion. When you use -compress, the compiler refers to the line numbers of the preprocessor’s output (which contains only nonblank lines) rather than to the line numbers of the original source file. Thus, with the -compress argument, a source module can contain up to 2,147,483,647 nonblank lines. With the -no_compress argument, the line numbers in the original source code are communicated to the compiler. The line numbers shown on a compilation listing are accurate whether the -compress or -no_compress argument is used. If the -compress argument is used, the compilation listing is the only way to determine what line number corresponds to what line of source code. Also, using the -compress argument eliminates all include file boundaries and changes the line numbers. Thus, it is more difficult to debug a program compiled with the -compress argument.

- list
  Creates a compilation listing. A compilation listing shows all source statements from the source module and, by default, from local include files. The listing also contains a summary of all data definitions and the path names of include files used. You need not select -list when you specify -full, -nesting, or -xref, since those arguments create a compilation listing in addition to other listings. By default, the compiler does not generate a compilation listing.

- xref
  Creates a compilation listing and an alphabetized cross-reference listing of all data actually referenced in the program. The following are the allowed values for this argument.

  - none
  - referenced
  - all

  The default value is none. If you specify referenced, the command includes in the listing only functions and objects referenced in the program. If you specify all, the command includes in the listing all function and object identifiers in the program.

- statistics
  Displays statistics about the compilation as it proceeds. The compiler displays the version number of the compiler as well as the following statistics for each phase:

  - disk I/O information
  - elapsed real time
  - amount of storage used
  - number of page faults taken
  - elapsed CPU time
  - time when the compiler completed the phase
The compiler also displays statistical information for the entire compilation, such as the number of source lines and the symbol table size.

You can specify `-statistics` to see the progress of the compilation and to determine the phase in which an error occurs. If the compiler produces a listing, it puts the statistics in the listing. By default, the compiler does not display compilation statistics.

- `-full`  
  Creates, from the compiled object code, an assembly language listing, with added comments, in addition to a compilation listing. The compiler uses a disassembler to produce the listing. By default, the compiler does not produce an assembly language listing.

- `-show_include string`  
  Controls which include (header) files the compiler shows in the compilation listing .list file. The allowed values for `string` are shown in the following table.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>local</td>
<td>Only show files included by an <code>#include</code> directive that uses the delimiters <code>&quot;</code> and <code>&quot;</code>. This value is the default.</td>
</tr>
<tr>
<td>standard</td>
<td>Only show files included by an <code>#include</code> directive that uses the delimiters <code>&lt;</code> and <code>&gt;</code>.</td>
</tr>
<tr>
<td>all</td>
<td>Show all include files.</td>
</tr>
<tr>
<td>none</td>
<td>Do not show any include files.</td>
</tr>
</tbody>
</table>

- `-show_macros string`  
  Specifies the form in which the source code for macros is shown in the compilation listing. The allowed values for `string` are shown in the following table.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>unexpanded</td>
<td>Macros are shown in unexpanded form in the .list file. This value is the default.</td>
</tr>
<tr>
<td>expanded</td>
<td>Macros are shown in expanded form in the .list file.</td>
</tr>
<tr>
<td>both_ways</td>
<td>Macros are shown in both the original and in the unexpanded form in the .list file.</td>
</tr>
</tbody>
</table>

- `-store_args`  
  This argument has no effect on programs compiled for ftServer modules but has been retained for compatibility with existing software build scripts.
**cc**

- **-check_arguments**

  Issues a warning message when the value, rather than the address of the value, of a char_varying string or struct longer than eight bytes is passed to a function with no prototype or variable argument list. By default, argument checking does not occur.

**Explanation**

The cc command compiles an OpenVOS Standard C source module into an object module. For a full explanation of the arguments and options that are available with the cc command, see the OpenVOS Standard C User’s Guide (R364).

**Note:** Many people use cc as an abbreviation for the OpenVOS C command, c. That is, they have a line similar to the following in their abbreviations file:

```
first cc by c
```

If you have such a line in your abbreviations file, you must either remove the line or you must precede the cc command with an exclamation point when you issue it (!cc). Otherwise, you will invoke the c command instead of the cc command (the OpenVOS Standard C compiler).

The name of the source module must have the .c suffix. You can either supply or omit the .c suffix when you give source_file_name. The compiler generates an object module, puts it in your current directory, and names it. By default, the name of the object module is the name of the source file with the suffix changed from .c to .obj.

In general, a value specified in a source module (using the #pragma preprocessor directive) takes precedence over values specified on the command line. The arguments -default_char, -check Enumeration, -extension checking, -fixed overflow, -mapcase, -mapping rules, -processor, -show macros, -system programming, and -type checking have corresponding #pragma options that can be entered in the source code.

When you are compiling for an ftServer module at all optimization levels, the module on which you are compiling must have at least 30,000 pages of paging partition available to avoid running out of virtual memory. In addition, the module on which you are compiling should have 64MB of physical memory available to achieve optimal compiler performance.

**Syntax for the Short Options**

For the short options, the syntax is somewhat different from that for arguments to other OpenVOS commands.

- You can specify short option names individually, separating each option with a space. For example:

  ```
  cc my_file -g -u -w
  ```
You can combine most short options into one or more option strings, each beginning with a hyphen. For example:

```
cc -guw my_file
cc -g my_file -uw
```

A space must appear before each new option string, and no space can appear between options when you concatenate them. Only those short options that take no arguments (\(-A\), \(-E\), \(-g\), \(-M\), \(-u\), and \(-W\)) or take single-character arguments (\(-O\), \(-q\), \(-X\), and \(-w\)) can be combined into option strings.

**The -O Short Option and Optimization**

The compiler optimizes the object code during compilation unless you explicitly specify optimization level 0. The `cc` options and arguments shown in Table 2-10 determine the optimization level that the compiler uses for a source module. If you do not specify an optimization level, the default optimization level is 3 unless you select the `-table` argument.

*Note:* If you compile a program with either the `-ql` or `-qc` option, you must specify an optimization level lower than 3. Otherwise, `-ql` and `-qc` might not return accurate information, because high optimization levels can cause code to be moved from one statement to another.

<table>
<thead>
<tr>
<th>Option or Argument</th>
<th>Optimization Level for the Source Module</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>-O</code></td>
<td>Specifies the level of optimization that the compiler uses. Allowed values for <code>level</code> are 0, 1, 2, 3, and 4. The default value is 3 if you do not use the <code>-O</code> option. If you specify <code>-O</code> without specifying a level, the optimization level is 4.</td>
</tr>
<tr>
<td><code>-table</code>†</td>
<td>Specifies optimization level 1. This argument overrides the <code>-O</code> option if that option is used.</td>
</tr>
</tbody>
</table>

† If you specify both the `-table` argument and the `-g` option, the compiler produces only a production symbol table and sets the optimization level to 3 unless you explicitly specify some other optimization level with the `-O` option. Unlike `-table`, `-g` does not suppress interstatement code optimization. As a result, the `set` and `continue` requests of the `debug` command can lead to unpredictable results. Also, the contents of register variables cannot be accurately displayed with the `display` request of the `debug` command. In addition, if the optimization level is greater than 1, the contents of any variables may not be accurately displayed with the `display` request of the `debug` command.

Although compilation time may be longer, the object code produced when you specify `-O4` should be significantly more efficient and will bind and execute faster than the code produced at a lower optimization level.
While you are developing a program and, thus, repeatedly modifying and recompiling it, you might not want to use optimization level 4 for the following reasons.

- Much more compilation time is required for optimization level 4 than for any of the lower levels. At optimization level 4, compilation time can increase by a factor of two or even more for source modules with functions having many lines of code.

- The code generated is often much harder to follow in machine-mode debugging because the values currently in registers may have been set far away from where they are used. Also, a variable’s value is less likely to be in storage for source-mode debugging.

**Optimizations for ftServer Modules**

The `-optimization_level` argument allows you to optimize programs at different levels. When you are compiling a source module to run on ftServer modules, the levels of optimization are 0, 1, 2, and 3. Specifying optimization level 3 or 4 causes the compiler to perform level 3 optimizations.

If you specify optimization level 0, only the following optimizations are performed:

- local register allocation
- the elimination of unreachable code

If you specify optimization level 1, the compiler performs all level-0 optimizations plus the following other local optimizations.

- local pattern replacement
- short-circuit evaluation of Boolean expressions
- recognition of identity constants
- constant folding
- result incorporation
- peephole optimizations within a single statement
- local combination of common subexpressions within a statement

If you specify optimization level 2, the compiler performs all level-1 optimizations plus the following global optimizations.

- branch retargeting
- global combination of common subexpressions
- removal of invariant expressions from loops
- subsumption
- peephole optimizations across statement boundaries
- global register allocation

If you specify optimization level 3, the compiler performs all level-2 optimizations plus the following global optimizations.

- constant propagation
- removal of invariant assignments from loops
- strength reduction
- linear test replacement
• elimination of dead assignments
• elimination of useless loops
• detection of uninitialized variables
• elimination of dead code and dead stores
• inline expansion
• instruction scheduling
• no allocation of stack space by automatic variables whose values are kept in registers

As stated above, unreachable code is eliminated at all optimization levels on ftServer modules. Sometimes, however, you might want your program to contain some code that will be executed only during a debugging session, not during normal program execution. To prevent the compiler from eliminating such unreachable code, you might consider changing your program as follows.

```c
volatile static int always_zero=0;
if (always_zero != 0) {
  /* Code that should not be eliminated goes here */
}
```

If you delete the `volatile` attribute from the preceding declaration, the compiler will eliminate the unreachable code. See the *OpenVOS Standard C User’s Guide* (R364) for more information on `volatile`.

If you specify optimization level 4, the compiler performs the same optimizations that it performs for level 3.

**The -X Short Options and ANSI-C Conformance**

You can use the -X short options to specify the degree of ANSI-C conformance that the compiler uses. By default or if you specify -xa, the OpenVOS Standard C compiler is ANSI C compliant with two exceptions: it does not recognize trigraphs (three-character sequences, beginning with the characters `??`, that represent single characters), and it defines several keywords not defined by the ANSI C Standard.

When you select -xc (strict ANSI-C conformance), the compiler does the following:

• does not recognize `char_varying`, `ext_shared`, and `accept`, which are extensions to OpenVOS Standard C, as keywords. With -xc, the compiler treats these items as undefined identifiers and issues a warning when one is used.

• requires that all variables be defined

• recognizes trigraphs

• diagnoses all of the constructs diagnosed by -extension_checking all. In addition, the compiler diagnoses the following ANSI C Standard violations, some of which are harmless and can typically be ignored.

  - incrementing or decrementing a type-casted pointer
  - conversions between a pointer to `void` and pointer to a function, and vice versa
  - conversions between a pointer to `long` and a pointer to `int`
– incomplete types, even if not allocated
– two union members having the same name, even if the member is not referenced
– octal and hexadecimal escape sequences producing a value too large to fit in wchar_t (greater than 255 decimal)
– using the comma operator in a constant expression
– any attempt to modify a string constant (if you do not specify -Xc, this is detected at run time)

Finally, with -Xc, unknown or syntactically incorrect #pragma options are not diagnosed and are ignored, even if you specify -system_programming, which normally causes these to be diagnosed.

The -Xt option causes the compiler to allow certain usages and constructs common in some older, “traditional” (pre-ANSI) C compilers, such as the OpenVOS C compiler (the c command). When you select -Xt, the compiler does the following:

- suppresses the definition of the __STDC__ predefined macro
- suppresses diagnostics for extraneous data on #else and #endif preprocessor directive lines
- expands macro parameter names that appear within character constants and character-string literals in macro-definition lines
- interprets external object definitions in the same manner as they were interpreted by the OpenVOS C compiler
- allows relational operators to be used with structures and unions
- recognizes the #options, #page, #list, and #nolist preprocessor directives
- limits values in a case label to between -32,768 through 32,767 or 0 through 65,535
- recognizes certain #pragma options that were associated with the OpenVOS C compiler

The -processor Argument
The -processor argument allows you to specify the processor on which the program is to run. The -processor argument also allows you to perform cross-compilation on a source module if the C cross compiler is available on your system. Cross-compilation occurs when a compiler running on one processor family translates a source module into object code for another processor family. The IA-32 cross compiler generates code to run on ftServer modules. Specify the value pentium4 for the -processor argument to target an ftServer module.

Depending on the value specified in the -processor argument or the corresponding #pragma option, the compiler automatically defines one preprocessor variable for the
processor family and one or more preprocessor variables corresponding to the processor type(s), as shown in Table 2-11.

Table 2-11. Predefined Preprocessor Variables

<table>
<thead>
<tr>
<th>Processor Value</th>
<th>Preprocessor Variable</th>
</tr>
</thead>
<tbody>
<tr>
<td>default</td>
<td>Varies, depending on the default system processor</td>
</tr>
<tr>
<td>pentium4</td>
<td><strong>PENTIUM4</strong>, <strong>IA32</strong>, and __i386</td>
</tr>
</tbody>
</table>

If you specify `-processor pentium4` on the command line, the preprocessor variables __PENTIUM4__, __IA32__, and __i386 are defined.

If the value specified in the `-processor` argument indicates the IA-32 processor, the maximum number of bytes available for each function’s initial stack frame is 2,147,483,584 bytes.

The amount of automatic storage you can actually declare is somewhat less than these limits because temporary variables generated by the compiler also count towards the limit. Note that although the OpenVOS Standard C compiler supports extremely large values (such as 2,147,483,646), the operating system does not support them.

See the OpenVOS Standard C User’s Guide (R364) for more information on the `-processor` argument and these initial stack frame limits.

The `-mapping_rules` Argument

The `-mapping_rules` argument allows you to specify the default alignment rules for a given compilation.

- The value default indicates the system-wide default. The default is site-settable.
- The value shortmap specifies that the shortmap alignment rules are to be used for the source module.
- The value longmap specifies that the longmap alignment rules are to be used for the source module.

In the `-mapping_rules` argument, the values default/check, shortmap/check, and longmap/check are equivalent to default, shortmap, and longmap, respectively, except that they also diagnose alignment padding within structures. For example, if you specify default/check, the compiler displays a diagnostic message stating how many bytes of padding exist within a structure. A `#pragma` preprocessor control line indicating a data alignment method overrides the alignment method specified in `-mapping_rules`, but alignment padding within structures is still diagnosed if you specify one of the checking values in the `-mapping_rules` argument. For more information on data alignment rules, see the OpenVOS Standard C Reference Manual (R363).

The `-type_checking` Argument

The `-type_checking` argument allows you to specify the level of type checking that the compiler uses to diagnose occurrences of implicit data-type conversions and other
programming constructs that can cause error conditions to occur. The four levels of data-type checking are none, minimum, normal, and pedantic. The normal level of type checking is the level most programmers will want for a typical program.

If you specify none as the level of type checking, the compiler does not perform any checks for data-type consistency.

If you specify minimum as the level of type checking, the compiler produces warnings for the following implicit data-type conversions involving pointer and char_varying conversions:

- pointer to pointer when the pointed-to types are incompatible
- non-char_varying to char_varying string
- char_varying string to non-char_varying

If you specify normal as the level of type checking, the compiler includes all minimum diagnostics and produces warnings for the following occurrences:

- use of an expression that does not produce code (for example, a == 0;)
- failure to declare or define a function before it is invoked
- failure to supply a value in a return statement for a function defined with a non-void return type
- implicit data-type conversion where precision is lost (for example, float or double or long double to int conversion)

If you specify pedantic as the level of type checking, the compiler includes all minimum and normal diagnostics and produces warnings for the following implicit data-type conversions where precision or value can be lost:

- signed to unsigned
- int or long to short or char
- short to char
- double or long double to float
- int or long to float

The -extension_checking Argument
The -extension_checking argument allows you to specify whether and how the compiler checks for OpenVOS Standard C language extensions. The three levels of extension checking are none, minor, and all.

If you specify none as the level of extension checking, the compiler does not perform any checks for OpenVOS Standard C extensions.

If you specify minor as the level of extension checking, the compiler produces warnings when you use the following OpenVOS Standard C language extensions:

- an undeclared identifier implicitly defined as int
- a partially qualified reference to a structure or union member (for example, referring to a structure member ex_struct.name as name)
If you specify all as the level of extension checking, the compiler includes all minor diagnostics and produces warnings when you use the following OpenVOS Standard C language extensions:

- the invocation of a function-like macro with missing arguments
- a value less than 1 or greater than 32,767 with a \#line directive
- the declaration of an anonymous data item (for example, a struct with no name)
- the declaration of a char_varying data item
- the declaration of a bit field having a type other than int or unsigned int
- two or more identical type definitions with the same name in the same scope
- an extern object declaration that appears in a block and that contains an initializer
- a static function declaration that appears within a block
- the ext_shared storage-class specifier
- a linkage specification, such as "pl1"
- a built-in function, such as $substr
- the Forms Management System accept or screen statement
- the address-of operator (\&) with a non-lvalue in an argument list

The -extension_checking argument diagnoses extensions, but it does not prevent their use. For example, while it diagnoses the use of keywords, such as char_varying, that are extensions to those allowed by the ANSI C Standard, it does not prevent their use. To prevent the use of extensions, use the -Xc command-line option. When you use the -Xc command-line option and also use the extension_checking option, the -Xc option overrides any value given in the extension_checking option.

The -check_uninitialized Argument
The optimization level for a source module also affects the functionality of the -check_uninitialized argument.

- If you specify the -check_uninitialized argument and an optimization level of at least 3, the compiler diagnoses instances of variables that it knows are uninitialized as well as some instances of variables that may be uninitialized (that is, variables that are initialized as part of code executed conditionally).

- If you do not specify the -check_uninitialized argument but do select an optimization level of at least 3, the compiler diagnoses instances of variables within the function definition that it knows are uninitialized. In this case, the compiler does not issue a diagnostic for a variable that is initialized as part of code executed conditionally.

- If you specify an optimization level of less than 3, either explicitly or implicitly (such as by specifying the -table argument), the compiler issues an error and does not diagnose uninitialized variables even if you select -check_uninitialized.

The -table Argument and -g Short Option
If you specify the -table argument, the compiler creates a symbol table, and allocates storage and generates addresses for all external references, including any that are not used. The compiler suppresses interstatement code optimization. The compiler also assures that the generated code never uses a value in one statement from a register that has been loaded in another. That is, all statements are completely self-contained; identifiers can be “set” to any value before executing a statement, and a continue request to branch to any statement will
work as expected. Variables defined with the register storage class are allocated and kept in memory locations.

If you select the -g short option, the compiler creates a production table for use in debugging. With -g, the compiler performs all of the operations that it does with the -table argument except that it does not suppress interstatement code optimization, register variables are not always kept in memory, and only variables actually referenced are placed in the symbol table (most unreferenced variables are from include files). Code produced with the -table argument executes more slowly than code produced with the -g short option. Code produced with -g may yield unpredictable results if you invoke the OpenVOS Symbolic Debugger set and continue requests. Also, in this case, the contents of register variables cannot be accurately displayed with the display request.

The -truncate_to Argument

The -truncate_to argument controls whether and how the compiler truncates externally visible identifiers. By default, the compiler maintains all names internally as specified. However, when the compiler enters the name of an externally visible object into the object file, it is truncated to 32 characters. Such truncation can cause inconsistent results in that the compiler will treat two long object names as distinct while the binder treats them as the same. For example, if your program contains two external variables with 35-character-long names that only differ in the final character, the compiler treats these as two separate variables. The binder, however, since it only sees the first 32 characters, treats them as the same variable.

If you specify default, the compiler will not truncate the names of the externally visible objects to 32 characters until it enters them into the object file. If you specify any of the values other than default, the compiler truncates external names to the specified number of characters and optionally issues a warning (indicating that compile-time truncation has occurred). When a name is referenced, if no existing identifier is found, an attempt is made to look for the truncated name. If the reference is resolved via the truncated name and a warn option has been given, the compiler issues a message indicating the line number and the name before and after every truncation. Thus, it is possible to have both external and internal names without conflict, even though the first 28 or 32 characters are not unique.

The compiler issues a severity-3 error (regardless of whether warn is part of the option) if two different external names are truncated and result in an identical name (that is, the first \( n \) characters are identical). This situation will generally cause incorrect program behavior and must be corrected.

The -mapcase Argument

When you compile a source module using the -mapcase argument, and the source module contains an external variable name or entry name with one or more uppercase letters, you may not be able to bind the resulting object module. If the binder encounters a reference to the original name (for example, in a binder control file), it will not recognize the original name and its lowercase version as the same name.

Compilation Listings

If you specify the -list, -full, -nesting, or -xref argument, the compiler creates a compilation listing file and puts it in your current directory. The name of the compilation listing is the name of the source file with the suffix changed from .c to .list. Any error messages produced or statistics requested are appended to the list file. The -full argument
creates an assembly language listing in addition to a compilation listing. The \texttt{-nesting} argument adds numbers showing the nesting depth of each source statement in a compilation listing. The \texttt{-xref} argument creates a list of cross-references in addition to a compilation listing.

\textbf{Interpreting Compiler Diagnostics}

If the compiler discovers any errors in your source module, it displays an error message on your terminal. The compiler also creates an error file in the current directory and writes the error messages to the file. The name of the error file is the name of the source file with the suffix changed from \texttt{.c} to \texttt{.error}. The compiler also appends error messages to a compilation listing if it produces one. Any \texttt{.error} file is deleted by the system if a subsequent compilation of the same source module contains no errors.

The OpenVOS Standard C compiler diagnoses five types of errors.

\begin{verbatim}
  SEVERITY 0: Advice  
  SEVERITY 1: Warning  
  SEVERITY 2: Correctable error  
  SEVERITY 3: Uncorrectable error: translation can continue  
  SEVERITY 4: Uncorrectable error: translation cannot continue
\end{verbatim}

The text of the error message usually explains the cause of the error.

A severity-0 error, although valid C, indicates that improvement is possible, usually in the area of performance. Because the source module is syntactically correct, the compiled object module can be bound and executed, but probably with less than optimum efficiency.

A severity-1 error, although possibly valid C, is most likely either a programming error or an implicit data-type conversion if you selected a level of type checking other than \texttt{none}. Since the source module is syntactically and semantically correct at the point of the error or conversion, the compiler continues to compile the source and produces an object module.

A severity-2 error is invalid C, but the compiler can reinterpret the source in such a way that it can continue to compile the program. The compiler proceeds as if the faulty code were replaced with the most likely syntactically and semantically correct code and produces an object module.

A severity-3 error is invalid C, and the compiler cannot reinterpret the source in such a way that it can continue to compile the program into a usable object module. Nevertheless, the compiler continues to process the program to detect additional errors.

A severity-4 error is invalid C, and the compiler cannot continue to process the program from the point of the error.

If there are one or more errors but there is no error of severity 3 or greater, the compiler creates an object module that you can bind, but the program may not perform as expected. If a severity-3 or a severity-4 error occurs, the object module is not created.

The compiler always overwrites an existing object module having the same name as the object module it produces.
Access Requirements

You need read access to the source module to compile it. You need modify access to the directory from which you are issuing the compile command, in which the `.obj` file will be created.

Related Information

See the *OpenVOS Standard C Reference Manual* (R363) for a complete description of the OpenVOS Standard C language. See the *OpenVOS Standard C User’s Guide* (R364) for information on using the OpenVOS Standard C command and its arguments.
**change_current_dir**

**Purpose**
This command changes the current directory to a new directory.

**Display Form**

```
------------------------------------------ change_current_dir ------------------------------------------
new_directory: Home_dir
```

**Command Line Form**

```
change_current_dir [new_directory]
```

**Arguments**

- `new_directory`
  The new current directory. By default, the command returns you to your home directory.

**Explanation**

The `change_current_dir` command changes your current directory to a new directory. You can specify the new directory using either a full path name or a relative path name.

**Access Requirements**

If you have status access to the directory that becomes your new current directory, you can display its contents. You must have modify access to a directory to create files in it or copy files to it. If you have null access to the new directory and modify or status access to its parent directory, you can change directories to the new directory, but you cannot perform any operations on its contents.

**Examples**

**Example 1.**
To change the current directory to your home directory, use this command.

```
change_current_dir
```
Example 2.
To change the current directory to the directory with path name `%s1#d02>Sales>Jones`, use this command.

```
change_current_dir %s1#d02>Sales>Jones
```

A new current directory can be on another disk, processing module, or system.

**Related Information**

For more information about directory management, see the command descriptions of `compare_dirs`, `copy_dir`, `create_dir`, `delete_dir`, `display_current_dir`, and `move_dir`. See also *Introduction to VOS* (R001) for a description of relative and full path names.
change_password

Purpose
This command changes your password.

Display Form — Prelogin

change_password

user_name: [REDACTED]
-password: [REDACTED]
-module:

Display Form — Post-Login

change_password

-module:

Command Line Form — Prelogin

change_password user_name
  [-password current_password]
  [-module module_name]

Command Line Form — Post-Login

change_password
  [-module module_name]

Arguments

- user_name Required
  A person name or a full user name as registered in the system’s registration file. This argument is case-insensitive. You can use an alias instead of your person name. If the value of the -require_full_person argument of the login_admin command is no, you can use a word of your person name if the word is unique on your system. For example, if you are registered as Tom_Clark and no one else named Clark is registered on your system, you can use Clark for your user name. However, if the value of the -require_full_person argument of the login_admin command is yes, you must use your person name, full user name, or an alias. In this case, you must use Tom_Clark as your person name.

You may specify an optional group name, in which case it will be validated. You do not need to specify a group name to change your password.
-password current_password
  Specifies your current password. If you omit this argument, you are prompted to enter it, and the password is not displayed on your terminal. If you supply your password on the command line, it is displayed.

  Note: Abbreviations are not expanded.

-module module_name
  Changes your password on a specific module. Valid forms of module_name are %system#module, and #module. The #module format specifies a module on the current system. Primarily, you use this argument to change your password on a remote system. If you do not specify this argument, your password is changed using the current module.

Explanation

The change_password command changes your password, either at prelogin or post-login time. This command is particularly useful if your site only allows SSH connections instead of TELNET connections.

The following sections explain the differences between specifying change_password before and after you are logged in.

Specifying change_password Before You Are Logged In
If you specify this command at prelogin time, all of the arguments are available.

This command first validates your current password. If it is correct, the command prompts you for your new password and then prompts you a second time for your new password. You must specify the same new password at each prompt.

Your system administrator can establish certain criteria that your passwords must meet, such as minimum length, presence of punctuation characters, non-reuse of previous passwords, non-use of null passwords, and so on.

If the new password passes the administrative criteria, it replaces the old password on all modules in the system. If the new password is rejected, the change_password command prompts you again for a new password.

  Note: If the new password contains certain punctuation characters that the operating system recognizes as delimiters, neither the login nor the change_password command will allow you to specify the password on the command line. You must either specify it in the command form or wait for the prompt.

Password changes are replicated to all modules in a system, so you need to change your password only once per system.

When the command finishes, it returns you to the prelogin prompt, and you can enter another prelogin command.

Specifying change_password After You Are Logged In
If you specify this command at post-login time, only the -module argument is available. During this time, you can “force” a password change instead of actually changing your
change_password

password. When you force a password change, the command invalidates your password and prompts you to change your password the next time you log in.

This command is effective at post-login time only if the current module and the module on which the system’s registration database files reside (if they are different modules) have their password expiration times set to a nonzero value. A system administrator sets the password expiration time with the login_admin -password_exp_time command.

Access Requirements

Your user name must have access to the terminal that you are using to change your password. If you attempt to change your password on a remote system, and the remote system requires a password to access it, you will be prompted to provide a password to verify your access to the remote system.

Examples

The following example illustrates how to use the change_password command.

change_password Tom_Clark

    Current password:
    New Password (first attempt):
    New Password (second attempt):
    ready

Related Information

For information about logging in, see the description of the login command. For additional information about changing passwords, see the OpenVOS Commands User’s Guide (R089). For information about setting login parameters for a module, see the description of the login_admin command in the manual OpenVOS System Administration: Registration and Security (R283). For information about how an OpenVOS system administrator can change the password of a specific user, see the description of the set_registration_info command in the manual OpenVOS System Administration: Registration and Security (R283).
check_posix

Purpose
The check_posix command checks that the current module’s configuration meets constraints imposed by the OpenVOS POSIX.1 implementation.

Display Form

```
--------------------------------- check_posix --------------------------------
-check_legacy_inodes: yes
```

Command-Line Form

```
check_posix
   [-no_check_legacy_inodes]
```

Arguments

- `-no_check_legacy_inodes`  
  Does not check whether local logical disks are configured such that 32-bit inode numbers can overflow. By default, the command performs this check.

Explanation
The check_posix command checks that the current module’s configuration meets constraints imposed by the OpenVOS POSIX.1 implementation. The following table lists the constraints and the type of error that occurs if the constraint is not met.

<table>
<thead>
<tr>
<th>Constraint</th>
<th>Type of Error If Constraint Not Met</th>
</tr>
</thead>
<tbody>
<tr>
<td>The POSIX kernel is present.</td>
<td>Fatal error</td>
</tr>
<tr>
<td><code>root.root</code> and <code>nobody.nobody</code> are registered in the user database; each has a valid POSIX UID, a group ID, and a home directory.</td>
<td>Error</td>
</tr>
<tr>
<td>POSIX error codes are available.</td>
<td>Warning</td>
</tr>
<tr>
<td>Local logical disks are not configured so that 32-bit inode numbers can overflow.</td>
<td>Warning</td>
</tr>
<tr>
<td>The system number is less than 128.</td>
<td>Advice</td>
</tr>
</tbody>
</table>
check_posix

A fatal error indicates that POSIX programs either will not execute, or if they execute, will not execute properly.

An error indicates that POSIX programs will execute but may not execute properly.

A warning indicates that some POSIX features may not work properly.

An advice message indicates that all POSIX features work properly. Some OpenVOS extensions may not work.

Since a module’s configuration can change over time due to normal system administration activity, such as adding additional disk space, you should run this command at regular intervals.

This command does not check remote modules; you must run it independently on each module in a system.
 clone_dir

Purpose
This command creates a directory that is a clone of another directory.

Display Form

<table>
<thead>
<tr>
<th>clone_dir</th>
</tr>
</thead>
<tbody>
<tr>
<td>source_dir:</td>
</tr>
<tr>
<td>target_dir:</td>
</tr>
<tr>
<td>-delete: no</td>
</tr>
<tr>
<td>-brief: no</td>
</tr>
</tbody>
</table>

Command Line Form

```
clone_dir source_dir
target_dir
[-delete]
[-brief]
```

Arguments

- **source_dir**
  The name or star name of the directory to be cloned.

- **target_dir**
  The path name of a directory to which the command is to clone the directory. By default, the command creates a directory of the same name as `source_dir` in the current directory. If you specify `target_dir` and if `source_dir` is a star name and `target_dir` is not, then `target_dir` is the name of the directory that will contain the cloned directories. Otherwise, `target_dir` is the name of the cloned directory.

- **-delete** [CYCLE]
  By default, the command asks you to confirm whether to delete a directory if its name conflicts with the name specified in `target_dir`. If you specify `-delete`, the command deletes a specified directory without first asking for confirmation.

- **-brief** [CYCLE]
  By default, if you specify a star name for `source_dir`, the command displays a list of the directories that it cloned. If you specify `-brief`, the command does not display this list.
**Explanation**

The `clone_dir` command has the same effect as copying a directory except that the `clone_dir` command creates an empty directory. Access and default access lists, and all attributes of the directory such as default open options, expandability, and both size and default size limits are identical. In other words, the cloned directory has all of the original directory’s attributes but none of its data.

If both `target_dir` and `source_dir` are star names, the command issues a message for each directory cloned if the source and target names are not identical. If the containing path name is the current directory, the command displays a partial directory name, rather than a full path name.

See the `copy_file` command for a description of how star names function.

**Examples**

Assume that the current directory contains two directories, x1 and x2.

```shell
clone_dir x* y*
  Cloning y* from x*.
  y1 cloned from directory x1.
  y2 cloned from directory x2.
```

In the preceding example, the command displays a message for each directory, since the source and target names are not the same. The command displays partial path names because the directories are in the current directory.

```shell
delete_current_dir x1
clone_dir <x*
  Cloning x* from %s#raid8>Smith>x*
```

In the preceding example, the command does not display any individual messages because the source and target names are identical. The full path name of the source directory is identified because it is not located in the current directory.

```shell
clone_dir x* y
  clone_dir: Some directory in the path name does not exist. Error in 'target_dir'. %s#raid8>Smith>x1>y
```

In the preceding example, the command fails because `y` is the name of the directory into which `x1` and `x2` would be cloned, and it does not exist.

```shell
clone_dir x* <y1
  Cloning %s#raid8>Smith>y1>x* from x*.
```

In the preceding example, the command creates `x1` and `x2` in `%s#raid8>Smith>y1`.

```shell
clone_dir x2 <y2
  Do you really wish to delete %s#raid8>Smith>y2?  (yes, no)
```
In the preceding example, the source directory ($x2$) does not have a star name. Because the target directory ($<y2$) exists, the command replaces it with the cloned directory after verifying that this is the desired behavior.

**Related Information**

`clone_file, copy_dir, copy_file`
**clone_file**

**Purpose**
This command creates a file that is a clone of another file.

**Display Form**

```
---------------------------------- clone_file --------------------------------

old_file: [blank]
new_file: [blank]
-delete: no
```

**Command Line Form**

```
clone_file old_file
   new_file
   [-delete]
```

**Arguments**

- **old_file**
  Required
  The name or star name of the file or files to be cloned.

- **new_file**
  The path name of a file or existing directory into which the command is to copy the cloned files. A file name can be a star name. If you specify a directory, the command creates the cloned files in that directory. If you specify a file name, the command creates the cloned files with that name.

- **-delete**
  By default, the command asks you to confirm whether to delete a file if its name conflicts with the name specified in new_file. If you specify -delete, the command deletes a specified file without first asking for confirmation.

**Explanation**

The clone_file command has the same effect as copying a file except that the clone_file command creates an empty file. Access lists, indexes, expiration date, and all attributes of the file such as safety-switch settings, implicit locking, allocation size, open options, and so on are identical. In other words, the cloned file has all of the original file’s attributes but none of its data.

See the copy_file command for a description of how star names function.
Related Information

clone_dir, copy_dir, copy_file
**Purpose**

This command compiles an OpenVOS COBOL source module.

**Display Form**

```
------------------------------------- cobol ------------------------------------
source_file_name: ____________________________  cobol ____________________________
-default_sign: ----------------------------- trailing_embedded
-define: -----------------------------------
-processor: default
-mapping_rules: default
-language: compatible -level: none
-list: no -full: no
-xref: no -xref_format: standard
-table: no -production_table: no
-optimize: yes -optimization_level: 3
-check_uninitialized: no -check: no
-format: no -mapcase: no
-profile: no -cpu_profile: no
-statistics: no -silent: no
-default_comp: none -debugging_mode: no
-default_class: static -main: no
-segmentation: no -ansi_replace: no
```
Command Line Form

cobol source_file_name
  [-default_sign sign_type]
  [-define variable_name...]
  [-processor processor_string]
  [-mapping_rules mapping_string]
  [-language language_string]
  [-level number]
  [-list]
  [-full]
  [-xref]
  [-xref_format format]
  [-table]
  [-production_table]
  [-no_optimize]
  [-optimization_level number]
  [-check_uninitialized]
  [-check]
  [-format]
  [-mapcase]
  [-profile]
  [-cpu_profile]
  [-statistics]
  [-silent]
  [-default_comp number]
  [-debugging_mode]
  [-default_class class]
  [-main]
  [-segmentation]
  [-ansi_replace]

Arguments

- **source_file_name** Required
  An OpenVOS COBOL source module.

- **-default_sign sign_type** [CYCLE]
  Sets the default sign for a compilation. Possible values for sign_type are leading_separate, trailing_separate, leading_embedded, and trailing_embedded. By default, the command sets the default sign to trailing_embedded.

- **-define variable_name**
  Defines variables to be used by the preprocessor. These variables are used during the preprocessor phase of the compilation. Preprocessor variables can contain letters, digits, or the underline character (_), in any position. (See the Explanation section of this command description or the description of the preprocess_file command for details.)
-processor \textit{processor\_string} \texttt{[CYCLE]}

Specifies the processor on which the program module (.pm) is to run. The display form for the \textit{-processor} argument restricts the values that you can choose to values for the processor family of the current module.

If the current module uses a processor from the IA-32 family, or if you specify, on the command line, the \textit{-processor} argument with the \texttt{pentium4} value, the allowed \textit{processor\_string} values are as follows:

- \texttt{default}
- \texttt{pentium4}

The \texttt{default} value indicates the system-wide default. Unless your system administrator has reset this value, \texttt{default} is \texttt{pentium4} for modules using IA-32 processors. To determine the \texttt{default} value, issue the \texttt{display\_error m$default\_processor} command. By default, the compiler produces code intended for the processor specified by \texttt{default}.

- \texttt{-mapping\_rules \textit{mapping\_string}} \texttt{[CYCLE]}

Specifies one of the following data alignment rules for a given compilation.

- \texttt{default}
- \texttt{default/check}
- \texttt{shortmap}
- \texttt{shortmap/check}
- \texttt{longmap}
- \texttt{longmap/check}

The \texttt{default} value indicates the system-wide default. The default alignment method is site-settable. To determine the \texttt{default} value, issue the \texttt{display\_error m$default\_mapping} command. By default, the compiler uses the data alignment rules specified by \texttt{default}. (See the \texttt{Explanation} section of this command description for details.)
-language language_string

Specifies one of the following levels of ANSI COBOL at which the program is to be compiled.

<table>
<thead>
<tr>
<th>Language</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>compatible</td>
<td>Provides COBOL-85 functionality that is compatible with COBOL-74, except for the introduction of new reserved words.</td>
</tr>
<tr>
<td>ansi85</td>
<td>Provides complete COBOL-85 support, except for the use of call and cancel statements to reference nested programs and static reinitialization of end-of-perform ranges.</td>
</tr>
<tr>
<td>ansi74</td>
<td>Provides full ANSI COBOL-74 support and some operating system extensions.</td>
</tr>
<tr>
<td>full_ansi85</td>
<td>Provides complete COBOL-85 support.</td>
</tr>
</tbody>
</table>

For more information on the differences between these levels, see the VOS COBOL User’s Guide (R142) or the VOS COBOL Language Manual (R010).

By default, the program is compiled at the compatible level.

-level number

Determines the level of error to report, depending on the value of language_string. If you specify ansi74 in the -language argument, this displays a message whenever the compiler detects a language element whose 1974 Federal Information Processing Standard (FIPS) level exceeds number. If you specify any other value for language_string, the level of errors reported is based on the 1985 ANSI standard. The compiler also puts the messages in any listing it produces. By default, the compiler does not check for validation levels.

-list

Creates a compilation listing. A compilation listing shows all source statements from the source module and include files, as well as a summary of all data definitions and the path names of include files used. You need not specify -list if you specify -full or -xref, since those arguments create a compilation listing in addition to other listings. By default, the compiler does not generate a compilation listing.

-full

Creates from the compiled object code an assembly language listing (with added comments) in addition to a compilation listing. By default, the compiler does not produce an assembly language listing.

-xref

Creates a compilation listing and a cross-reference listing of all data actually referenced in the program. The cross-reference information is sorted for Level-1 group identifiers, unless you specify otherwise with -xref_format. By default, the compiler does not generate a cross-reference listing.
-xref_format format

Specifies one of the following formats for the cross-reference listing.

<table>
<thead>
<tr>
<th>Format</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>standard</td>
<td>The cross-reference information is sorted for Level-1 group identifiers. If you specify -xref without specifying -xref_format, the information is displayed in this format.</td>
</tr>
<tr>
<td>sorted</td>
<td>The cross-reference information is sorted for all names in the program rather than only by Level-1 group identifiers. This value gives no allocation information; it gives only lines on which an identifier is defined and lines on which it is referenced.</td>
</tr>
<tr>
<td>all_sorted</td>
<td>The cross-reference information is sorted for all names in the program, whether or not they are referenced.</td>
</tr>
</tbody>
</table>

This argument is ignored unless you also specify -xref.

-table

Creates a symbol table in the object module, for use by the OpenVOS Symbolic Debugger. The compiler also performs some related operations. (See the Explanation section of this command description for details.) In addition, -table suppresses interstatement code optimization, which results in code that is slower than normal. Specifying -table sets the maximum optimization level to 1, unless you explicitly set the level to 0. By default, the compiler does not create a symbol table, suppress interstatement code optimization, or perform any related operations.

Note: A symbol table greatly increases the size of an object module.

If you specify both -production_table and -table, the compiler produces only a production table and sets the maximum optimization level to 3, unless you explicitly specify some other value.

.production_table

Creates a symbol table in the object module, for use by the OpenVOS Symbolic Debugger in a production environment. Only variables actually referenced in the program are placed in the symbol table. The compiler also performs some related operations. (See the Explanation section of this command description for details.) Unlike -table, .production_table does not suppress interstatement code optimization. As a result, the set and continue requests of the debug command can lead to unpredictable results. Also, the contents of variables in registers cannot be accurately displayed with the display request of the debug command. In addition, if the optimization level is greater than 2, the contents of any variables may not be accurately displayed with the display request of the debug command. Specifying .production_table sets the maximum optimization level to 3, unless you explicitly specify some other value. By default, the compiler does not create a symbol table, suppress interstatement code optimization, or perform any related operations.

Note: A symbol table greatly increases the size of an object module.
If you specify both -production_table and -table, the compiler produces only a production table and sets the maximum optimization level to 3, unless you explicitly specify some other value.

- **-nooptimize**
  Generates the object code without optimizing it. Optimization produces more compact object code by removing unnecessary or redundant computations. Specifying -nooptimize sets the optimization level to 0. This overrides any other specification of the optimization level. By default, the compiler optimizes the object code.

- **-optimization_level number**
  Specifies the degree of optimization. The possible values are 0, 1, 2, 3, and 4. (See the Explanation section of this command description for details.)

- **-check_uninitialized**
  Issues diagnostics for all references to uninitialized variables if you also specify an optimization level of 3 or 4. If you specify this argument and an optimization level of less than 3, the compiler issues an error. This argument is useful when verifying new code or checking for possible bugs, but it can return misleading diagnostics, as in the case of variables that are initialized within a conditional statement. The categories of uninitialized variables diagnosed by the compiler vary, depending on whether you choose both -check_uninitialized and an optimization level of at least 3, or choose only an optimization level of at least 3. By default, the compiler does not check variables for initialization.

- **-check**
  Checks for out-of-bounds array subscripts, indexes, and reference modifiers. The compiler performs these checks while compiling and inserts code to check further when the program is run. By default, the compiler does not check or insert checking code.

- **-format**
  Compiles a source module whose textual format differs from the standard COBOL format. By default, the source module must conform to the COBOL indentation rules.

- **-mapcase**
  Interprets all uppercase letters, except those in nonnumeric literals that are not the first operand of a call statement, as lowercase letters. If you specify -mapcase, and the source module contains an external variable name or entry name, you may not be able to bind the resulting object module. (See the Explanation section of this command description for details.) By default, the compiler distinguishes between uppercase and lowercase letters, and reserved words must be in lower case.

- **-profile**
  Inserts code in the compiled program that counts the number of times each source statement is executed when the program runs. (See the description of the profile and add_profile commands.) By default, the compiler does not insert the counting code. You cannot specify both -profile and -cpu_profile in the same command.
-cpu_profile

Inserts code in the compiled program that counts the number of times each source statement executes, the amount of CPU time spent executing each statement, and the number of page faults taken executing each statement when the program runs. (See the description of the profile and add_profile commands.) By default, the compiler does not insert the counting code. Note that the code inserted by this argument uses much more CPU time, but provides more useful information, than the code inserted by -profile. You cannot specify both -cpu_profile and -profile in the same command.

-statistics

Displays the following statistics about the compilation as it proceeds:

- version number of the compiler
- elapsed CPU time
- elapsed real time
- number of page faults taken
- amount of storage used

You can specify -statistics to see the progress of the compilation and to determine the phase in which an error occurs. If the compiler produces a listing, it puts the statistics in the listing. By default, the compiler does not display compilation statistics.

-silent

Suppresses the warning messages of severity-1 or severity-0 errors on your terminal during compilation. The compiler, nevertheless, puts the messages in an error file and in any listing it produces. By default, the compiler writes all error messages on your terminal.

-default_comp number

Produces data in the specified comp format where number is an integer from 1 through 6. By default, the compiler selects the most efficient computational type for each variable defined with computational usage that is compatible with the variable’s picture.

-debugging_mode

Compiles the program as though the debugging-mode clause were specified in the special-names paragraph.

-default_class class

Specifies how uninitialized working-storage data should be allocated. The possible values are static and auto. By default, the compiler allocates all uninitialized working-storage data as static.

-main

Compiles the source module in a form that identifies the program as a main program, thereby allowing the setting of external switches. By default, the source module is not compiled as a main program. This argument is infrequently used.

-segmentation

Inserts code in the object module that initializes every independent segment each time control is transferred to it, if required. An independent segment requires initialization.
if it contains, for example, an alterable go to statement. By default, the compiler does not insert the code. This argument is infrequently used.

-ansi_replace

Processes replace statements as defined by ANSI X3.23-1985. By default, simple replace statements are treated as $replace compiler directives.

Explanation

The `cobol` command compiles an OpenVOS COBOL source module into an object module.

The name of the source module must have the suffix `.cobol`; you can either supply or omit the suffix when you give `source_file_name`. The compiler generates an object module, puts it in your current directory, and names it. The name of the object module is the name of the source file with the suffix changed from `.cobol` to `.obj`.

When you are compiling programs for an ftServer module at all optimization levels, the module on which you are compiling must have at least 30,000 pages of paging partition available to avoid running out of virtual memory. In addition, the module on which you are compiling should have 64MB of physical memory available to achieve optimal compiler performance.

Using the -define Argument

The `define` argument defines variables to be used during the preprocessor phase of the compilation. For example, if you specify the following on the command line, the preprocessor variables `var_a` and `var_b` will be initially defined during the preprocessing phase of the compilation:

```
cobol prog1 -define var_a var_b
```

You use preprocessor variables with preprocessor statements to perform conditional compilation on a program. Conditional compilation enables you to switch on or off various statements in a program. This is useful, for example, if you want your program to compile different lines of source code on different processors. There are six preprocessor statements.

- @define
- @undefine
- @if
- @else
- @elseif
- @endif

Preprocessor statements must begin in the first column of the source program. Therefore, indentation of nested @if statements is not allowed.

A preprocessor statement must be contained on a single line. A line containing a preprocessor statement cannot contain comments or parts of the source language. (An exception is the @endif statement, which ignores any text following it on the line, thus allowing you to comment on the source code.)
For more information on the preprocessor, see the description of the `preprocess_file` command.

### Using the `-processor` Argument

The `-processor` argument allows you to specify the processor on which the program is to run. The `-processor` argument also allows you to perform cross-compilation on a source module if the COBOL cross compiler is available on your system. Cross-compilation occurs when a compiler running on one processor family translates a source module into object code for another processor family. The IA-32 cross compiler generates code to run on fhsServer modules. Specify the value `pentium4` for the `-processor` argument to target an fhsServer module.

Depending on the value specified in the `-processor` argument, the compiler automatically defines one preprocessor variable for the processor family and one or more preprocessor variables corresponding to the processor type(s), as shown in Table 2-12.

#### Table 2-12. Predefined Preprocessor Variables

<table>
<thead>
<tr>
<th>Processor Value</th>
<th>Preprocessor Variable</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>default</code></td>
<td>Varies, depending on the default system processor</td>
</tr>
<tr>
<td><code>pentium4</code></td>
<td><code>__PENTIUM4__</code>, <code>__IA32__</code>, and <code>__i386</code></td>
</tr>
</tbody>
</table>

If you specify `-processor `pentium4` on the command line, the preprocessor variables `__PENTIUM4__`, `__IA32__`, and `__i386` are defined.

If the value specified in the `-processor` argument indicates the IA-32 processor, the maximum number of bytes available for each function’s initial stack frame is 2,147,483,584 bytes.

The amount of automatic storage you can actually declare is somewhat less than these limits because temporary variables generated by the compiler also count towards the limit.

**Note:** Although the OpenVOS COBOL compiler supports extremely large values (such as 2,147,483,646), the operating system does not support them.

### Using the `-mapping_rules` Argument

The `-mapping_rules` argument allows you to specify the data alignment rules for a given compilation. The value `default` indicates the system-wide default. The default is site-settable. The value `shortmap` specifies that the shortmap alignment rules are to be used for the source module. The value `longmap` specifies that the longmap alignment rules are to be used for the source module. The values `default/check`, `shortmap/check`, and `longmap/check` are equivalent to `default`, `shortmap`, and `longmap`, respectively, except that they also diagnose alignment padding within records. For example, if you specify `default/check`, the compiler displays a severity-0 message stating how many bytes of padding exist between fields within a record. The using `longmap/shortmap` clause overrides `-mapping_rules` values, but alignment padding within records is still diagnosed if you specify one of the checking values.
For more information on data alignment rules, see the VOS COBOL Language Manual (R010).

**Using the -full, -list, or -xref Argument**

If you specify the -full, -list, or -xref argument, the compiler creates a compilation listing file and puts it in your current directory. The name of the compilation listing is `source_file_name.list`. The -full argument creates an assembly language listing in addition to a program listing. The -xref argument creates a list of cross-references in addition to a program listing.

Normally, data defined as computational in COBOL gets a type based on the picture description. The compiler selects the best data type for the picture description. However, with the -default_comp argument, you can specify explicit computational values. For example, computational-3 and computational-6 are reasonable defaults because they can store data having any valid numeric picture; computational-3 data is packed.

**Using the -table or -production_table Argument**

If you specify the -table argument, the compiler creates a symbol table, and allocates storage and generates addresses for all external references, including any that are not used. Symbol-table capacity is 2,147,483,647 nodes. The compiler generates internal subroutines that calculate size, offset, and bound expressions that determine the characteristics of adjustable data. This allows the OpenVOS Symbolic Debugger to display and modify variable-length data according to its current length. In addition, the compiler suppresses interstatement code optimization. Code produced with -table executes more slowly than code produced with -production_table.

If you specify the -production_table argument, the compiler performs all of the same operations, except that it does not suppress interstatement code optimization, and only variables actually referenced in the program are placed in the symbol table (most unreferenced variables are from include files). Code produced with -production_table can yield unpredictable results if you invoke the OpenVOS Symbolic Debugger set and continue requests.

**Optimizations for ftServer Modules**

As mentioned in the previous section, the -optimization_level argument allows you to optimize programs at different levels. When you are compiling a source module to run on ftServer modules, the levels of optimizations are 1, 2, 3, and 4. Specifying optimization level 3 or 4 causes the compiler to perform level 3 optimizations.

If you specify optimization level 0, the compiler performs the following local optimizations.

- local register allocation
- elimination of unreachable code
If you specify optimization level 1, the compiler performs all level 0 optimizations plus the following other local optimizations.

- local pattern replacement
- short-circuit evaluation of Boolean expressions
- recognition of algebraic identities
- constant folding
- local combination of common subexpressions within a statement
- peephole optimizations within a single statement
- result incorporation

If you specify optimization level 2, the compiler performs all level 1 optimizations plus the following global optimizations.

- branch retargeting
- global combination of common subexpressions
- removal of invariant expressions from loops
- subsumption
- peephole optimizations across statement boundaries
- global register allocation

If you specify optimization level 3, the compiler performs all level 2 optimizations plus the following global optimizations.

- constant propagation
- removal of invariant assignments from loops
- strength reduction
- linear test replacement
- elimination of dead assignments
- elimination of useless loops
- check for uninitialized variables
- elimination of dead code and dead stores
- inline expansion
- instruction scheduling

**Using the -no_optimize, -table, or -optimization_level Argument**

The level of optimization is determined by the arguments -no_optimize, -table, and -optimization_level. Specifying -no_optimize sets the optimization level to 0. Specifying -table sets the level to 1, unless you explicitly set the level to 0. The -optimization_level argument sets the level to any of the permitted levels: 0, 1, 2, or 3.

The compiler sets the actual level to the lowest level set by any of the three arguments. By default, the level is 3.

**Note:** If you compile a program with either the -profile or -cpu_profile argument, you must specify an optimization level lower than 3. Otherwise, -profile or -cpu_profile might not return accurate information, since high optimization levels can cause code to be moved from one statement to another.
Using the -check_uninitialized Argument

The optimization level for a source module also affects the functionality of the -check_uninitialized argument.

- If you specify the -check_uninitialized argument and an optimization level of at least 3, the compiler diagnoses instances of variables that it knows are uninitialized as well as some instances of variables that may be uninitialized (that is, variables that are initialized as part of code executed conditionally).

- If you do not specify the -check_uninitialized argument but do select an optimization level of at least 3, the compiler diagnoses instances of variables within the function definition that it knows are uninitialized. In this case, the compiler does not issue a diagnostic for a variable that is initialized as part of code executed conditionally.

- If you specify an optimization level of less than 3, either explicitly or implicitly (such as by specifying the -table argument), the compiler issues an error and does not diagnose uninitialized variables even if you select -check_uninitialized.

Using the -mapcase Argument

When you compile a source module using the -mapcase argument, and the module contains an external variable name or entry name with one or more uppercase letters, you may not be able to bind the resulting object module. If the binder encounters a reference to the original name (for example, in a binder control file), it will not recognize the original name and its lowercase version as the same name.

Interpreting Compiler Diagnostics

If the compiler discovers any errors in your source module, it displays an error message on your terminal. Severity-1 and severity-0 messages are not displayed on your terminal when you specify the -silent argument. The compiler also creates an error file named source_file_name.error in the current directory and writes the error messages to the file. The compiler also appends error messages to a compilation listing if it produces one. Any .error file is deleted by the system if a subsequent compile to the same source file is successful (contains no errors).

The OpenVOS COBOL compiler diagnoses five types of errors.

SEVERITY 0: Advice
SEVERITY 1: Warning
SEVERITY 2: Correctable error
SEVERITY 3: Uncorrectable error: translation can continue
SEVERITY 4: Uncorrectable error: translation cannot continue

The text of the error message explains the cause of the error.

A severity-0 error, although valid COBOL, indicates that improvement is possible, usually in the area of performance. The source module is syntactically correct, so the compiled object module can be bound and executed, but probably with less than optimum efficiency.

A severity-1 error, although valid COBOL, is probably a programming error. Since the source module is syntactically correct at the point of a severity-1 error, however, the compiler
continues to compile the source. The compiled object module can be bound and executed, but the program probably will not perform as expected.

A severity-2 error is invalid COBOL, but the compiler can reinterpret the source in such a way that it can continue to compile the program. The compiler proceeds as if the faulty code were replaced with the most likely syntactically correct code. The compiled object module can be bound and executed, but it probably will not perform as expected.

A severity-3 error is invalid COBOL, and the compiler cannot reinterpret the source in such a way that it can continue to compile the program into a usable object module. Nevertheless, the compiler continues to process the program to detect any additional errors. However, the object module is not created.

A severity-4 error is invalid COBOL, and the compiler cannot reinterpret the source in such a way that it can continue to process the program from the point of the severity-4 error. The object module is not created.

**Note:** If the compilation results in more than 100 errors, in any combination (excluding severity-0 errors), compilation terminates.

The compiler always overwrites an existing object module having the same name as the object module it produces.

**Access Requirements**

You need read access to a source module to compile it. You need modify access to the directory from which you are issuing the command, in which the .obj file will be created.

**Examples**

This command compiles the OpenVOS COBOL source module `make_report.cobol` in the current directory, using three arguments.

```bash
cobol make_report -list -table -mapcase
```

The compiler interprets uppercase letters according to the description of the -mapcase argument. The object module `make_report.obj` is created and put into the current directory. A symbol table is produced. The compiler creates the compilation listing `make_report.list` and puts it in the current directory. If the compiler finds any errors, it creates the error file `make_report.error`, writes error messages to it, and puts the file in the current directory.

**Related Information**

See the *VOS COBOL User’s Guide* (R142) for more information on compiling OpenVOS COBOL programs. See the *VOS COBOL Language Manual* (R010), for a complete description of the OpenVOS COBOL language.
compare_dirs

Purpose
This command compares two directories and reports the differences.

Display Form

<table>
<thead>
<tr>
<th>Argument</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>directory_A:</td>
<td></td>
</tr>
<tr>
<td>directory_B:</td>
<td></td>
</tr>
<tr>
<td>-output_path:</td>
<td></td>
</tr>
<tr>
<td>-data_compares:</td>
<td>logical</td>
</tr>
<tr>
<td>-start_record:</td>
<td>1</td>
</tr>
<tr>
<td>-end_record:</td>
<td></td>
</tr>
<tr>
<td>-check_acls:</td>
<td>yes</td>
</tr>
<tr>
<td>-check_block_count:</td>
<td>no</td>
</tr>
<tr>
<td>-check_times:</td>
<td>no</td>
</tr>
<tr>
<td>-check_author:</td>
<td>no</td>
</tr>
</tbody>
</table>

Command Line Form

```
compare_dirs directory_A
directory_B
[-output_path output_path_name]
[-data_compares compare_type]
[-start_record start_record]
[-end_record end_record]
[-no_check_acls]
[-check_block_count]
[-check_times]
[-check_author]
```

Arguments

- `directory_A` Required
  The path name of one directory to be compared. The directory is referred to as the A-directory.

- `directory_B` Required
  The path name of the other directory to be compared. The directory is referred to as the B-directory.

- `-output_path output_path_name`
  Directs the command output to the sequential file or device specified by `output_path_name`. If `output_path_name` does not exist, the command creates
that file in the specified directory. By default, the command directs output to your default_output port.

- `-data_comparisons compare_type`  
  Compares directories by different categories of data. Possible values for compare_type are logical, by_block_first, by_block_only, and none. By default, the command makes a logical comparison.

- `-start_record start_record`  
  Starts comparing the files in the two directories at the records numbered start_record. By default, the command starts at the first record in each file. This argument lets you disregard insignificant header information, such as a date that may vary from file to file.

- `-end_record end_record`  
  Stops comparing the files in the two directories at the records whose numbers are end_record. By default, the command stops after the last record.

- `-no_check_acls`  
  Omits comparing the access control lists of files and subdirectories in the two directories. By default, the command compares the access control lists of a pair of files as it compares the files, and it compares the access control lists and default access control lists of subdirectories. It also compares the default access control lists of the A-directory and the B-directory.

- `-check_block_count`  
  Compares the number of storage blocks occupied by files in the two directories. By default, the command does not compare the block counts as it compares the files.

- `-check_times`  
  Compares the times that all objects in the two directories were created, last modified, last saved, and last used. By default, the command does not compare the times.

- `-check_author`  
  Compares the authors of files in the two directories. By default, the command does not compare the authors as it compares the files.

**Explanation**

The `compare_dirs` command compares the contents of two directories you specify and reports on the differences in a selected set of attributes.

The `compare_dirs` command truncates an output file before writing the differences to it. No message is sent if the objects are equal.

See the description of the `compare_files` command for an explanation of file comparison.
The `-data_compares` argument allows you to specify one of several methods of comparing the files in the directory.

- By default, the value is `logical`. In this case, comparison is by logical records.

- If you select the value `by_block_only`, the comparison is by disk blocks (4096 bytes). The comparison in this case is faster than in the `logical` case; however, logically identical files can have different disk block contents.

- If you select the `by_block_first` value, the comparison is first between disk blocks and, if the files differ, between logical file content. If they still differ, it is a reportable difference.

Comparison by block first is intended to permit a faster comparison than that permitted by the `logical` value alone. However, if there are a great many differences in the logical content of the directories compared, the `by_block_first` value is slower than the `logical` value. If you select the `none` value, data in indexes and files are not compared; only file attributes are compared.

**Access Requirements**

You need status access to the A-directory and the B-directory and either read access to a file being compared or modify access to a superdirectory of the A-directory or the B-directory containing a file to be compared. The latter allows the `compare_dirs` command to override any access on the file for you while it compares the files; access is always left in its original state when the command finishes.

**Examples**

This command compares the two specified directories.

```
compare_dirs customers <Smith>prospects
```

The `compare_dirs` command reports the differences to the `default_output` port, which is normally attached to your terminal.

**Related Information**

For more information about directory management, see the command descriptions of `change_current_dir`, `copy_dir`, `create_dir`, `delete_dir`, `display_current_dir`, and `move_dir`. 

```
**compare_files**

**Purpose**
This command compares one or more files with another file and reports the differences.

**Display Form**

```
compare_files

file_A:  
path_B:  
-ascii_records:  yes
-ignore_blank_lines:  yes
-binary_records:  no
-index_for_A:  
-index_for_B:  
-check_block_count:  no
-check_times:  no
-check_author:  no
-use_random_access:  no
-compare_blocks:  no
-compare_all_blocks:  no
-continue:  no
-output_path:  
-start_record:  1
-end_record:  
-difference_count:  
-line_numbers:  yes
-file_A_is_template:  no
```
Command Line Form

```
compare_files file_A
  [path_B]
  [-no_ascii_records]
  [-no_ignore_blank_lines]
  [-binary_records]
  [-index_for_A index_for_A]
  [-index_for_B index_for_B]
  [-check_block_count]
  [-check_times]
  [-check_author]
  [-use_random_access]
  [-compare_blocks]
  [-compare_all_blocks]
  [-continue]
  [-output_path output_path_name]
  [-start_record start_record]
  [-end_record end_record]
  [-difference_count difference_count]
  [-no_line_numbers]
  [-file_A_is_template]
```

Arguments

- `file_A`  
  Required  
  The name or star name of a file or files to be compared with the file or files specified by `path_B`. The file is referred to as an A-file.

- `path_B`  
  Specifies the file or files to be compared with the A-files by giving either their file names or the name of the containing directory. A file selected by this argument is referred to as a B-file. If you specify a directory, the command compares each file in that directory whose name matches the name of some A-file with the A-file it matched. A `path_B` term that is a file name can be a star name. When `path_B` is a star name, the command compares each B-file whose name matches an A-file with that A-file, using star name pairs to determine matches where appropriate. By default, the command proceeds as if you have specified the name of your current directory.

- `-no_ascii_records`  
  Suppresses the display of ASCII data in any records in the compared files that differ. By default, the command displays the records as ASCII text.

- `-no_ignore_blank_lines`  
  Includes empty lines and trailing spaces in a record when comparing files. By default, empty lines and trailing spaces are disregarded when comparing the files.

- `-binary_records`  
  Displays as hexadecimal integers the data in any records in the compared files that differ. By default, `no_ascii_records` defines the way the command displays the differing records.
compare_files

- **-index_for_A** *index_for_A*
  Specifies an index for the designated A-files. The command uses the index to define the order of the records in the A-file. By default, the command accesses the records in the order in which they were written to the file.

- **-index_for_B** *index_for_B*
  Specifies an index for the B-files. The command uses the index to define the order of the records in the B-file. If you specify an equal sign (=) for *index_for_B*, the command uses *index_for_A*. By default, the command accesses the records in the order in which they were written to the file.

- **-check_block_count**
  Compares the block counts of the compared files. By default, the command disregards the block counts when comparing the files.

- **-check_times**
  Compares the times when the compared files were created, last modified, last saved, and last used. By default, the command disregards these times when comparing the files.

- **-check_author**
  Compares the authors of the files. By default, the command disregards the authors when comparing the files.

- **-use_random_access**
  Accesses the files being compared in random access mode. By default, the command accesses the files in sequential access mode or indexed access mode.

- **-compare_blocks**
  Compares files block-by-block. This argument is useful for comparing files containing binary data. By default, the comparison stops at the first block of the files being compared that contains differences. If you specify **-compare_blocks** and **-continue**, the file comparison does not stop after the command finds a difference.

  **Note:** Structured files (that is, files other than stream files) may be logically identical, but their blocks are not necessarily identical.

- **-compare_all_blocks**
  Compares all blocks in the two files, including unallocated blocks in sparse files.

- **-continue**
  Determines whether the command continues to compare files after finding a difference. If you do not specify this argument, a stream file that contains records longer than 32,767 is considered to contain binary data, and the file comparison stops after the first difference that follows the occurrence of a long record.

  If you specify **-compare_blocks** or **-compare_all_blocks**, **-no_continue** is the default. Otherwise, **-continue** is the default.

  You can specify **-no_continue** to stop any type of file comparisons after the first difference. This argument produces different behavior than if you specify **-difference_count** with a value of 1. With **-difference_count 1**, if the
command finds a difference, it displays that difference and continues to compare the files and shows the totals. With -no_continue, if the command finds a difference, it stops without analyzing the remainder of the file.

- output_path output_path_name
  Directs the output from the command to the sequential file or I/O device output_path_name. If the file does not exist, the operating system creates it. By default, the command writes the output to your default_output port.

- start_record start_record
  Starts comparing the files at the records numbered start_record. By default, the command starts comparing the files at the first record in each file.

- end_record end_record
  Stops comparing the files at the records numbered end_record. By default, the command compares the files through the last record.

- difference_count difference_count
  Sets the maximum number of differences the command displays. This argument allows you to restrict the amount of output the command produces when comparing large files. Specify the value zero to print the number of differences when comparing files, if that is the only information that is needed. By default, the command displays all records in the compared files that differ.

- no_line_numbers [CYCLE]
  Suppresses the display of line numbers with the differences between the files.

- file_A_is_template [CYCLE]
  Indicates that records in the file specified by file_A contain templates that do pattern matching when comparing against similar records in path_B.

Explanation

The compare_files command allows you to compare several sets of files simultaneously. The first argument, specifying a set of A-files, can be a star name. The optional second argument, path_B, can be the name of a file or directory; a file name can be a star name.

See the Explanation of the copy_file command for a description of how star names function.

If path_B specifies the name of a file or set of files, those files are the B-files for all the comparisons; star name pairs are used where necessary to determine name matches. If path_B specifies the name of a directory, either explicitly or by default, the compare_files command looks in that directory for files with names matching the selected A-files to use as the B-files.

If you specify a link for path_A and specify a star name for path_B, be careful that the link does not point to a file that is identified by the star name, or the command may determine that path_A and path_B are identical. For example, if you have a subdirectory called subd, a
The compare_file command behaves as follows when you specify path_A as a link and path_B as a star name.

```
compare_files xxx subd>*
```

Warning: Comparing %sys#m1>Sales>Joe_Smith>subd>yyy against itself.

The command expands the link xxx to subd>yyy and uses the expanded link name to resolve the star name to subd>yyy. However, if you specify path_A as a link and path_B as a directory name, the command behaves as follows.

```
compare_files xxx subd
```

A (%sys#m1>Sales>Joe_Smith>subd>yyy) does not match B (%sys#m1>Sales>Joe_Smith>subd>xxx).

The command expands the link xxx to subd>yyy and uses the unexpanded link name to resolve the star name to subd>xxx.

You can specify the -ascii_records and -binary_records arguments together. In that case, the compare_files command displays records that differ both as hexadecimal numbers and as ASCII text.

If you specify the arguments -no_ascii_records and -no_binary_records, the compare_files command displays file attributes, but does not display any records that differ. It stops comparing the files when it finds a difference, and returns to the command line.

The output file shows the differences in file attributes such as file type, locking mode, record size, etc. Any records that do not match are printed in full. The records are numbered according to file; A1 is the first record of the first file. If the length of the record is less than 150 characters, the record wraps; see the first example. If the length of the record is greater than 150 characters, each line begins with the record number and the hexadecimal number of the first character’s position within the record.

No message is displayed if the objects are equal. When the comparison is completed, you return to command level.

If compare_files sees a record longer than 32,767 bytes, it assumes that the files contain binary data. If you specify -ascii_records, the command displays binary data as well (as if you had specified both -ascii_records and -binary_records). When the command finds a difference, it stops comparing the files, returns to the command line, and returns the error code e$long_stream_file_record (7843).

Even with long records, compare_files continues its comparison as long as it does not find any differences. If the files are identical, the command returns 0 (as command_status). You can specify the -continue argument to continue the comparison when stream files contain records longer than 32,767 bytes or after the first mismatched block when the -compare_blocks argument is specified.

When comparing binary stream files, you should specify -no_ignore_blank_lines. When compare_file encounters a record longer than 32,767 bytes, the command will no longer ignore blank lines, even if you specified -ignore_blank_lines (the default).
The **-compare_blocks** argument compares allocated blocks. This argument is useful when comparing binary stream files, since 4096-byte disk blocks are compared and the discrepancies are reported in terms of the block number. You can also specify the **-continue** argument if you want to continue comparing blocks after the first difference. Doing so allows you to see the total number of blocks that are different as well as their ranges.

The **-compare_all_blocks** argument compares all blocks, treating unallocated blocks in sparse files as either blocks of binary zeros (for 64-bit stream files) or binary ones (for all other file types). This is guaranteed to produce accurate results despite differences in the underlying extent structure of the files. This comparison can take significantly longer than a comparison using **-compare_blocks** if the file is sparsely allocated, but it is still faster than a record-by-record comparison.

If you specify the **-file_A_is_template** argument, the records in file A may consist of ordinary data bytes and metasymbols. All metasymbols begin with the characters { and end with the characters }. The metasymbols tell the command how to match zero or more bytes in the corresponding record in path B. The command currently cannot perform metasymbol matching while doing differential file comparisons. When differences are found, the metasymbols appear as differences in the output. In the unlikely situation that compare_files can resynchronize before reaching the end of the file, it resumes metasymbol processing.

If you specify the **-file_A_is_template** argument, the compare_files command recognizes the metasymbols listed in Table 2-12.

**Table 2-12. Metasymbols Used by compare_files**

<table>
<thead>
<tr>
<th>Metasymbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>{~}</td>
<td>Escape convention: Matches ~.</td>
</tr>
<tr>
<td>{-ANY_NAME-}</td>
<td>Matches any alphanumeric name.</td>
</tr>
<tr>
<td>{-CURRENT_DIR-}</td>
<td>Matches the path name for the current directory.</td>
</tr>
<tr>
<td>{-CURRENT_DISK-}</td>
<td>Matches the disk name from the path name for the current directory.</td>
</tr>
<tr>
<td>{-HOME_DIR-}</td>
<td>Matches the path name for the home directory.</td>
</tr>
<tr>
<td>{-HOME_DISK-}</td>
<td>Matches the disk name from the path name for the home directory.</td>
</tr>
<tr>
<td>{-MASTER_DISK-}</td>
<td>Matches the path name of the master disk.</td>
</tr>
<tr>
<td>{-MODULE-}</td>
<td>Matches the current module name.</td>
</tr>
<tr>
<td>{-SYSTEM-}</td>
<td>Matches the current system name.</td>
</tr>
<tr>
<td>{-USER-}</td>
<td>Matches the current user name.</td>
</tr>
<tr>
<td>{-GROUP-}</td>
<td>Matches the current group name.</td>
</tr>
<tr>
<td>{-MONTH-}</td>
<td>Matches the current month name.</td>
</tr>
<tr>
<td>{-WEEK-}</td>
<td>Matches the current week name.</td>
</tr>
<tr>
<td>{-INTEGER-}</td>
<td>Matches an integer: 1 or more digits (0-9).</td>
</tr>
</tbody>
</table>
In the preceding table, `picture` consists of one or more picture characters. Each picture character matches one character in `path_B`. The following table lists the picture characters.

<table>
<thead>
<tr>
<th>Character</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Space</td>
<td>Matches a space character.</td>
</tr>
<tr>
<td>,</td>
<td>Matches , or a space character.</td>
</tr>
<tr>
<td>-</td>
<td>Matches - or a space character. If - occurs at the beginning of the picture and is followed by Z, ., or 9, it is a leading sign and will also match + or a digit (0-9).</td>
</tr>
<tr>
<td>.</td>
<td>Matches . or a space character.</td>
</tr>
<tr>
<td>/</td>
<td>Matches / or a space character.</td>
</tr>
<tr>
<td>9</td>
<td>Matches a single digit (0-9).</td>
</tr>
<tr>
<td>:</td>
<td>Matches : or a space character.</td>
</tr>
<tr>
<td>A</td>
<td>Matches an alphanumeric character (A-Z, a-z, 0-9, ., _,$).</td>
</tr>
<tr>
<td>D</td>
<td>Matches an exponent character (DEde) followed by an optional sign.</td>
</tr>
<tr>
<td>E</td>
<td>Matches an exponent character (DEde) followed by an optional sign.</td>
</tr>
<tr>
<td>F</td>
<td>Matches a single hex digit (0-9, a-f, A-F).</td>
</tr>
<tr>
<td>H</td>
<td>Matches a hex digit or a space.</td>
</tr>
<tr>
<td>X</td>
<td>Matches any single character.</td>
</tr>
<tr>
<td>Z</td>
<td>Matches a digit (0-9), a space, or a sign.</td>
</tr>
</tbody>
</table>

**Access Requirements**

You need read access to any files you compare.

**Examples**

The following examples illustrate some of the various ways in which you can invoke the `compare_files` command.
Example 1.
To compare the two files `customers.old` and `customers` in the current directory, use this command.

```
compare_files customers.old customers
```

The following output might result.

```
A (%s1#d02>Sales>Smith>customers.old) does not match B (%s1#d02>Sales>Smith>customers).
- Some attributes of the two files do not match:
  file organization relative file sequential file
  transaction file switch yes no
  record size 60 0
- Some attributes of the index `ix1` do not match:
  separate key switch no yes
A1 #01442912340876542876 Davis, M. L. 123 Riverside Drive
+Sheraton Ma 02000 413 555-0002
changed to
B1 #01442912340876542876 Davis, M. L. 456 Hill Street
+Oakley Ma 01000 617 555-0001
```

1 data difference(s).

Example 2.
Note that the length of each of the records in the output file is less than 150 characters.
Suppose you invoke the same command, but the records that differ are both longer than 150 characters. In this case, the following output file might result.

```
A (%s1#d02>Sales>Smith>customers.old) does not match B (%s1#d02>Sales>Smith>customers).
- Some attributes of the two files do not match:
  file organization relative file sequential file
  record size 100 0
A1 0000 #01442912340876542876 Davis, M. L. 123 Riverside Drive
A1 0040 Sheraton Ma 02000 413 555-0002 M
A1 0080 S 123 45 6789 IA
changed to
B1 0000 #01442912340876542876 Davis, M. L. 456 Hill Street
B1 0040 Oakley Ma 01000 617 555-0001 M
B1 0080 S 123 45 6789 IA
```

1 data difference(s).

The four-digit hexadecimal number following the record number is the character position of the first character on that line.

Example 3.
To compare all OpenVOS COBOL source modules in the directory `>east>Smith>source_progs` with source modules in the current directory having the same names, use this command.
compare_files

```
compare_files >east>Smith>source_progs>*.cobol
```

**Related Information**

For a description of how to compare program modules, see the description of the `compare_pm_var_file` in the manual *OpenVOS System Administration: Administering and Customizing a System* (R281). See also the description of the `compare_dirs` command.
consolidate_dir

Purpose

This command consolidates a directory’s blocks or author data.

Display Form

```
consolidate_dir

directory_names: [directory_names]
-trim: [yes]
-authors: [yes]
-brief: [no]
-revert: [no]
-ask: [yes]
```

Command Line Form

```
consolidate_dir directory_names
[ -no_trim]
[ -no_authors]
[ -brief]
[ -revert]
[ -no_ask]
```

Arguments

- **directory_names** Required
  The name or star name of the directory or directories to be consolidated. You must have modify access to `directory_names`.

- **-no_trim** [CYCLE]
  By default, the command removes unused blocks at the end of a directory. If you specify `-no_trim`, `-authors` must be set to `yes`.

- **-no_authors** [CYCLE]
  By default, the command compacts an author’s data. If you specify `-no_authors`, `-trim` must be set to `yes`.

- **-brief** [CYCLE]
  Displays information about the block/author status for each directory, both before and after the command executes. By default, the command does not display this information.
consolidate_dir

► -revert
Reverts expandable directories to standard directories after the consolidation occurs, if possible. A standard directory can contain no more than 527 data blocks and cannot have author entries. (Author entries are created in expandable directories when the author list has been filled.) If you specify this argument for a standard directory, the command ignores this argument unless you specify an explicit directory or if the star name expands to a single directory. In either case, the command displays a warning if it cannot revert the directory. If a directory is reverted, the command displays a message if you specify multiple directories or if you specify the -no_brief argument.

► -no_ask
By default, asks you for confirmation before attempting to consolidate a large directory that may require significant time to consolidate. If you specify -no_ask, the command does not ask you for confirmation first.

Explanation
The consolidate_dir command reduces the size of a directory (without any impact to subdirectories or files) by trimming unused blocks at the end. The command relocates the author list entry, which tends to migrate toward the end of the directory as it becomes full, to a lower block, if possible. Otherwise, no compaction occurs. This means that it is possible that the size of a sparsely populated directory cannot be effectively reduced, since consolidation success depends on entry distribution within the blocks. However, a directory consolidation takes only a few seconds, whereas a directory copy can take many minutes or longer. In addition, directory consolidation is nondisruptive and eliminates any concern about the contents of the new and old directory being identical.

This command does not reduce a directory size to less than 16 blocks.

When you move, copy, or save/restore a directory, the space used is compacted, since the directory is completely rebuilt and thus uses only as many blocks as needed. However, copying a high-level directory with many subdirectories has some disadvantages:

• It can be very time-consuming.

• It may have a negative impact on performance.

• Because a directory’s contents may be changing, the copy may not be identical to the original.

• You can use a save/restore operation to get an identical copy (and to retain author information), but the directory must not be modified after the save finishes and before the restore is complete.

Using the consolidate_dir command avoids the preceding issues.

Note: Because the directory is inaccessible during consolidation, avoid consolidating a directory that is very large or has many authors when other processes require ongoing time-critical access to the directory.
Examples

Consider the following example:

```
consolidate_dir .
Size of %s#raid8>mydir (12943 entries) may lead to access delays.
  Proceed? (yes, no) y
Last block in %s#raid8>mydir cannot be reduced to lower than 823.
ready  13:32:28  0.004  5
```

In the preceding example, the current directory’s last block number (823) contained data, and the command could not reduce the directory’s block usage because the command can trim only unused blocks at the end of a directory.

In the following example, `dir` is an expandable directory that is less than 16 blocks long (and thus will not be reduced) and has no author entries (and thus none to eliminate). If a directory is not eligible for a reduction in block usage or number of authors, the command does not display information about block usage/number of authors.

```
consolidate_dir dir -revert
Expandable directory %s#raid8>dir reverted to standard.
```

In the following example, the command displays a message for reverted directories unless you specify `-brief` or only one directory is applicable.

In the next example, `dir1`, `dir2`, and `dir3` are expandable directories. The last block of `dir1` is numbered 20, and the size cannot be reduced; `dir2` and `dir3` are larger. The last directory, `dir4`, is an empty standard directory.

```
consolidate_dir dir* -revert
Last block in %s#raid8>dir1 cannot be reduced to lower than 20.
Expandable directory %s#raid8>dir1 reverted to standard.
Size of %s#raid8>dir2 (17801 entries) may lead to access delays.
  Proceed? (yes, no) y
Last block in %s#raid8>dir2 cannot be reduced to lower than 1186.
Expandable directory %s#raid8>dir2 cannot be reverted to standard.
Size of %s#raid8>dir3 (17725 entries) may lead to access delays.
  Proceed? (yes, no) n
Last block in %s#raid8>dir4 reduced from 501 to 15.
```

Because `-revert` did not apply, the command did not display a message for `dir4`. If the request had been made explicitly for `dir4`, however, the command would have displayed a message:

```
consolidate_dir dir4 -revert
Directory %s#raid8>dir4 is already standard.
ready  13:34:22  0.012  4
```
consolidate_dir

The following examples show the use of authors.

```
consolidate_dir . -no_trim
Size of %s#raid8>exp (3702 authors) may lead to access delays.
    Proceed? (yes, no) y
Authors in %s#raid8>exp cannot be reduced to less than 3702.
  ready 16:37:37 0.012  5

consolidate_dir . -no_ask
Authors in %s#raid8>exp cannot be reduced to less than 3702.
Last block in %s#raid8>std2 reduced from 1186 to 277.
  ready 16:42:43 0.000  5
```

**Note:** The last block number of a directory (as shown in the preceding command) is the zero-based number of the highest data block used by the directory. Directory growth is limited, based on the number of data blocks; the number of data blocks in a directory (shown in `display_dir_status` output as `current_blocks`) is always one more than the last block number. The `display_dir_status` and `list` commands display the value `blocks used`, which represents all blocks used by the directory, including indirect blocks (that is, blocks that hold the address of other blocks, not data).

**Related Information**

See the descriptions of the `copy_dir`, `locate_expandable_dirs`, `move_dir`, `restore_object`, and `save_object` commands.
**convert_stream_file**

**Purpose**
This command converts the organization of an existing file to the specified file organization.

**Display Form**
```
----------- convert_stream_file -------------------
source_file: [ ]
destination: [ ]
  -to: [ ]
  -extent_size: [ ]
  -ask: [ ]
  -check: [ ]
  -safe_only: [ ]
  -brief: [ ]
  -long: [ ]
```

**Command Line Form**
```
convert_stream_file source_file
  [destination]
  [to value]
  [-extent_size [size]]
  [-no_ask]
  [-check]
  [-no_safe_only]
  [-brief]
  [-long]
```

**Arguments**

- **source_file**
  The path name or star name of the file to be converted.

- **destination**
  The path name of the converted file. If you do not specify *destination*, the converted file replaces *source_file*. The *destination* must not be the same name as *source_file*.

- **-to value**
  Specifies the file organization to which to convert *source_file*. Values are stream64 (the default), stream, sequential, and ext_sequential.
-extent_size [size]

Specifies the size, in blocks, of the extents used by the converted file. Extents are dynamically allocated and are a value between 8 and 256 that is also a power of two. See the Explanation for more information.

If you specify -extent_size with a value for size, the converted file’s extent size is the specified value. The size must be a valid DAE value, or it must be 1 (or 0, meaning no extents). If the converted file will be extended sequential, size must be 8 or greater.

If you specify -extent_size with no value, the command uses a default value:

- For sequential and stream files, the extent size is 1 (that is, no extents).
- For extended-sequential files, the extent size is 8.
- For 64-bit stream files, the extent size is the target module’s default value: normally flex (or 8 for modules running a release earlier than OpenVOS Release 18.x). Flexible extents do not have a fixed size; rather, the blocks in an extent increase as the size of the file increases. See the create_file description for more information about flex 64-bit stream files.

If you do not specify -extent_size, the converted file’s extent size is the extent size of source_file, with the following exceptions:

- For 64-bit stream files, the extent size will be 8 or greater. (A 64-bit stream file with the same extent as the source file is always large enough to hold its contents.)
- For extended-sequential files, the extent size will be 8 or greater and will be large enough to hold the contents of the source file, if possible, or 256.

-no_ask

Specifies whether the command asks before overwriting existing files with the same name. By default (yes), the command asks before overwriting existing files with the same name. This argument is meaningful only when you also specify destination.

-check

Displays a message for each file without converting it. This argument is useful for determining which files are convertible.

-no_safe_only

Attempts to convert source_file even if a failure may occur. This argument may often be successful in non-worst-case scenarios. By default (-safe_only), the command attempts to convert source_file only in cases where conversion will be successful even in worst-case situations. See the Explanation for more information.

-brief

Suppresses all successful conversion messages and messages indicating that source_file is neither stream or sequential. However, the command always displays

1 In this case, the value -1 appears in the display form. Explicitly specifying -1 on the command line has the same effect as not specifying size.
convert_stream_file

messages indicating that a stream, 64-bit stream, sequential or extended-sequential file has not been converted. If you specify a star name with the -no_brief argument, the command displays a message for each file converted; -brief suppresses these messages. By default (no), the command displays all messages.

- long
  Displays successful conversion messages even if only one file is converted. By default (no), the command does not display successful conversion message if only one file is converted.

Explanation

The convert_stream_file command converts an existing file whose organization is stream, 64-bit stream, sequential, or extended sequential to the specified file organization, optionally changing extent size. Specifically, the command performs the following types of conversions:

- Any type of stream file to another
- Sequential files to stream files, and vice versa
- Sparse 64-bit stream files to normal stream files
- Sequential files to extended sequential files, and vice versa

If you specify a star name, the command converts all stream, 64-bit stream, sequential, or extended sequential files matching the star name. If the file organization and extent size for source_file match the values specified in the -to and -extent_size arguments, no conversion occurs.

When converting between files of different extents, the size of source_file determines if it is convertible to the selected target type and extent size. The command cannot convert stream files with records longer than 32,767 to sequential files. Similarly, the command cannot convert sequential files containing records with a stream file delimiter (0x0A) to stream files. Pipe files, either stream or sequential, cannot be converted to 64-bit stream files. Files containing indexes cannot be converted to other types.

When converting between stream and sequential file organizations, the command often cannot predict with certainty whether the conversion will succeed. The uncertainty occurs because the additional overhead required for sequential files is dependent on the size of the records, which is variable. Fortunately, there is a certain capacity that always results in successful conversion regardless of record-size variability, while another capacity always results in failure.

When the command converts a file from stream to sequential, the converted file grows by three or more bytes per record. The convert_stream_file command first analyzes the source file’s current size to calculate best- and worst-case growth ratios, and then selects a target extent size that allows for the worst-case expansion. (The worst case would represent a stream file containing all null-length records.) Unless you specify -no_safe_only, the command returns an error if the converted file may not be able to hold all bytes in the worst-case expansion. If you do specify -no_safe_only, the command attempts the conversion, even though it might not succeed.
Failure occurs when the target file’s capacity is actually exceeded, resulting in the error message e$max_file_exceeded(2630). However, the command never attempts to convert a file if the target file’s capacity is insufficient for source_file’s current size, even in best-case record-size scenarios. In this case, even if you do not specify destination and a conversion failure occurs, source_file is not affected.

Access Requirements
You need modify access for any directory in which the converted file will reside (unless you specify the -check argument), and you must have at least read access to source_file.

Examples
The following command shows the results of attempting to convert files with the prefix report.

```
convert_stream_file report*
%sales#m3>Sales>Brown>report1 converted to stream64:8.
%sales#m3>Sales>Brown>report1a requires no conversion.
%sales#m3>Sales>Brown>report3 requires no conversion.
%sales#m3>Sales>Brown>reportm requires no conversion.
%sales#m3>Sales>Brown>reportml requires no conversion.
%sales#m3>Sales>Brown>reports1 converted to stream64:8.
%sales#m3>Sales>Brown>reports2 requires no conversion.
%sales#m3>Sales>Brown>reportxx requires no conversion.
%sales#m3>Sales>Brown>reportss3 converted to stream64:8.
```

The following command shows the results of attempting to convert files with the prefix report while specifying the -check argument.

```
convert_stream_file -check report* -to sequential
%sales#m3>Sales>Brown>report1 would be converted to sequential.
%sales#m3>Sales>Brown>report1a would be converted to sequential.
%sales#m3>Sales>Brown>report3 would be converted to sequential.
%sales#m3>Sales>Brown>reportm is too big to be converted to sequential.
%sales#m3>Sales>Brown>reportml would be converted to sequential.
%sales#m3>Sales>Brown>reports1 would be converted to sequential.
%sales#m3>Sales>Brown>reports2 would be converted to sequential.
%sales#m3>Sales>Brown>reportss3 may be too big to be converted to sequential.
```

In the preceding example, if the -no_safe_only argument had been specified, the last message regarding reportss3 would change to indicate that it would be converted or at least that an attempt would be made to convert it.

Related Information
See the description of the set_stream_files command in OpenVOS System Administration: Disk and Tape Administration (R284).
convert_text_file

Purpose
This command converts the contents of an existing text file to conform to the specified default character set and shift mode.

Display Form

```
file_name: [ -character_set character_set ]
[ -shift_mode shift_mode ]
```

Command Line Form

```
convert_text_file file_name
[ -character_set character_set ]
[ -shift_mode shift_mode ]
```

Arguments

- **file_name**
  The path name or star name of a file.

- **-character_set character_set**
  Assigns one of the following default character sets to the file.

  - none
  - ascii
  - latin_1
  - latin_9
  - kanji
  - katakana
  - hangul
  - simplified_chinese
  - chinese1
  - chinese2
  - user_dbcs

  By default, a value of none is assigned to the file. Specify a character set only for a fixed, relative, sequential, or extended sequential file.
**convert_text_file**

- **-shift_mode shift_mode**
  Specifies the shift combinations allowed in the file. The values for `shift_mode` are single, locking, all, and none. By default, both single- and locking-shift combinations (all) are allowed. If the value of `character_set` is none, `-shift_mode` is ignored.

**Explanation**

The `convert_text_file` command converts the contents of an existing text file to conform to the newly specified default character set and shift mode. Character sets that are supported for fixed, relative, sequential, or extended sequential files include `ascii`, `latin_1`, `latin_9`, `kanji`, `katakana`, `hangul`, `simplified_chinese`, `chinese1`, `chinese2`, and `user_dbs`. Indexes on files having one of these default character sets are allowed only if the file’s shift mode allows no shifts; indexes are not allowed for files with double-byte default character sets. If the shift mode is `all`, file data is stored as compactly as possible at the expense of I/O execution speed.

The default character set and shift mode of a file are attributes used by file and I/O services to store and present text file data in a compatible format. At times, changing needs regarding the contents of a text file or the use of a text file may necessitate altering the file’s default character set or allowed shift modes. This command performs the necessary data conversions to allow these attributes to be changed without loss of data.

This command should not be used to convert binary files to text files without specific knowledge of the contents of the original file. During conversion, binary file data will be interpreted as having a default character set of Latin alphabet No. 1 and allowing single shifts. To convert a binary file containing locking shifts or some other default character set data to a text file, first use the `set_text_file` command to set the correct default character set with `-shift_mode all` and the `-force` arguments. Then use the `convert_text_file` command to convert the file to any other desired text file format.

**Access Requirements**

To convert a text file, you need modify access to its containing directory, and write access to the file.

**Examples**

The following command converts a text file named European_report.

```bash
convert_text_file European_report -character_set latin_1 -shift_mode single
```

The default character set becomes Latin alphabet No. 1, and the shift mode allows single shifts.

If no shifts are allowed then the file can only contain one character set as specified with the character set attribute.
Related Information

See the descriptions of the `create_file` command for information on creating a text file, and the `set_text_file` command for information on setting text attributes for an existing file.
**Purpose**

This command copies a directory and its contents from one place to another.

**Display Form**

```
---------------------------------- copy_dir ----------------------------------
source_directory:             destination:
-pack:  no
-delete:  no
-files:  yes
-dirs:  yes
-links:  yes
-translate_links:  yes
-keep_dates:  no
-keep_safety_switch:  no
-brief:  no
-pacing:  disk_type
```

**Command Line Form**

```
copy_dir source_directory
[destination]
[-pack]
[-delete]
[-no_files]
[-no_dirs]
[-no_links]
[-no_translate_links]
[-keep_dates]
[-keep_safety_switch]
[-brief]
[-pacing pacing_value]
```

**Arguments**

- **source_directory**  
  Required  
  The path name or star name of the directory or directories to be copied.

- **destination**  
  The path name of the new copy of the directory. By default, the command makes `source_directory` a subdirectory of your current directory.
copy_dir

- **-pack**
  Packs any files being copied and discards deleted records. You cannot specify -pack if any file to be copied has separate-key or item indexes, or if the source directory contains any type of queue file. An error message is returned for each file that cannot be packed; those files are not copied. This argument is ignored if -no_files is also specified. By default, the command does not pack files.

- **-delete**
  Deletes a directory if its path name matches the path name of the new directory. By default, the command prompts you before deleting a directory with a conflicting path name.

- **-no_files**
  Suppresses the copying of all files in the specified directory. By default, the command copies all files in the directory.

- **-no_dirs**
  Suppresses the copying of all subdirectories in source_directory. By default, the command copies all subdirectories in source_directory.

- **-no_links**
  Suppresses the copying of all links in the specified directory. By default, the command copies all links in the directory.

- **-no_translate_links**
  Suppresses the translation of links whose targets specify objects in the directory tree subsidiary to source_directory. This argument is ignored if -no_links is also specified. By default, the command translates all links in source_directory. (See the Explanation for more information.)

- **-keep_dates**
  Assigns the creation, modification, and last-used dates of the source directory and each of its subordinate objects to the target directory and its corresponding subordinate objects. The initial last-saved date of all objects is never, indicating that the directory and its subordinate objects have never been saved. By default, the time that copy_dir was given is used for the creation, modification, and last-used dates.

  **Note:** If you have set an expiration date for files in source_directory, this command removes the expiration date from the copied files in the destination directory, even if you specify -keep_dates. For more information, see the description of the set_expiration_date command.

- **-keep_safety_switch**
  Keeps the safety switch of files being copied if the safety switch for a file is set on. By default, a file’s safety switch is off.

- **-brief**
  Suppresses the display of each directory name that matches a star name before the directory is copied. By default, the command displays the names.
copy_dir

- pacing pacing_value

Determines the pacing behavior of the copy operation. Possible values are disk_type (the default value), yes, and no. Pacing occurs during the copy operation if either of the following is true:

- If you specify disk_type and the source or target disk is optimized for fast response time

- If you specify yes

If you specify no, pacing does not occur, regardless of the type of the source or target disk. Only privileged users can specify the no value. See the Explanation section for more information about pacing.

Explanation

The copy_dir command copies a directory, including any contained files, subdirectories, links, and access control lists (ACLs). If the source directory contains a pipe file, the command creates an empty pipe file.

Specify the path name of the directory to be copied with the source_directory argument. The destination argument gives the path name of the new directory. The destination directory should not match the name of an existing directory unless you want to delete the existing directory and replace it with the copied directory. If you give the name of an existing directory for destination, the copy_dir command asks you before deleting the existing directory. All objects in a directory and in all of its subdirectories must be able to be deleted, or the directory cannot be deleted. For example, you cannot delete files that have the safety switch set, nor can you delete hidden files. (See Using OpenVOS Extended Names (R631) for information about hidden files.) If it is not possible to delete the existing destination directory, copy_dir returns an error message and does not copy any objects into that directory. However, copy_dir will delete all objects in the destination directory that can be deleted.

The copy_dir command copies the contents of directories in the following manner unless the current execution environment does not allow the name of the directory being copied or the name of the target directory.

If the directory being copied contains objects with extended names, those objects are copied only if both the module on which copy_dir is invoked and the module containing the destination directory supports them.

Depending on the module you are logged in to, different results can occur if objects with extended names are not copied:

- If you are logged in to a module that supports only legacy names, no error is returned.

- If you are logged in to a module that supports version 1 or version 2 extended names, hidden objects are ignored and are not diagnosed individually. Otherwise, an error is reported for all other objects that cannot be copied. If no error has been reported, the operating system returns the error e$all_objects_not_copied(7771).
When you copy a directory, `copy_dir` also copies its type and limit attributes. However, if the destination module is running a release that does not support expandable directories, the result is a normal directory without limits.

If the directory being copied (or any of its subdirectories) is expandable or has nonstandard limits, the type and limit information is retained only if both the module on which `copy_dir` is issued and the module containing the destination directory support expandable directories.

If you attempt to copy a directory that is expandable or has nonstandard limits to a disk that is either set with restricted expand mode or is on a module running an OpenVOS (or VOS) release that does not support expandable directories, the result is a copy into a standard directory (assuming standard capacity is sufficient), with the type and limit information removed, along with a warning message issued for the top-level directory (only) affected. For more information about restricted expand mode, see the description of the `set_dir_expand_mode` in *OpenVOS System Administration: Disk and Tape Administration* (R284).

If the directory contains more entries than will fit in a normal directory, an error occurs after the command copies all objects that will fit. Since the maximum number of entries in a normal directory is based on blocks used, the number of entries that will fit depends on various factors, including the order in which the objects are copied. Before you copy an expandable directory or a directory containing expandable directories to a module that does not support them, first convert them to normal directories to ensure that `copy_dir` does not result in an error. To identify such directories, use the `locate_expandable_dirs` command. Use `set_dir_type` or `consolidate_dir` with the `-revert` argument to convert them to normal if it is possible to do so.

If you omit the `destination` argument, `source_directory` and all its contents are built as a subdirectory of the current directory. The command uses the directory name portion of the path name specified in the `source_directory` argument as the name of the copied directory. If a subdirectory of that name already exists in the current directory, for example as the result of a previous use of the `copy_dir` command, `copy_dir` asks if you really want to delete the existing directory.

If you specify the `destination` argument, and no directory with that path name exists, the `copy_dir` command creates the directory `destination`.

If you omit the `destination` argument, and `source_directory` is an existing subdirectory in your current directory, `copy_dir` tells you that both `source_dir` and `destination` name the same object.

The value of `source_directory` and `destination` can be a star name. See the `copy_file` command for a description of how star names function.

If you specify a link for `source_directory` and specify a star name for `destination`, be careful that the link does not point to a directory that is identified by the star name, or the command may determine that `source_directory` and `destination` are identical. For example, if you have a subdirectory called `subd`, a link called `xxx` to `subd>yyy`, and two
subdirectories in \texttt{subd} called \texttt{xxx} and \texttt{yyy}, the \texttt{copy_dir} command behaves as follows when you specify \textit{source_directory} as a link and \textit{destination} as a star name.

\begin{verbatim}
  copy_dir xxx subd>*
  Copying xxx to %sys#m1>Sales>Joe_Smith>subd>xxx.
  Do you really wish to delete %sys#m1>Sales>Joe_Smith>subd>xxx? (yes, no) n
\end{verbatim}

The command does not expand the link and uses the \textbf{unexpanded} link name to resolve the star name to \texttt{subd>xxx}. The command behaves in a similar fashion if you specify \textit{source_directory} as a link and \textit{destination} as a directory name.

\begin{verbatim}
  copy_dir xxx subd
  copy_dir: Either source or destination directory is a subdirectory of the other. %sys#m1>Sales>Joe_Smith>subd.
\end{verbatim}

When you specify the \texttt{-pack} argument, all indexes are re-created regardless of file organization, though in some cases the resulting indexes are empty. It is not possible to delete a record from a fixed file with no record index. If you ask the operating system to delete a record from such a file, it updates embedded-key and deleted-record indexes appropriately, but does not actually delete the record. Therefore, such records reappear if their file is packed.

If you specify the \texttt{-delete} argument, the \texttt{copy_dir} command deletes a directory whose path name matches the path name of the new directory. When you specify the \texttt{-delete} argument and \textit{destination} is a link, the command replaces the target of the link with the copied directory.

During the copying process, links to targets in the directory being copied become links to the copy of that target in the new directory. Links to targets outside the directory being copied are copied exactly. If you specify the \texttt{-no_translate_links} argument, all links are copied exactly, regardless of their targets.

Any file with its safety switch turned \texttt{on} is protected against operations that could destroy or damage it. For additional information about the safety switch, see \texttt{set_safety_switch} later in this manual.

\textit{Pacing} prevents the copy operation from dominating the disks, and it allows other processes to access other files on the disks involved (both source and target) without long delays. Pacing is relevant only to block-mode copies; the value of the \texttt{-pacing} argument is ignored for record-mode copies (that is, those for which the \texttt{-truncate} or \texttt{-pack} argument has been specified).

The \texttt{copy_dir} command assigns ownership of any copied files to the user name of the person doing the copying.

**Access Requirements**

To copy a directory, you need status access to the directory \textit{source_directory}, status access to all its subdirectories, and read access to all the files contained in all the copied directories.
Examples

The following examples show a variety of ways that you can invoke the `copy_dir` command.

Example 1.
Suppose that this is the current directory.

```
%$1#d02>Sales>Jones
```

The following command copies the directory `%$1#d02>Sales>Clark>orders` into the current directory.

```
copy_dir >Sales>Clark>orders clarks_orders
```

The copy of the directory has the following path name.

```
%$1#d02>Sales>Jones>clarks_orders
```

Example 2.
Again, assume that this is the current directory.

```
%$1#d02>Sales>Jones
```

The following command copies the directory `%$1#d02>Sales>Clark>orders` into the current directory.

```
copy_dir >Sales>Clark>orders
```

The copy of this directory has the following path name.

```
%$1#d02>Sales>Jones>orders
```

Example 3.
Again, assume that this is the current directory.

```
%$1#d02>Sales>Jones
```

The following command copies all of the subdirectories of `>Sales>Clark>orders` into the current directory hierarchy.

```
copy_dir >Sales>Clark>orders>*
```

When given with a star name, the command displays the names of the objects being copied.

```
Copying `%$1#d02>Sales>Clark>orders>*` to *
  new_accounts
  new_orders
  closed_accounts
```

The directory `%$1#d02>Sales>Jones` now has three new subdirectories: `new_accounts`, `new_orders`, and `closed_accounts`. 
Related Information
For more information about directory management, see the command descriptions of change_current_dir, compare_dirs, consolidate_dir, create_dir, delete_dir, display_current_dir, give_default_access, move_dir, locate_expandable_dirs, remove_access, set_dir_type, and set_safety_switch.
**copy_file**

**Purpose**
This command copies the contents of a file or set of files to another file or set of files.

**Display Form**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>source_file:</td>
<td></td>
</tr>
<tr>
<td>destination:</td>
<td></td>
</tr>
<tr>
<td>-pack:</td>
<td>no</td>
</tr>
<tr>
<td>-parallel:</td>
<td>no</td>
</tr>
<tr>
<td>-truncate:</td>
<td>no</td>
</tr>
<tr>
<td>-delete:</td>
<td>no</td>
</tr>
<tr>
<td>-keep_dates:</td>
<td>no</td>
</tr>
<tr>
<td>-keep_acl:</td>
<td>no</td>
</tr>
<tr>
<td>-keep_extents:</td>
<td>yes</td>
</tr>
<tr>
<td>-brief:</td>
<td>no</td>
</tr>
<tr>
<td>-keep_safety_switch:</td>
<td>no</td>
</tr>
<tr>
<td>-keep_audit:</td>
<td>no</td>
</tr>
<tr>
<td>-pacing:</td>
<td>disk_type</td>
</tr>
<tr>
<td>-avoid_fragmentation:</td>
<td>no</td>
</tr>
</tbody>
</table>

**Command Line Form**

```
copy_file source_file [destination]
    [-pack]
    [-parallel]
    [-truncate]
    [-delete]
    [-keep_dates]
    [-keep_acl]
    [-no_keep_extents]
    [-brief]
    [-keep_safety_switch]
    [-keep_audit]
    [-pacing pacing_value]
    [-avoid_fragmentation]
```

**Arguments**

- `source_file` **Required**
  The name or star name of the file or files to be copied.
**copy_file**

- **destination**
  The path name of a file or existing directory into which the command is to copy the files. A file name can be a star name; if the character strings that replace the asterisks cause a destination file name to exceed the maximum length of a file name, the leftmost characters beyond that limit are truncated. If you specify a directory, the command copies the files into the directory. If you specify a file name, the command copies the file into the current directory and names the copy **destination**. By default, the command copies the files into the current directory without changing their names, as long as the current directory is not the source directory.

- **-pack**
  Packs a file being copied and discards deleted records. You cannot specify **-pack** if the file to be copied has separate-key or item indexes, or if the file is a queue or pipe file. By default, the command does not pack the file.

- **-parallel**
  Used with the **-pack** argument; when both arguments are set to yes, the command simultaneously rebuilds all indexes for each file being copied. By default, the command has no effect.

- **-truncate**
  Truncates an existing **destination** file before copying an input file to it. If you specify **-truncate** and the file named by **destination** is an existing file, then the command preserves the file’s attributes and indexes but deletes its contents (that is, its data records). Any embedded key indexes previously defined on the destination are rebuilt from the new records. However, if the file named by **destination** does not exist, the command creates an output file with the same organization, maximum record length, and allocation size as the file to be copied but creates no indexes on the new file. If you specify **-truncate**, the file is packed regardless of the value of the **-pack** argument. You cannot specify **-truncate** if the file to be copied has separate-key or item indexes, or if the file is a queue or pipe file. By default, **-delete** determines what happens if there is a name conflict.

- **-delete**
  Deletes a file if it has the same path name as the destination path name of the copied file. By default, the command asks you whether to delete a file that has a conflicting path name.

- **-keep_dates**
  Assigns to the new file the creation date, modification date, and last-used date of the file being copied. The initial last-saved date of all objects is never. By default, the current time of the copy is used for the creation date, modification date, and last-used date.

  **Note:** If you have set an expiration date for **source_file**, this command removes the expiration date from the file specified by **destination**, even if you specify **-keep_dates**. For more information, see the description of the **set_expiration_date** command.
-keep_acl

Keeps the access control list with the file. If you specify -keep_acl, the default access of the directory is not saved; only the specific access control list on the file (if one exists) is saved. By default, access is not moved with the file.

-no_keep_extents

Specifies that the command does not copy the source file’s extent characteristics to the target file. By default, the target file has the same extent characteristics as the source file, which means that if the source file is extents-based, the target file will also be extent-based with the same extent size and, in the case of statically-allocated extents (SAE) files, with the same number of blocks initially allocated. This argument is meaningless for non-extents files. See the Explanation section for more information about how extents are copied.

Specify this argument with the -truncate argument to migrate non-extents-based files to dynamically-allocated extents (DAE) files (that is, files that allocate and initialize extents as the file grows). See the Explanation section for more information about DAE files.

-brief

Suppresses the display of each file name that matches a star name before the file is copied. By default, the command displays the name.

-keep_safety_switch

Keeps the safety switch of files being copied if the safety switch for a file is set on. By default, a file’s safety switch is off.

-keep_audit

Specifies that the new file retains the audit options of the source file. By default, the command does not retain the audit options of the source file.

-pacing pacing_value

Determines the pacing behavior of the copy operation. Possible values are disk_type (the default value), yes, and no. Pacing occurs during the copy operation if either of the following is true:

- If you specify disk_type and the source or target disk is optimized for fast response time
- If you specify yes

If you specify no, pacing does not occur, regardless of the type of the source or target disk. Only privileged users can specify the no value. See the Explanation section for more information about pacing.

-avoid_fragmentation

Allocates the target file in nonfragmented portions of the disk, if possible. This means that if \( N \) contiguous blocks are available on the disk, they are used first, then \( N/2 \), and so on, where \( N \) starts at 256. By default, the target of copy_file is not allocated in such a way as to avoid fragmentation.
copy_file

The command uses noncontiguous blocks as long as disk space is available. Therefore, the success of producing a **defragmented** file (that is, a file with few gaps between blocks) depends on the allocation patterns currently existing on the disk. (Note that you can use software tools to analyze allocation patterns.) This affects striping on multi-disk volumes, in that \( N \) blocks are allocated in each member. The resulting file has the same attributes as source_file. This argument affects only non-extent files, since extent files are always allocated based on their extents. Also, this argument has no affect if you specify -truncate and the target file exists. In that case, allocation is dictated by the attributes of the existing target file.

**Explanation**

The **copy_file** command copies the contents of a file or set of files to another file or set of files. If the source is a pipe file, the command creates an empty pipe file.

The **copy_file** command assigns ownership of any copied files to the user name of the person doing the copying.

**Note:** The **display_file_status** command shows index names in order of the index address inside the file. This order may change if the file is specified as the subject of the **copy_file**, move_file, restore_object, or save_object command.

**Specifying the Source File and Destination**

You can give a star name for source_file and destination. Star names function in the following manner.

- An asterisk can be in any position in a star name.
- In a path name, a star name can be in the final name position only.
- When the operating system matches non-star names to a star name, each asterisk represents zero or more characters.
- A name cannot contain consecutive asterisks; there must always be an intervening character.

The command also allows the destination to match the name of the source.

If you omit the destination argument, the **copy_file** command copies all of the files whose names match source_file into your current directory, using the same file names. If the files to be copied already reside in your current directory, the operating system displays this message.

> Both the source and destination name the same object.

If you give a directory name for the destination argument, you can also give a star name for source_file that matches more than one file name. The **copy_file** command copies all the files whose names match the source_file argument into the specified directory using the same file names.

If you give a path name for destination that is not the path name of an existing directory, the **copy_file** command assumes that destination is a file name. If this file name is not
a star name, then `source_file` must match only one file path name. In this case, the `copy_file` command copies the file and names the copy as specified in `destination`. Unless you specify this last argument, the copy retains its name.

If you specify a link for `source_file` and specify a star name for `destination`, be careful that the link does not point to a file that is identified by the star name, or the command may determine that `source_file` and `destination` are identical. For example, if you have a subdirectory called `subd`, a link called `xxx` to `subd>yyy`, and two files in `subd` called `xxx` and `yyy`, the `copy_file` command behaves as follows when you specify `source_file` as a link and `destination` as a star name.

```
copy_file xxx subd>*
Copy file \%sys#m1>Sales>Joe_Smith>xxx to
   \%sys#m1>Sales>Joe_Smith>subd>xxx.
\%sys#m1>Sales>Joe_Smith>subd>xxx already exists.
   Delete the old one? (yes, no) n
```

The command does not expand the link and uses the unexpanded link name to resolve the star name to `subd>xxx`. The command behaves in a similar fashion if you specify `source_file` as a link and `destination` as a directory name.

```
copy_file xxx subd
%sys#m1>Sales>Joe_Smith>subd>xxx already exists.
   Delete the old one? (yes, no) n
```

**Specifying the `-pack` and `-parallel` Arguments**

When you specify the `-pack` argument, all indexes are re-created, regardless of file organization; however, in some cases, the resulting indexes are empty. If an index that is re-created is large, temporary files are required for sorting. The temporary files are created in the `process_dir` directory (if the user has access) or the current working directory.

If you specify the `-pack` argument, `copy_file` attempts to discard deleted records. However, you cannot delete a record from a fixed file that has no record index. If you ask the operating system to delete a record from such a file, it updates embedded-key indexes and deleted-record indexes appropriately, but does not actually delete the record. Therefore, such records can reappear if their file is packed. To prevent this, `copy_file` ignores records that consist entirely of hexadecimal FF when packing a fixed file without a record index.

The `-parallel` argument is used with the `-pack` argument; when both arguments are set to `yes`, the command simultaneously rebuilds all indexes for each file being copied, enabling the specified files to be packed and copied faster. The default value for the `-parallel` argument is `no`, in which case, the command rebuilds the indexes individually. The `-parallel` argument has no effect if the `-pack` argument is set to `no`.

You cannot copy a transaction-protected file if you specify the `-pack` argument. See `s$copy_file` in the OpenVOS Subroutines manuals for more information.

**Specifying the `-truncate` Argument**

If you specify the `-truncate` argument and the destination file is an existing fixed file, all records in the source file must have a length equal to the fixed record size of the destination file.
Warning messages are displayed if indexes are not copied or rebuilt. If you specify -trunc and the target file exists, copy_file checks the indexes and displays a message for each embedded index that is not on the target file and for each non-embedded index. If you specify -trunc and the target file does not exist, copy_file displays a message for each index that is on the source file. If you specify -no_trunc and -pack, copy_file displays a message for each embedded-separate-key index that is on the source file.

You cannot copy a transaction-protected file if you specify the -trunc argument. See $copy_file in the OpenVOS Subroutines manuals for more information.

Copying Files with Extents
By default, when you copy a file, the extent allocation is copied.

Statically-allocated extents (SAE) files cannot be created across the network and therefore cannot be copied, unless the target file exists and you specify the -trunc argument. If you want to copy such a file across the network and the target file does not exist, you must specify the -no_keep_extents argument. In this case, the newly created file is a non-extent file. You cannot copy an SAE extended-sequential file across the network in this way, because this type of file is always extent-based. To duplicate an SAE file on another module, create an identical empty file with the create_file or clone_file command while running on that module, and specify the -trunc argument of copy_file.

If you specify -no_keep_extents for a dynamically-allocated extents (DAE) file, the newly created file is a non-extent file, with two exceptions:

- If you specify -no_keep_extents for an extended-sequential file, the extent value is reduced to the maximum record size specified when the file was created, or 8 if that is larger.

- If you specify -no_keep_extents for a 64-bit stream file and the size of the file requires extents, the value is the minimum needed to hold the current contents of the source file.

- If you specify -no_keep_extents for a flex 64-bit stream file and copy that file to a module running a release prior to OpenVOS Release 18.x, and if a non-extent file is too small to hold the contents of the source file, the new file is created with large enough extents to hold the contents.

If you copy a sparse 64-bit stream file that has grown and that may contain blocks containing all binary zeros, copy_file may eliminate those blocks in destination, which means that the resulting file may become smaller.

You can copy 64-bit stream files that have not grown larger than 2 GB to modules running releases prior to OpenVOS Release 17.2.x. These files become normal stream files on the module running the earlier release. You can copy the files regardless of whether your current module is running the earlier release or Release 17.2.x or later.

If your current module is running an earlier release and you copy a 64-bit stream file to a module running Release 17.2.x, the file becomes a normal stream file, even on the module running Release 17.2.x.
If you copy a flex 64-bit stream file to a module running OpenVOS Release 17.2.x, the destination file is a normal (that is, non-flex) 64-bit stream file with no extents as determined by the module default (normally, 8).

A 64-bit stream file that is 2 GB or greater cannot be copied to a release prior to OpenVOS Release 17.2.x; any attempts to do so results in an error message. This is also true if you execute the copy_file command running on an older release. However, you can copy sparse files that are less than 2 GB, except that the resulting normal stream file will not be sparse and therefore may require far more disk space (and more time to copy).

To migrate a non-extent-based file to a DAE file, create an empty DAE file, and then specify the copy_file command with the destination and -truncate arguments (destination should be the empty DAE file). This procedure copies the data from source_file into destination while keeping the extent attributes of destination intact. For more information about DAE files, see the description of the create_file command.

Using the Safety Switch
When a file’s safety switch is on, the file is protected against operations that could destroy or damage it. For additional information about the safety switch, see set_safety_switch later in this manual.

Copying RAM Files
If a non-empty RAM file is copied, the newly created file does not have RAM file usage. This allows you to copy a RAM file while it is activated in order to capture its contents; when a RAM file is deactivated, its contents are discarded. If the target of copy_file is a RAM file, it would be truncated immediately after being copied. Specifying an existing RAM file as the target and using the -truncate or -pack argument is not allowed, since the result of the copy would always be an empty file.

When a server queue has RAM usage, the new server queue retains RAM usage and is thus always empty. The contents of a server queue are never copied.

If a RAM file’s containing directory is being copied or moved, the RAM file in the newly created directory retains its RAM usage and is always empty. See the description of set_ram_file for more information about RAM files.

Using the -pacing Argument
Pacing prevents the copy operation from dominating the disks, and it allows other processes to access other files on the disks involved (both source and target) without long delays. Pacing is relevant only to block-mode copies; the value of the -pacing argument is ignored for record-mode copies (that is, those for which the -truncate or -pack argument has been specified).

Access Requirements
You must have read access to a file that you want to copy, status access to the source directory, and modify and, by default, write access to the directory that is to contain the copied file. You will be given write access to the destination.
Examples
The following three examples show a variety of ways that you can invoke the `copy_file` command.

Example 1.
To copy all OpenVOS COBOL source modules in the current directory into the directory `>east>Clark`, use this command.

```plaintext
copy_file *.cobol >east>Clark
```

The object names of the copies are the same as the original files.

Example 2.
The following command copies all OpenVOS COBOL source modules in the current directory into the directory `>east>Clark>save` and changes the suffix `.cobol` to the suffix `.old` for each file copied.

```plaintext
copy_file *.cobol >east>Clark>save>*.old
```

Example 3.
This command makes a copy of the file `make_reports.pm` in the current directory, names the copy `make_reports.old.pm`, and puts the copy in the current directory.

```plaintext
copy_file make_reports.pm make_reports.old.pm
```

When you give the copy a different name from the original (as in this example) you can copy only one file; that is, `destination` must be the path name of a file.

Related Information
See also the command descriptions of `compare_files`, `copy_dir`, `create_file`, `delete_file`, `display_file_status`, `locate_files`, `move_dir`, `move_file`, `set_file_allocation`, `set_ram_file`, `set_safety_switch`, and `truncate_file`. 
**copy_tape**

**Purpose**
This command makes logical copies of all tape formats.

**Display Form**
```
---------------------------------- copy_tape ----------------------------------
source_tape_device_or_port:     copy_tape source_tape_device_or_port
destination_tape_devices_or_port: destination_tape_devices_or_port
-create_destination_volume:    no
-simultaneous_rewind:          no
```

**Command Line Form**
```
copy_tape source_tape_device_or_port
destination_tape_devices_or_port
[-create_destination_volume]
[-simultaneous_rewind]
```

**Arguments**
- **source_tape_device_or_port** Required
  The name of the tape device, or the name of the port attached to the tape drive, holding the tape to be copied. You can attach the port or mount the tape before using the `copy_tape` command. If `copy_tape` implicitly attaches the port or implicitly mounts the tape, `copy_tape` also implicitly dismounts the tape or implicitly detaches the port.

- **destination_tape_devices_or_port** Required
  The names of ports attached to tape drives, or the names of tape drives that will contain the copies. You can attach the ports and mount some or all of the tapes before using the `copy_tape` command.

After `copy_tape` is executed, any destination tapes that were already mounted remain mounted, and any destination ports that were already attached remain attached.

If a port is not attached, `copy_tape` implicitly attaches a port. If `copy_tape` implicitly attaches a port, it implicitly detaches the port after execution.

If a tape is not mounted and `create_destination_volume` is set to `no` (the default), `copy_tape` attempts to mount the tape implicitly, using the tape’s volume label. If an error occurs during this process, or if the tape does not have a volume label, the command is aborted. If, however, the command succeeds in mounting the tape implicitly using the volume label, the command is executed, and overwrites any data...
currently contained on the destination tape. If \textit{copy\_tape} implicitly mounted the tape, it implicitly dismounts the tape after execution.

If a tape is not mounted and \textit{-create\_destination\_volume} is set to \textit{yes}, \textit{copy\_tape} automatically mounts the tape, issuing the \textit{mount\_tape} prompts for creating and mounting a tape. The \textit{copy\_tape} command prompts you to mount the destination tapes in the order that you specified the ports or devices in the \textit{destination\_tape\_devices\_or\_port} argument. You must respond to each of these prompts. If \textit{copy\_tape} implicitly attached a port for this tape before it automatically mounted the tape, \textit{copy\_tape} implicitly detaches the port and unloads the tape after execution. Otherwise, if the port was already attached, the command dismounts the tape after execution, and the port remains attached.

**\textit{-create\_destination\_volume}**

Specifies whether \textit{copy\_tape} is to create the volumes on destination tapes. This argument affects only destination tapes, not the source tape. The default setting is \textit{no}. The operating system ignores the setting of this field for any destination tapes that you mounted with the \textit{mount\_tape} command before issuing the \textit{copy\_tape} command. Otherwise, if you cycle to \textit{yes}, any destination tapes that were not yet mounted will have volumes created on them.

For information about the state of the tape after \textit{copy\_tape} is executed, see the \textit{destination\_tape\_devices\_or\_port} argument.

**\textit{-simultaneous\_rewind}**

Makes many copies simultaneously. If you set this argument to \textit{yes}, you must switch reels or cartridges on all the destination tapes whenever any one reaches the end-of-tape mark. The command prompts you to switch reels or cartridges. This setting saves time because all the drives wind together. You waste a small amount of tape on the end of the reels and cartridges that are forced to switch before they reach their end-of-tape marks. When tape drives rewind at different times, the copying process stops on all tapes until you load a new tape.

You cannot use \textit{-simultaneous\_rewind} if all the destination tape drives do not use the same density. If you attempt to use the argument when the densities are different, the command displays an error message. By default, \textit{copy\_tape} does not make destination tapes rewind simultaneously.

**Note:** You can mix the destination tape devices between cartridge and reel tapes, but it is recommended that you not do this if you use \textit{-simultaneous\_rewind}. Depending on the type of reel and cartridge tape drives you use, much tape could be left unused.

**Explanation**

The \textit{copy\_tape} command makes logical copies of all tape formats, including ANSI; IBM; UNIX \textit{tar}, \textit{cpio}, and \textit{cpioc}; and unlabeled formats. Use this command to create multiple copies of an entire tape volume or multivolume set. The copies are \textit{logical}, meaning that the file IDs and data are identical on each copy, but the volume IDs are different and the physical mapping of data can vary between tapes.
If you want to copy an entire tape volume or multivolume set without having to intervene, you must do the following:

1. Give the `create_tape_volumes` command to establish the destination volumes.
2. Give the `mount_tape` command with the `-unattended` argument to mount the tapes.
3. Give the `copy_tape` command.

**Note:** You must give this combination of commands in the order listed above, since you cannot explicitly invoke `copy_tape` in unattended mode.

For more information about tape mounting, see Explanation in the `mount_tape` command description.

**Access Requirements**

By default, you have write access to all devices. If your system administrator restricts access to a device, you need read access to read from tapes, or write access to read from and write to tapes.

**Related Information**

See also the command descriptions of `create_tape_volumes`, `dismount_tape`, `display_tape_params`, `dump_tape`, `list_save_tape`, `mount_tape`, `position_tape`, `read_tape`, `restore_object`, `save_object`, `set_second_tape`, `set_tape_drive_params`, `set_tape_mount_params`, `set_tape_file_params`, `verify_save`, and `write_tape`. For information about the `save` and `restore` commands, see *OpenVOS System Administration: Backing Up and Restoring Data* (R285).
Purpose

This command produces a fully expanded OpenVOS Standard C source module.

**Note:** The OpenVOS GNU Tools layered product also supplies a command named `cpp`. When the GNU Tools product is installed and used as directed, the GNU Tools version of this command is found before the OpenVOS version, depending upon command library paths. Since the OpenVOS and GNU Tools commands behave differently, you can use the alternate name (`vcpp`) to ensure that you invoke the OpenVOS version of the command.

Display Form

```
source_file_name: [option_menu: cycle for available options...]
option_selection:
- suppress_diag:
- include:
- processor: default
- check_incompatible: no
- system_programming: no
- list: no
- statistics: no
- show_include: local
- show_macros: unexpanded
- compress: no
```

Command Line Form

```
cpp source_file_name
[ option_selection...]
[ -suppress_diag number...]
[ -include include_file_name...]
[ -processor processor_string]
[ -check_incompatible]
[ -extension_checking string]
[ -system_programming]
[ -default_char string]
[ -list]
[ -mapcase]
[ -statistics]
[ -nesting]
[ -show_include string]
[ -show_macros string]
[ -compress]
```
Arguments

The arguments are a subset of those to the cc command. Refer to the description of the cc command for argument definitions.

Explanation

The cpp command preprocesses an OpenVOS Standard C source module and produces a fully expanded source module.

The arguments to this command are a subset of those to the cc command and produce results that are nearly identical to those produced by the corresponding arguments of the cc command. For the short options, you can use only those options that are applicable to preprocessing.

When you invoke cc with the -E option, the cpp command duplicates the functionality of the cc command, except that cpp makes no assumptions about the suffix of either the input or output file. On input, cpp does not assume that the input file has the suffix .c. On output, if you use the -o option to name the output file, cpp does not append .ex.c to the name you specify.

For example, the command cc -E my_file preprocesses the file my_file.c and creates the output file my_file.ex.c. On the other hand, the command cpp my_file produces a “file not found” error. Therefore, you would have to specify the command as cpp my_file.c.

See the description of the cc command for more information.
create_data_object

Purpose
This command creates an object module of a specified size to be used by processes that share
virtual memory.

Display Form

Command Line Form

create_data_object object_name
data_length

Arguments

- **object_name** Required
  A data object module to be bound into program modules that use the subroutine
  $\text{connect\_vm\_region3}$ to access a file. The object name cannot be an extended
  name.

- **data_length** Required
  The size, in bytes, of the data object module.

Explanation
The create_data_object command supports the sharing of virtual memory by different
programs. It creates an object module, that, when bound, has the specified name and size. By
binding this object module at the same virtual address in each program, and by passing this
address to $\text{connect\_vm\_region3}$ in each process, the region of virtual memory can be
shared among all of the executing programs.

**Note:** The size of the object module file itself bears no relation to the size of the data
object when bound; the former is always 1 block long while the latter can be arbitrarily
large.

Because entire 4096-byte pages are shared, you will generally want to define the
data_length in multiples of 4096 bytes and use the page_aligned directive in the
corresponding binder control files to start the data object on a page boundary.
See the description of the `s$connect_vm_region3` subroutine in the OpenVOS Subroutines manuals for details.

**Examples**

The following sequence of commands creates two files, `region1.obj` and `region2.obj` that define 8192 and 4096 bytes of space, respectively.

```
create_data_object region1 8192
create_data_object region2 4096
```

You can bind the resulting object modules by including them in a binder control file of the form.

```
name:   example;

modules: region1 page_aligned,
         region2 page_aligned,
         example,       /* other user code */
         .
         .
         .
end;
```

Note that the object created by `create_data_object` contains a variable of the same name as the object name. You must give the external variable in your program the same name as the data object in order for your program to reference it.

An OpenVOS PL/I user program might reference the data objects by declaring them as follows:

```
dcl region1 char (8192) external static;
dcl region2 char (4096) external static;
```

**Related Information**

For information about the `modules` and `high_water_mark` bind control directives, see the description of the `bind` command and Appendix C. See also the description of the `s$connect_vm_region3` subroutine in the OpenVOS Subroutines manuals.
create_deleted_record_index

Purpose
This command creates a list of records deleted from a file as potential locations for new records.

Display Form

```
--------------- create_deleted_record_index ---------------
path_name:      [ REQUIRED ]
-unpack:        no
-detail:        no
```  

Command Line Form
```
create_deleted_record_index path_name
   [-unpack]
      [-detail]
```

Arguments

- **path_name**  
  The path name of the file for which an index of deleted records is to be maintained.

- **-unpack**  
  Prevents adjacent deleted records from being merged into a single record before they are added to the deleted record index. This argument is especially important when you are building a deleted record index for an existing sequential file with data. By default, the command causes adjoining deleted records to be combined into a larger record entry in the index. Note that you can use the -detail argument in conjunction with -unpack for use in debugging the deleted records in the sequential file.

- **-detail**  
  Provides information you can use to debug the index of deleted records. Note that you can use the -detail argument in conjunction with the -unpack argument to debug the deleted records in the sequential file.
create_deleted_record_index

Explanation

The create_deleted_record_index command creates a list of all reusable locations in a particular file. (A location is reusable when a record is deleted from that location.) A deleted record index enables more efficient file space use, since the space taken up in the file by deleted records is reused for new records.

**Note:** You should not create deleted record indexes for fixed files. If you attempt to do so by issuing the create_deleted_record_index command, the command returns the error e$ve_no_sequential_dri (7331). To avoid this error, issue the create_record_index command for fixed files.

When you create a deleted record index for a file, the file system reads through the file and creates deleted record entries for all records currently deleted from the file. This index stores the size and location of deleted records in the file. The next time the file system writes a new record to the file, it writes it in place of a deleted record of the same size or smaller, instead of at the end of the file. The file system also reuses the deleted record’s record number (in contrast to what happens with a record index created by the create_record_index command). In addition, the file system tracks subsequent deletions from the file, adding new deleted record entries to the list of reusable locations in the file.

Use the create_deleted_record_index command when you need to save file space and do not need to keep track of record numbers; for example, when you use a key to access an indexed file.

In order for the file to include separate entries for adjacent deleted records from a sequential file, invoke the command with the -unpack argument. By default, adjacent deleted records are merged into a single, larger entry in the index. In this situation, record space may not be used as efficiently as possible in applications using sequential files with fixed length records. However, use of the -unpack argument will ensure that record space is not wasted.

When you specify the -unpack argument, the create_deleted_record_index command opens the file for DIRTY_INPUT_TYPE, thus allowing other processes to have the file open in other opening modes. Any deleted records found during the normal file scan are entered into the index. A record that is deleted after the command processes that part of the file is not entered into the index. For this reason, you should not “purge” the file of old data while this command is executing. After the index is created, the create_deleted_record_index command closes the file and tries to reopen it for exclusive input. If this attempt fails, the command displays the following message:

```
create_deleted_record_index: file_name. path_name
```

In order for the _deleted_record_index to be rebuilt, you must shutdown the application(s) that are using this file. If you answer stop the temporary files will be deleted and the command will have to be re-issued. Is the application shutdown? (yes, no, stop)

If you see this message, shut down the application or applications that are using this file, and then type yes. After the command converts the deleted record index and threads it into the file entry in the directory, the command terminates. This step is generally much faster than
create_deleted_record_index

the sequential reading of the entire file (seconds versus hours). You can then restart the application processes.

Example

Consider a relative file named %s1#d02>Sales>Jones>discounts, with 10 records. Create a deleted record index by issuing this command.

    create_deleted_record_index %s1#d02>Sales>Jones>discounts

Suppose you subsequently delete record number 2 from discounts. The next record written to the file could be written at position 2, reusing that space; the new record would also become record number 2.

Related Information

See the description of the create_record_index command for information about another way to save file space.
create_dir

Purpose
This command creates one or more directories.

Display Form

```
create_dir directory_name...
[ -parents]
```

Arguments

- `directory_name` **Required**
  One or more path names of new directories.

- `parents` **CYCLE**
  Creates any missing intermediate directories. By default (no), the command does not create any intermediate directories.

Explanation
The `create_dir` command creates one or more directories. The `directory_name` argument is a path name or set of path names of a new directory or directories.

If a path name for a new directory conflicts with that of an existing object, the `create_dir` command does not create the directory unless the existing object is a link whose target does not exist. In this case, the `create_dir` command creates the directory and locates it so that it becomes the target of the link.

The access control list of the new directory is a copy of the access control list of its containing directory. The default access control list for files in the new directory is a copy of the default access control list for files in its containing directory.

Access Requirements
To create a directory, you need modify access to the containing directory of the new directory.
create_dir

Examples

Example 1.
To create a directory named reports in your current directory, use this command.

    create_dir reports

Example 2.
The following command creates a directory named new_customers in the person directory Jones in the group directory Sales.

    create_dir %s1#d02>Sales>Jones>new_customers

Example 3.
The following command creates a directory named reports, as well as the intermediate directories sales and q2.

    create_dir -parents sales>q2>reports

Related Information
See also the command descriptions of change_current_dir, compare_dirs, copy_dir, delete_dir, display_current_dir, and move_dir.
**create_file**

**Purpose**

This command creates and names an empty file.

**Display Form**

```
create_file
```

```
file_name:    
-organization: sequential
-record_size:    
-num_records:    
-extent_size:    1
-dynamic_extents: no
-character_set: none
-shift_mode: all
```

**Command Line Form**

```
create_file file_name

  [-organization organization]
  [-record_size record_size]
  [-num_records number]
  [-extent_size [size]]
  [-dynamic_extents]
  [-character_set character_set]
  [-shift_mode shift_mode]
```

**Arguments**

- **file_name**  
  Required  
  The path name of the new file. The file name must conform to the rules for valid file names on the module where the file is being created.
create_file

-organization organization
   Specifies one of the following types of file organization for the new file.
   - sequential
   - stream
   - relative
   - fixed
   - server_queue
   - message_queue
   - one_way_server_queue
   - ext_sequential
   - stream64

By default, the command creates a sequential file.

-record_size record_size
   Specifies the maximum record size, in bytes. If you specify fixed, relative, or ext_sequential for -organization, record_size is required. If you specify stream64 for -organization, record_size is optional. The command ignores the value if you specify sequential, stream, or one of the queue types for -organization. See the Explanation section for more information.

-num_records number
   Specifies the number of records to be pre-allocated in the created file. For a fixed or relative file, the operating system uses number and record_size to determine the number of blocks to allocate to hold these records. For other file types, the operating system allocates number blocks. If the file is extent-based, this value is the number of records or blocks stored in the extents. See the Explanation section for more information on how to calculate the maximum number of records in extent-based files. This argument is optional for DAE files and required for SAE files.

-extent_size [size]
   Specifies the size, in blocks, of the extents used by the file. A file extent is a contiguous group of blocks represented by a single disk address. The use of extents allows a file to grow beyond the capacity of the file map, allowing one map entry to represent multiple blocks, and causes allocation of the blocks in each extent to be contiguous, often resulting in better performance. Queue files cannot be extent-based. The number of extents in a file and the maximum size in blocks to which a file may grow depends on the file organization, extent size, record size, and number of records. For information on calculating the maximum number of file blocks, see the Explanation section.

   If size is a value other than 1, create_file uses the values specified for -extent_size and -dynamic_extents to create the file.

   Extents may be dynamically allocated (DAE) or statically allocated (SAE), depending on the value of the -dynamic_extents argument. The allowable values of size, if it is specified, depend on the value of -dynamic_extents. A size value of 1 indicates a non-extent file (that is, there is one block per file map entry).
For statically-allocated extents, you must specify the -num_records argument to indicate the initial size of the file. Only this initial, pre-allocated portion of the file uses the specified extent size. Further file growth is non-extent.

**Note:** If size is greater than the actual number of blocks needed to store the specified number of records, only one extent of size blocks is actually allocated. For example, if you create a relative file with a record size of 4094 bytes (one block) and 100 records, requiring 100 blocks of storage, but specify an extent size of 128 blocks, the create_file command allocates one extent of 128 blocks even though only 100 blocks is needed.

For dynamically allocated extents, the -num_records argument is optional, indicating the portion of the file to be pre-allocated. The specified extent size applies to all subsequent file growth. If you specify size, it must be either 1 or a value from 8 to 256 that is a power of 2. If you do not specify size, the operating system sets the extent size to 64, except for 64-bit stream files. In that case, variable flexible extents are used. The value of -record_size for both extended sequential and 64-bit stream files can affect size.

If you specify -extent_size with no value

- For sequential and stream files, the extent size is 1 (that is, no extents).
- For extended-sequential files, the extent size is 8.
- For 64-bit stream files, the extent size is the target module’s default value: normally, flex (or 8 for modules running a release earlier than OpenVOS Release 18.x). Flexible extents do not have a fixed size; rather, the blocks in an extent increase as the size of the file increases.
- For DAE files that are not 64-bit stream files, the extent size is 64.
- For SAE files, the extent size is 128.

If you do not specify -extent_size, the file’s extent size is the extent size of source_file, except that for extended-sequential files, the extent size must be 8 or greater and must be large enough to hold the contents of the source file, if possible, or 256.

---

**-dynamic_extents**

Creates a dynamically-allocated extents (DAE) file. By default, the create_file command does not create DAE files.

DAE files offer the following features.

- No pre-initialization of the entire file, which is costly with very large files
- Maximum extent size of 256
- Dynamic file growth by extent size

---

2 In this case, the value -1 appears in the display form. Explicitly specifying -1 on the command line has the same effect as not specifying size.
• A transparent network model: you can create DAE files across networks

DAE files grow by extent size units. Extent sizes are a power of 2 between 8 and 256. DAE files can support fixed, relative, extended sequential files larger than 2 GB, up to 512 GB, depending on file characteristics. With DAE, you can specify zero-length files at file creation, thus allowing for the file to grow dynamically over time.

The extent size is given at file creation, but an extent is not initialized until a block within the extent is first written. When you access a given block in a DAE file, the corresponding extent is examined. If the disk address has not been assigned, it is calculated on the extent-size boundary, and all blocks in the extent are allocated.

See the Explanation for more information about creating DAE files.

► -character_set character_set
   Specifies one of the following default character sets to be assigned to the file.
   - none
   - ascii
   - latin_1
   - latin_9
   - kanji
   - katakana
   - hangul
   - simplified_chinese
   - chinese1
   - chinese2
   - user_dbcs

By default, a value of none is assigned to the file. Specify a character set only for a fixed, relative, or sequential file.

► -shift_mode shift_mode
   Specifies the shift combinations allowed in the new file. Possible values for shift_mode are all, none, locking, and single. By default, both single- and locking-shift combinations (all) are allowed. The -shift_mode argument is ignored if the value of character_set is none. If the shift mode is locking or all, file data is stored as compactly as possible at the expense of execution speed. For more information on shift modes, see the National Language Support User’s Guide (R212).

Explanation

The create_file command does not create the new file if the path name of a new file conflicts with that of an existing object. There is one exception to this: if the existing object is a link whose target does not exist, the create_file command creates the file and locates it so that it becomes the target of the link.

File Organization Characteristics

This section describes the main characteristics of the file organizations supported by the create_file command. A file is a named, logical unit of storage containing a sequence of records. You can use the create_file command to create the following types of files:
- sequential files
- extended sequential files
- stream files
- 64-bit stream files
- flex files (a type of 64-bit stream file)
- relative files
- fixed files
- queue files

Note: You can create pipe files with the set_pipe_file command. You can create paging files with this command and the add_paging_file command. For a description of the add_paging_file command, see the manual OpenVOS System Administration: Disk and Tape Administration (R284).

Sequential Files
A sequential file contains records that can vary in length from 0 to 32,767 bytes. The disk space allocated for each record also varies. As shown in the following illustration, records start and end with two bytes (one word) that contain the length of the record.

<table>
<thead>
<tr>
<th></th>
<th>This is a 16-byte long record.</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>16</td>
<td></td>
<td>16</td>
</tr>
<tr>
<td>34</td>
<td>This is a 34-byte long record.</td>
<td>34</td>
</tr>
</tbody>
</table>

When you delete a record in a sequential file, OpenVOS does not delete the data in the record. Instead, it changes the record length to a negative number to indicate that the record has been deleted. The space from logically deleted records can be regained by using the copy_file command and specifying the -pack or -truncate argument.

The value sequential is the default value of the create_file command’s -organization argument.

Record size is not meaningful for sequential files. The value of the -record_size argument must not be less than or equal to zero; all positive values are ignored.

Extended Sequential Files
An extended sequential file is a sequential SAE or DAE file defined in such a way that it can grow larger than $2^{31}$-1 bytes. The records within an extended sequential file are formatted like those in a sequential file.

The difference between a sequential file and an extended sequential file is that when you use the latter file organization, records always begin on a byte boundary corresponding to the value of -record_size. A record size of 8 indicates that each record begins on a 0 mod 8 byte boundary. A record size of 64 indicates that each record begins on a 0 mod 64 byte boundary.

The -organization and -record_size arguments control extended sequential file creation as follows:
The value of `-record_size` indicates the boundary granularity and must be a power of two between 2 and 256, inclusive.

The command creates a DAE or SAE extended sequential file, based on the value of `-extent_size`. If the value is 1, `create_file` creates a DAE extended sequential file. In this case, the extent size corresponds to the value of `-record_size`, except that the values 2 and 4 result in an extent size of 8, the minimum DAE extent size.

You cannot use `create_file` to create an extended sequential file without extents. Instead, you must use the `s$create_extent_file` subroutine, as documented in the OpenVOS Subroutines manuals. The `s$create_extent_file` subroutine description also provides information about accessing extended sequential files across modules.

Note: Where this manual refers to sequential files, it also refers to extended sequential files, unless otherwise noted. The two are identical except that the maximum record size, which is meaningless for normal sequential files, indicates record offset unit size for extended sequential files, thereby allowing a greater growth potential.

A text file cannot have the extended sequential file format.

Stream Files

A stream file contains either binary data or records that can vary in length from 0 to 32,767 bytes. The disk space allocated for each record also varies. As shown in the following illustration, the end of each record is marked by an ASCII line-feed character (0A hexadecimal). If you insert a line-feed character in a record, the record becomes two records.

This is a record. ‘0A This is another, longer record. ‘0A

‘0A = line-feed character

You cannot physically delete a record in a stream file; however, you can truncate and zero extend stream files containing binary data.

The `-record_size` argument is ignored if you specify the `-organization stream` argument.

64-bit Stream Files

A 64-bit stream file is a stream file that has certain capabilities that a normal stream file does not have, along with certain limitations. It can grow larger than $2^{31}$ bytes if it is defined as an extent file. The extent size determines the limit of its growth. It cannot be defined with statically-allocated extents (SAE).

If the value of `-organization` is `stream64`, `create_file` creates a 64-bit stream file, based on the following:

- If you specify `-record_size`, the extent size is assumed to be the value of `record_size`, and it must be a value that is a power of 2 in the range of 8 to 256, inclusive. In this case, a DAE 64-bit stream file is created. If you also specify `-extent_size`, it must be the same as the value of `record_size`. 

2-212
If you do not specify -record_size, the extent size corresponds to the value of -extent_size, if specified. If the specified size is 1 or if you do not specify -extent_size, the command creates a non-extent file. If you specify size with -extent_size, size must be either 1 or a value that is a power of 2 between 8 and 256. If you do not specify size, extents are flexible, increasing as the file becomes larger. Files with flexible extents are called flex files.

A 64-bit stream file can be sparse (that is, no actual disk space is required for blocks that consist of all binary zeros) and can grow to the size allowed by its extents. If a 64-bit stream file is larger than 2 GB, it cannot be copied or moved to modules running releases earlier than OpenVOS Release 17.2.x. However, such a file can be accessed across the network from a module running an earlier release as long as byte-positioning operations are not attempted.

Note: Where this manual refers to stream files, it also refers to 64-bit stream files, unless otherwise noted. The two are identical except that 64-bit stream files can be sparse; they cannot contain indexes, cannot be pipe files, and cannot be SAE files; and their size can grow beyond 2 GB.

Flex Files
A flex file is identical to a 64-bit stream file, except that the size of its extents varies in different regions of the file. Each size depends on the offset of the block that begins the extent within the file. Only 64-bit stream files can be flex files.

The maximum size of a flex file is 546,704,523,264 bytes.

You create a flex file by specifying -organization stream64 and -extent_size without specifying size. The extent size is the target module’s default value: normally, flex (or 8 for modules running a release earlier than OpenVOS Release 18.x).

Relative Files
A relative file contains fixed-length records whose size you specify. Although all records on the disk are the same length, the amount of data in each record can vary in length. As shown in the following illustration, each record starts with two bytes (one word) that contain the length of the data in the record. Thus, the physical record is two bytes longer than the specified record size.

<table>
<thead>
<tr>
<th>Size</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>16</td>
<td>This record contains 16-bytes of data.</td>
</tr>
<tr>
<td>34</td>
<td>This record contains 34-bytes of data.</td>
</tr>
</tbody>
</table>

OpenVOS only allocates those file blocks containing records that are accessed by a read or write operation.

When you delete a record in a relative file, OpenVOS does not delete the data in the record. Instead, it changes the record length to -1 to indicate that the record has been deleted. The space from logically deleted records can be regained by using the copy_file command and specifying the -pack argument.
For a relative file, you must specify the \texttt{-record\_size} argument, and the value must be between 1 and 32,767. You must specify the \texttt{-num\_records} argument if you specify a value other than 1 for \texttt{-extent\_size}.

**Fixed Files**

A fixed file contains fixed-length records whose size you specify. As shown in the following illustration, each record contains the same amount of data.

<table>
<thead>
<tr>
<th>This record contains 16 bytes of data.</th>
</tr>
</thead>
<tbody>
<tr>
<td>This record also contains 16 bytes of data.</td>
</tr>
</tbody>
</table>

You cannot physically delete a record in a fixed file.

For a fixed file, you must specify the \texttt{-record\_size} argument, and the value must be between 1 and 32,767. You must specify the \texttt{-num\_records} argument if you specify a value other than 1 for \texttt{-extent\_size}.

**Queue Files**

A queue is a special kind of file that one process can use to communicate with another process. A process or task that sends a message to the queue is called a requester. A process or task that reads a message from the queue is called a server. With a one-way queue, a requester can send a message but the server cannot reply to the message. With a two-way queue, a requester can send a message and the server can reply to the message. Each message unit in queue consists of a header and a message. The message is the character string sent by a process or task. The header contains information such as the message priority, message ID, the process IDs of both the requester and the server programs, and time and date data. The \texttt{create\_file} command supports message queues, one-way and two-way server queues, and one-way and two-way direct queues.

- A message queue is the least restrictive in terms of the actions that can be performed on the queue. Message queues are always one-way queues. Since both the header and the message reside on disk, message queues are the slowest type of queue. You can create a message queue by specifying the \texttt{message\_queue} value for the \texttt{-organization} argument of the \texttt{create\_file} command.

- One-way and two-way server queues are more restrictive than message queues in terms of the actions that can be performed on a queue. The header and message are not written to permanent storage; rather, they reside in kernel memory, which is managed by the operating system. Server queues are faster than message queues, but slower than direct queues. You can create a one-way server queue by specifying the \texttt{one\_way\_server\_queue} value for the \texttt{-organization} argument of the \texttt{create\_file} command. You can create a two-way server queue by specifying the \texttt{server\_queue} value for the \texttt{-organization} argument of the \texttt{create\_file} command.

- One-way and two-way direct queues are the fastest type of queue, but also the most restrictive in terms of actions that can be performed on a queue. Both the header and the message reside in wired memory. Each requester can handle only one message at a
time. You can create a one- or two-way direct queue by specifying the sequential value for the -organization argument of the create_file command. The file becomes a direct queue when it is opened by the $msg_open_direct subroutine.

For more information on the file organizations supported by the create_file command, see the VOS Reference Manual (R002). For information about queue file organizations, see the VOS Transaction Processing Facility Guide (R215).

**File Entries and Extent-based Files**

Files are allocated and stored on disks in units of blocks. A block contains 4096 bytes. When a file is created, storage is allocated on a disk to hold information about the file, such as its organization, maximum record size, and disk addresses for file data blocks. This stored information is called the file map. A file map can contain a maximum of 523,792 disk addresses.

An extent is a contiguous group of blocks represented by a single disk address. Subject to the limits described later in this Explanation in “File Size Limits,” extents permit files larger than 523,792 blocks. OpenVOS provides two types of extent-based files:

- **Statically-allocated extents (SAE) files** have their extent region entirely allocated and initialized when the file is created.
- **Dynamically-allocated extents (DAE) files** allocate and initialize extents as the file grows.

Table 2-13 lists advantages and disadvantages to using extent-based files.

**Table 2-13. Extent-based Files: Advantages and Disadvantages**

<table>
<thead>
<tr>
<th>Extent File Type</th>
<th>Advantages</th>
<th>Disadvantages</th>
</tr>
</thead>
<tbody>
<tr>
<td>Both SAE and DAE</td>
<td>Fewer disk addresses are required to represent a file. Sequential access may be optimized on the disk. Extents allow you to create larger files than you could create without extents. The system can often read the file map into the cache, improving performance.</td>
<td>Fragmentation could cause file-creation failure or growth failure.</td>
</tr>
<tr>
<td>SAE</td>
<td>Provides a larger extent size that is required for paging files.</td>
<td>The entire extent region must be allocated and initialized at file creation, which can take a long time. Subsequent growth occurs one block at a time. You cannot create SAE files over the network.</td>
</tr>
</tbody>
</table>
If the file is an extent-based file, the first \( n \) disk addresses in the file map are extents, rather than individual blocks. The number of extents in the file (the value of \( n \)) depends on the values of the \(-\text{extent\_size}\), \(-\text{dynamic\_extents}\), \(-\text{record\_size}\), and \(-\text{num\_records}\) arguments, all of which are described in this Explanation.

When the extent space allocated during file creation is filled with data, the file continues to grow one block at a time (for SAE files) or by extent size units (for DAE files), in combination with the arguments specified for the \textit{set\_file\_allocation}\ command. In either case, the address of each new block or extent is stored in a separate disk address in the file map.

### File Extents

The \textit{create\_file}\ command can set three types of file extents at file creation: simple files, SAE files, or DAE files.

- **Simple files**—The default \(-\text{extent\_size}\) value is 1. The \(-\text{extent\_size}\) value specifies the size, in blocks, of the extents used by the file. The number of records at creation time can be zero or more. If you enter a zero at creation time, a zero-length file is created. If you specify a nonzero number in the \(-\text{num\_records}\) argument, a single block at a time is allocated for the desired number of records. Disk blocks are allocated but not initialized. File growth occurs in single blocks.

- **SAE files**—At file creation, you must specify an \(-\text{extent\_size}\) value that is a multiple of 8 and a \(-\text{num\_records}\) value that is greater than zero. For fixed, relative, and extended sequential files, the value specified in \(-\text{num\_records}\) should be the maximum number of records in your file; for sequential and stream files, the value specified in \(-\text{num\_records}\) should be the maximum number of blocks in your file. The \(-\text{extent\_size}\) value multiplied by the number of extents determines the initial size of the file. All disk blocks are allocated and initialized for this file. After the initial area has been used, the file grows in single blocks. By default, OpenVOS creates extent files as SAE files. Note that you cannot use the \textit{create\_file}\ command with non-local SAE files if the \(-\text{extent\_size}\) size argument is greater than 1.

- **DAE files**—At file creation, you must specify an \(-\text{extent\_size}\) value that is a power of 2 between 8 and 256, and specify the \(-\text{dynamic\_extents}\) argument. You may also specify a \(-\text{num\_records}\) value of zero or greater. If zero is entered, a zero-length file is created. The file grows in extent-size units as long as enough contiguous space is available.
Creating Extent-based Files

The `create_file` command uses the arguments `-extent_size` and `-num_records` to control the creation of extent-based files. The `-extent_size` value multiplied by the number of extents determines the size of the file. You do not need to specify the number of records to create DAE files.

The `-dynamic_extents` and `-extent_size` arguments control the type of file that is created, as follows:

- If you specify the `-dynamic_extents` argument with a value of `no` and specify the `-extent_size` argument with a value of 1, the command creates a simple file.
- If you specify the `-dynamic_extents` argument with a value of `no` and specify the `-extent_size` argument with a value that is a multiple of 8, the command creates an SAE file.
- If you specify the `-dynamic_extents` argument with a value of `yes` and specify the `-extent_size` argument with a size other than 1, the command creates a DAE file. If you specify `size` with the `-extent_size` argument, it must be either 1 or a value from 8 to 256 that is a power of 2. If you do not specify `size`, the command uses an extent size of 64 for all file types except for 64-bit stream files, for which extents are flexible, increasing as the file becomes larger.

Extent Files and Disk Fragmentation

If you attempt to create or grow an extent file but sufficient contiguous space for an extent is not available, the error `e$insufficient_extent_space (4720)` is returned, and the action fails. This situation can occur if the disk is low on space or is significantly fragmented. The following sections describe how to address these issues in order to successfully create an extent file.

Handling Disk Fragmentation

Handle disk fragmentation as follows:

- When you create a file, specify the minimum extent size necessary to satisfy your requirements. If future fragmentation is a concern, preallocate the required blocks at file creation (block preallocation at file creation always occurs for SAE files and is optional for DAE files).
- Defragment the disk by saving files (especially simple files), deleting the files, and then restoring them. If there is not enough disk capacity to save these files to another disk, you can save the files to tape and restore them from tape. You can perform this process on a wider scale so that the entire disk is saved, and the files are deleted and then restored. For non-extent files, specify the `copy_file` command with the `-avoid_fragmentation` argument. Doing so relocates a file so that as many as possible of its disk blocks are contiguous. The success of this as a defragmentation tool depends on the disk’s current allocation patterns. This technique is more effective if you first free up significant disk space.
**SAE Failures**

If SAE file creation fails on a particular disk volume, try to reclaim excess disk space on the volume in an attempt to alleviate the failure. To reclaim excess disk space, log in as privileged, and then use `analyze_system` to examine the excess disk space for a volume, as follows:

```
as> list_disks
as> dump_ldte ldtep
```

The `list_disks` request lists each disk and its associated `ldtep`. The output of the `dump_ldte` request includes `Total fp excess`, the number of excess blocks available. If excess disk space is available, reclaim it with the following OpenVOS command:

```
set_partition_size disk_name excess size
```

The `excess` partition is not a true disk partition but rather a collection of excess disk space. The value of `size` is the amount of excess disk space, in blocks, that should remain. A value of 0 returns all excess disk space to the disk free space. A `size` value greater than the excess disk space has no effect.

**DAE Failures**

When you issue a request to create or grow a DAE or simple file and the request fails, a free-space emergency occurs. This situation is usually caused by low disk space or fragmentation issues. At this point, the free-space emergency recovery may be able to free enough space, and the request eventually succeeds. In other cases, however, the request will completely fail.

If the system variable `disk_space_emergency_warn$` is enabled (set to 1), the system writes the following error message to the system error log, regardless of whether the request has failed or eventually succeeded.

```
disk Allocators: space shortage on disk disk_name, extent_size=number, total_excess_blocks=number
```

The system variable `disk_space_emergency_warn$` has no effect on SAE files.

By default, the system variable `disk_space_emergency_warn$` is disabled (set to 0), which means that the system will not log the preceding error message if a free-space emergency occurs. If you want to receive this error message during free-space emergencies (regardless of whether the request fails or eventually succeeds), contact the CAC, and they will set the variable for you.

**File Size Limits**

File growth is limited by the following:

- the maximum number of records and the size of the extents for fixed, relative, or extended sequential files
- the maximum record byte offset for sequential or stream files
- the maximum number of blocks that can be allocated to a file by the file map
The following sections describe these limits.

**Maximum -num_records Value for Relative and Fixed Files**
The `-num_records` argument specifies the number of records in a relative or fixed file, and the number of 4096-byte blocks for a sequential or stream file. The maximum possible value of `-num_records` is \(2^{31} - 1\), although this is typically limited by other file characteristics, such as extent size, record size, and, in the case of extended sequential files, allocation unit size. In addition, file growth is limited from the theoretical maximum due to the reservation of the last 16 file-map entries and, in the case of sequential and stream files, the last seven blocks.

**Maximum Record Byte Offset for Sequential and Stream Files**
Records in sequential files and lines in stream files occupy a varying amount of space. The operating system uses the record byte offset to locate a record in a sequential or stream file. A record byte offset is the byte location of a record from the beginning of a file. The first record or first line in a file has a record byte offset of 0. The maximum record byte offset for a non-extent file is 2,145,452,032 (that is, 523,792 blocks). For a sequential file, growth may be limited to 2,145,415,168 (that is, 523,783 blocks), depending on the size of the last record written.

**Maximum Number of Blocks for Files without Extents**
Given enough disk space, an OpenVOS file with a relative, fixed, sequential, or stream organization that does not use extents can grow to the maximum file-map offset of 2,145,452,032 bytes or 523,792 data blocks (a total of 524,304 blocks, including the file map). Sequential files are limited to 9 blocks less than the maximum file-map offset (523,783).

**Maximum Number of File Blocks for Sequential Files with Extents**
The maximum size of an OpenVOS extended sequential file is based on whichever is less: the maximum file-map offset multiplied by the extent size, or the allocation unit size (as specified in the `-record_size` argument) multiplied by \(2^{31}\). The maximum number of data blocks in an extended sequential file is this value divided by 4096, minus 9, which is the number of blocks needed to hold a record of maximum size.

The maximum size of a sequential file with extents is the same as without extents. For extended sequential files, `create_file`, by default, selects an extent size that allows the file to grow approximately as large as the limit imposed by the allocation unit size.

**Maximum Number of File Blocks for Stream Files with Extents**
An OpenVOS 64-bit stream file that uses extents may grow as large as the maximum file-map offset of 2,145,452,032 bytes (523,792 blocks) multiplied by the extent size, or in the case of flexible extents, 131,870,736 blocks.

The maximum size of an ordinary (32-bit) stream file with extents is the same as without extents.

**Pre-allocating Disk Space for Stream and Sequential Files**
Use the `-num_records` argument to specify the initial number of blocks in a stream or sequential file. The file may grow beyond this initial size, as described in “File Entries and
create_file

Extent-based Files” earlier in this command description. For files with dynamically allocated extents, pre-allocation reserves blocks on disk but does not write to them.

To determine the number of extents that the create_file command allocates, divide the -num_records value by the -extent_size value. For example, to create a sequential file of 512 blocks with extents of 256 blocks, issue the following command. Note that the file will initially contain two \( \frac{512}{256} = 2 \) extents.

```
create_file seq_ex -organization sequential -num_records 512 -extent_size 256
```

To create a large 64-bit stream file with or without pre-allocating disk space, reset its end-of-file, as shown in the following example. Extents for 64-bit stream files are always dynamically allocated.

```
create_file flex -organization stream64 -extent_size reset_eof flex 540142534656
```

The preceding example creates a 64-bit stream file consisting of approximately 500 GB of binary zeros, but it uses virtually no disk space until a record is written that requires a disk block. If you do not specify a value for -extent_size, the command creates a file with flexible extents; a flex file can hold up to 540,142,534,656 bytes. Using -num_records to pre-allocate disk space creates an empty file but guarantees sufficient disk space for file growth, up to the specified amount. For example:

```
create_file str64 -organization stream64 -num_records 8380672 -extent_size 16 reset_eof str64 34327232512
```

The preceding example first creates a 64-bit stream file requiring approximately 32 GB of disk space. It then sets the file to contain 34,327,232,512 of binary zeros; 323,223,512 is the maximum number of bytes that fits in a file with extent size 16 (that is, 16 * 2,145,452,032). The operating system reserves disk space but does not write blocks, resulting in a fast operation.

Maximum Number of File Blocks with SAE for Relative Files

Given enough disk space, an OpenVOS relative file with extents may grow larger than an OpenVOS relative file without extents. The maximum blocks for a relative file are determined by the -extent_size, -record_size, and -num_records arguments. Use the formulas described in the following paragraphs to calculate the maximum number of records that a relative file with extents can contain.

Use the following formula to calculate the number of extent blocks in a relative file, first by adding 2 bytes to the specified -record_size byte value. OpenVOS appends two bytes to each relative record to store the actual record length. Multiply this sum by the specified -num_records value to obtain the number of bytes stored in the extents. When you specify the -num_records value for a relative file with extents, you are actually specifying the
number of records contained by the extents, **not** the number of records contained by the entire relative file. To convert the byte value into a block value, divide the product by 4096.

\[
\text{extent_blocks} = \frac{\text{num_records} \times (\text{record_size} + 2)}{4096}
\]

Use the following formula to calculate the number of extents in a relative file by dividing the \text{extent_blocks} value determined by the previous formula by the specified \text{-extent_size} block value.

\[
\text{extents} = \frac{\text{extent_blocks}}{\text{extent_size}}
\]

Use the following formula to calculate the maximum number of blocks in a relative file. The maximum number of blocks in a relative file includes both the blocks not in extents and the blocks in extents. To calculate the number of blocks not in extents, subtract the \text{extents} value determined by the previous formula from 523,792 blocks. The value 523,792 is the maximum number of block addresses that can be stored in a file map. One block address is needed to store the address of each extent. To the number of blocks not in extents, add the number of \text{extent_blocks} as calculated in the first formula.

\[
\text{max_num_blocks} = (523,792 - \text{extents}) + \text{extent_blocks}
\]

Use the following formula to calculate the maximum number of records that can be stored in a relative SAE file. This value always exceeds the specified \text{-num_records} value because it includes the blocks that are not in the extents. However, the value cannot exceed 2,147,483,647, as described in the section “Maximum \text{-num_records} Value for Relative and Fixed Files” earlier in this command description.

\[
\text{max_num_records} = \frac{\text{max_num_blocks} \times 4096}{\text{record_size} + 2}
\]

In the following example, the command creates a relative file with 500,000 records in file extents, with 8190 byte records, and with an extent size of 1000 blocks.

create_file ex_rel -organization relative -record_size 8190 -num_records 500000 -extent_size 1000

\[
\text{extent_blocks} = \frac{\text{num_records} \times (\text{record_size} + 2)}{4096} = 500,000 \times (8190 + 2) / 4096 = 1,000,000
\]

\[
\text{extents} = \frac{\text{extent_blocks}}{\text{extent_size}} = 1,000,000 / 1,000 = 1000
\]

\[
\text{max_num_blocks} = (523,792 - \text{extents}) + \text{extent_blocks} = (523,792 - 1000) + 1,000,000 = 1,522,792
\]

\[
\text{max_num_records} = \frac{\text{max_num_blocks} \times 4096}{\text{record_size} + 2} = 1,522,792 \times 4096 / (8190 + 2) = 761,396
\]

Note that the \text{max_num_records} value of 761,396 exceeds the specified \text{-num_records} value of 500,000.
Maximum Number of File Blocks with DAE for Relative Files
Use the following formula to calculate the maximum number of records that can be stored in a relative DAE file.

\[
\text{max\_num\_records} = 523\,792 \times 4096 \times \text{extent\_size} / (\text{record\_size} + 2)
\]

Maximum Number of File Blocks with SAE for Fixed Files
File block limits for fixed files are very similar to the file block limits for relative files. The only difference is that a record in a fixed file is not preceded by length bytes. OpenVOS determines the maximum file blocks for a fixed file by using the `-extent_size`, `-record_size`, and `-num_records` arguments. Use the formulas described in the following paragraphs to calculate the maximum number of records that a fixed file with extents can contain.

Use the following formula to calculate the number of extent blocks in a fixed file by multiplying the specified `-record_size` byte value by the specified `-num_records` value. When you specify the `-num_records` value for a fixed file with extents, you are actually specifying the number of records contained by the extents, not the number of records contained by the entire fixed file. To convert the byte value into a block value, divide the product by 4096.

\[
\text{extent\_blocks} = \text{num\_records} \times \text{record\_size} / 4096
\]

Use the following formula to calculate the number of extents in a fixed file by dividing the `extent_blocks` value determined by the previous formula by the specified `-extent_size` block value.

\[
\text{extents} = \text{extent\_blocks} / \text{extent\_size}
\]

Use the following formula to calculate the maximum number of blocks in a fixed file. The maximum number of blocks in a fixed file includes both the blocks not in extents and the blocks in extents. To calculate the number of blocks not in extents, subtract the `extents` value determined by the previous formula from 523,792 blocks. The value 523,792 is the maximum number of block addresses that can be stored in a file map. One block address is needed to store the address of each extent. To the number of blocks not in extents add the number of `extent_blocks` as calculated in the first formula.

\[
\text{max\_num\_blocks} = (523\,792 - \text{extents}) + \text{extent\_blocks}
\]

Use the following formula to calculate the maximum number of records that can be stored in a fixed SAE file. This value always exceeds the specified `-num_records` value because it includes the blocks that are not in the extents. However, the value cannot exceed 2,147,483,647, as described in the section “Maximum `-num_records` Value for Relative and Fixed Files” earlier in this command description.

\[
\text{max\_num\_records} = \text{max\_num\_blocks} \times 4096 / (\text{record\_size} + 2)
\]
In the following example, the command creates a fixed file with 1,500,000 records in file extents, with 4096 byte records, and with an extent size of 1000 blocks.

```
create_file ex_fixed -organization fixed -record_size 4096
-num_records 1500000 -extent_size 1000
```

extent_blocks = num_records * record_size / 4096
= 1,500,000 * 4096 / 4096 = 1,500,000

extents = extent_blocks / extent_size
= 1,500,000 / 1,000 = 1,500

max_num_blocks = (523,792 - extents) + extent_blocks
= (523,792 - 1,500) + 1,500,000 = 2,022,292

max_num_records = max_num_blocks * 4096 / record_size
= 2,022,292 * 4096 / 4096 = 2,022,292

Note that the max_num_records value of 2,022,292 exceeds the specified -num_records value of 1,500,000.

**Maximum Number of File Blocks with DAE for Fixed Files**

Use the following formula to calculate the maximum number of records that can be stored in a fixed DAE file.

```
max_num_records = 523792 * 4096 * extent_size / record_size
```

**Initial Maximum Extent-based File Size and Index Size**

You can create an extent-based index for an extent-based file by using the `create_index` command and specifying the `-extent_size` argument. The size of the index for an extent-based file, whether or not the index is extent-based, does **not** affect the initial maximum extent-based file size.

**Converting between Stream and Sequential Files**

To convert between stream files, 64-bit stream files, sequential files, and extended sequential files, you can use the `convert_stream_file` command. This command allows conversion among these types and any desired extent value. Not all files can be converted. For example, stream files may not be convertible to sequential files due to record-size limits, and stream files containing an index may not be convertible to 64-bit stream files. In general, files may not be convertible due to size limits together with extent size changes.

You can also use `convert_stream_file` to check whether files can be converted, without actually converting them.

**Converting Non-extent-based Files to Extent-based Files**

To convert a non-extent-based file to an extent-based file, first use the `save` command to save the file to tape. Then use the `restore` command and specify the number of records in the file, the type of extent file, and the size of the extents with the `-extent_num_records`, `-dynamic_extents` (if you are creating a DAE file), and `-extent_new_size` arguments, respectively. For SAE files, depending on the size of the extent file, it may take more than an hour just to create the new extent file.
create_file

Default Character Set and Shift Mode
The default character set and shift mode of a file are used by file and I/O services to store and return text file data in an appropriately translated format. The character sets that are supported for fixed, relative, or sequential files include ascii, latin_1, latin_9, kanji, katakana, hangul, simplified_chinese, chinese1, chinese2, and user_dbcs. Indexes on files having one of these default character sets are allowed only if the file’s shift mode allows no shifts; therefore, specify a value of none for shift_mode if you expect to create indexes for the text file. Indexes are not allowed for files with a multiple-byte default character set.

Access Requirements
To create a new file, you need modify access to the directory containing the new file.

Examples
The following examples show a variety of file characteristics that you can specify when you invoke the create_file command.

Example 1
To create a sequential file named memos in the current directory, use this command.

```
create_file memos
```

Example 2
The following command creates a relative file named this_week in the current directory.

```
create_file this_week -organization relative -record_size 7
```

The maximum size of a record in the file this_week is 7 bytes. That is, each record can have any size from 0 to 7 bytes.

Example 3
The following command creates a fixed file named this_month in the current directory.

```
create_file this_month -organization fixed -record_size 508
```

The size of each record in the file this_month is 508 bytes.

Example 4
The following command creates an extended sequential file where each record begins on a 0 mod 32 byte boundary.

```
create_file big_file -organization ext_sequential -record_size 32
```

The record size indicates the type that corresponds to the granularity of the record boundaries. The higher the value, the larger the file can grow; however, a higher value increases the potential of unused space between records. The file created here can grow to approximately 64 GB if it is an extent-based file. Because non-extent-based files are limited to approximately 2 GB, there would be no reason to create an extended sequential file without extents. For this reason, the default values applied in this example are -dynamic_extents yes and -extent_size 32.
Example 5
The following command creates a DAE 64-bit stream file with 1000 blocks preallocated and a maximum record size of 16 bytes.

```
create_file filex -organization stream64 -record_size 16 -num_records 1000
```

In the preceding example, `filex` can grow to approximately 32 GB. The following command provides information about `filex`.

```
display_file_status filex
name: %abcd_user>Stratus>JLS>filex
file organization: stream file (64-bit)
last used at: 13-02-15 15:29:58 est
.
.
.
dynamic extents: yes
extent size: 16
next byte: 0
blocks used: 1009
sparse: no
mode: w
```

The designation `(64-bit)` indicates that `filex` is a 64-bit stream file. The field `next byte` shows that the file is empty. The field `blocks used` shows that 63 extents were reserved (1008 blocks, which is 1000 rounded up to extent size), plus the indirect file map block. The field `sparse` indicates that `filex` does not use unallocated disk blocks to represent 4096 zero bytes. (A preallocated file is not sparse unless unallocated blocks occur before the end of the file.)

Related Information
See the command descriptions of `compare_files`, `copy_file`, `create_index`, `delete_file`, `display_file_status`, `dump_file`, `locate_files`, `move_file`, `set_file_allocation`, `set_ram_file`, `set_pipe_file`, `set_text_file`, and `truncate_file` for more information on files. For descriptions of the `save` and `restore` commands, see the manual *OpenVOS System Administration: Backing Up and Restoring Data* (R285). For a description of the `add_paging_file` command, see the manual *OpenVOS System Administration: Disk and Tape Administration* (R284). For detailed descriptions of each type of file organization, see the *VOS Reference Manual* (R002).
create_index

Purpose

This command creates an index to a file.

Display Form

```
------------------------------- create_index ------------------------------
path_name:
index_name:
key_components:
  -type:      embedded_key
  -collation: ascii
  -order:     ascending
-max_key_len: 64
-extent_size: 1
-dynamic_extents: no
-num_blocks: 0
-duplicates: yes
-null_keys: no
-automatic_update: yes
-work_dir:
-duplicate_path:
```

Command Line Form

```
create_index path_name
  [index_name]
  [key_components] ...
  [-type index_type]
  [-collation collation_code]
  [-order order_code]
  [-max_key_len length]
  [-extent_size size]
  [-dynamic_extents]
  [-num_blocks number]
  [-no_duplicates]
  [-null_keys]
  [-no_automatic_update]
  [-work_dir path_name]
  [-duplicate_path path_name]
```

Arguments

- **path_name**
  - Required
  - The path name of the file to be indexed.
**index_name**

The name of the index. This argument is optional only for an embedded-key index. An embedded-key index derives its name from the starting position of the initial-key component if you do not explicitly specify a name.

**key_components**

One or more key components. You must give the key components if you are creating an embedded-key index. You cannot give any key components when creating an item index or a separate-key index. A key component is defined by the position in a record of the first byte of the component and by the length of the component. Specify the position and length of a component as a pair of integers separated by a comma, and separate component definitions by spaces. For example, \(1,6\) \(80,1\) defines a two-component key. A key component for a varying length record is defined by the starting position of the length word of the record and the maximum record minus the length word. A varying string can have only one key component.

**-type index_type**

Specifies the type of index to be created. Possible values for `index_type` are `embedded_key`, `separate_key`, `embedded_separate_key`, and `item`. By default, the command creates an embedded-key index, and you must define a key with the `key_components` argument. You cannot create an embedded-key index on a stream file.

**-collation collation_code**

Specifies the data type of the keys, which determines the collating sequence for sorting the keys. The following are the possible values for `collation_code`.

- `ascii`
- `alphabetic`
- `numeric`
- `ascii_varying`
- `alphabetic_varying`
- `numeric_varying`

If you select `ascii`, the command interprets the data in the keys as ASCII characters and sorts the keys according to the ASCII collating sequence. If you select `alphabetic`, the command interprets the data in the keys as ASCII letters, treating uppercase and lowercase as one case, and sorts the keys into alphabetical order. If you select `numeric`, the command interprets the data in the keys as numbers (the data type is actually either fixed-length character string, or varying-length character string for `numeric_varying`, but each string is converted to an 8-byte integer), and sorts the keys into numerical order. An index sorted numerically usually requires approximately half the space of one collated in ASCII sequence. By default, the command uses the `ascii` collation code.

**Note:** If the value of `-type` is `separate_key`, `embedded_separate_key`, or `item`, the collating type is always varying. Even if you specify `ascii` or `alphabetic`, it is translated as `ascii_varying` or `alphabetic_varying`. 

---

*OpenVOS Commands Reference Manual (R098) 2-227*
create_index

- **-order order_code**
  Specifies the order in which to sort the keys. Possible values for order_code are ascending and descending. By default, the command sorts the keys in ascending order.

- **-max_key_len length**
  Specifies the maximum key length, in bytes, that is allowed in the index. For length, specify a value in the range 64 (the default) through 1280. The index type determines the value of length:
  - For the embedded_key and embedded_separate_key index types, length should be the total size of the specified key components.
  - For the item and separate_key index types, length is 64 bytes, by default.
  - An index that allows duplicate keys must not have keys longer than 1268 bytes. For separate-key indexes, length must be 1268 or less. For embedded or embedded-separate-key indexes, the total length of the components that make up the key must not exceed 1268. The -max_key_len argument has no effect on embedded keys, except that if specified, the value must be between 64 and 1280.

- **-extent_size size**
  Specifies the extent characteristics of the index. The permitted values are 1, positive multiples of 8 (up to 524,280), or a power of 2 between 8 and 256. If the size is 1 (the default value), the index is not extent-based.

  **Note:** Only extent-based files can have extent-based indexes.

- **-dynamic_extents**
  Creates a DAE index. By default, the create_index command does not create DAE indexes.

  For more information about creating DAE indexes, see the Explanation.

- **-num_blocks number**
  Specifies the number of blocks to preallocate for the index. The operating system uses this number to determine how many extents to create. If -extent_size is greater than 1, then -num_blocks must be greater than 0 (the default value). The maximum number of blocks is $2^{31}-1$.

- **-no_duplicates**
  Suppresses duplicate keys in the index. By default, the command allows duplicate keys.

- **-null_keys**
  Ignores keys consisting entirely of blanks. By default, the command treats blank keys as significant.

- **-no_automatic_update**
  Disables automatic update of indexes when you write or rewrite a record to a file with embedded keys or embedded separate keys. By default, the command adds new keys to indexes automatically.
-work_dir path_name

Specifies a directory to be used by the process using the index. If a work directory is not specified, the process directory is used. The work directory is used primarily by the disk sort subroutines.

-duplicate_path path_name

Logs invalid duplicate keys in the specified file but does not insert them into the index. This argument requires the path name of a file that contains the log of records with invalid duplicate keys. This file contains information enabling you to locate the records containing the invalid duplicate keys, as well as information about the record that the key locates and about the specific file and index being created. If no invalid duplicate keys exist or if the index allows duplicate keys, the file is deleted when the command terminates. If the file already exists, the command overwrites it.

If you do not specify this argument and the index contains duplicate keys, create_index terminates.

Explanation

The create_index command creates an index to the specified file. If the file is a text file with a default character set of ASCII, Latin alphabet No. 1 or katakana, you can add an index only if the shift mode is no; indexes are not allowed for files with a multiple-byte default character set. You cannot add an index if implicit locking is set on the file, and you cannot add an index to a 64-bit stream file.

The create_index command manages four types of indexes: embedded-key, separate-key, embedded-separate-key, and item. An index to a file is an ordered list of keys; one record in the file is associated with each key or, in the case of an item index, one item is associated with each key.

After creating an embedded-key or embedded-separate-key index, the command reads the file and adds to the index one key for each record. Thus, an embedded-key or embedded-separate-key index is not empty upon creation if the file itself is not empty. The added key is the concatenation of substrings of a record. Use the key_components argument to define the substrings.

In contrast, when the command creates a separate-key index or an item index, it does not add any keys.

For an embedded-key or embedded-separate-key index, the file system adds a key to the index when you add a record to the file, unless you select the -no_automatic_update argument. You must explicitly add keys to update a separate-key index. You can add keys to a separate-key index or an embedded-separate-key index by using programming language routines or by calling the s$add_key subroutine. To update an item index, call the s$add_item subroutine to add key-item pairs.

A varying string consists of two bytes that represent the current length of the record, followed by that number of bytes of data. The key component specification for a varying string should use the starting position of the two bytes that represent the length of the record (the length of the word) and the maximum length of the data itself.
Use the `collation` argument to specify the data type of the keys and the collating sequence used to sort the keys. The values for `collation` and their meanings are shown in the following table.

**Note:** This distinction between varying-length and fixed-length keys applies only to embedded keys. Separate keys are always varying-length keys.

<table>
<thead>
<tr>
<th>Value</th>
<th>Data Type</th>
<th>Collating Sequence</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>ascii</code></td>
<td>Fixed-length character string</td>
<td>ASCII</td>
</tr>
<tr>
<td><code>alphabetic</code></td>
<td>Fixed-length character string</td>
<td>Alphabetic</td>
</tr>
<tr>
<td><code>numeric</code></td>
<td>Fixed-length character string</td>
<td>Numeric</td>
</tr>
<tr>
<td><code>ascii_varying</code></td>
<td>Varying-length character string</td>
<td>ASCII</td>
</tr>
<tr>
<td><code>alphabetic_varying</code></td>
<td>Varying-length character string</td>
<td>Alphabetic</td>
</tr>
<tr>
<td><code>numeric_varying</code></td>
<td>Varying-length character string</td>
<td>Numeric</td>
</tr>
</tbody>
</table>

When the collating sequence is alphabetic, the command does not distinguish between uppercase and lowercase letters. When the collating sequence is numeric, the command converts the key (which is a character string) to an 8-byte integer, using the OpenVOS PL/I rules for conversion. For example, the value `'12'` is converted to the integer 12.

You can specify an ASCII, alphabetic, or numeric collating sequence of the character-varying data type for an embedded-key or embedded-separate-key index.

A key can contain up to 1280 characters (or bytes). If a numeric key is used, it can contain up to 18 digits.

If the collating sequence is `alphabetic` or `alphabetic_varying`, uppercase letters are translated to lowercase letters, and each character in Set 1 below that appears in a key is translated to the corresponding character in Set 2.

<table>
<thead>
<tr>
<th>Set 1</th>
<th>Set 2</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>[ ' ] ^</code></td>
<td>`{</td>
</tr>
</tbody>
</table>

An *extent* is a contiguous group of blocks represented by a single disk address. Subject to the limits described in the section “File Size Limits” (in the `create_file` command description), extents permit indexes larger than 523,792 blocks. OpenVOS provides two types of extent-based indexes:

- **Statically-allocated extents (SAE indexes)** have their extent region entirely allocated and initialized when the index is created.

- **Dynamically-allocated extents (DAE indexes)** allocate and initialize extents as the index grows.
The `-dynamic_extents` and `-extent_size` arguments control the type of index that is created, as follows:

- If you specify the `-dynamic_extents` argument with a value of `no` and specify the `-extent_size` argument with a value of 1, the operating system creates a simple index.

- If you specify the `-dynamic_extents` argument with a value of `no` and specify the `-extent_size` argument with a value that is 8 or a multiple of 8, the operating system creates an SAE index.

- If you specify the `-dynamic_extents` argument with a value of `yes` and specify the `-extent_size` argument with a value that is a power of 2 between 8 and 256, the operating system creates a DAE index. If you do not specify a value for the `-extent_size` argument, the error `e$invalid_dae_extent_size (7629)` is returned.

  Note: See the `create_file` command for more information about DAE and SAE files.

Access Requirements

You need write access to the file `path_name` and modify access to the directory containing the file to create an index to the file. You also need write default access to this directory so the index can be created.

Examples

**Example 1.**
Assume the file `sales` in the current directory contains records of sales with the name of the customer in a 16-character field starting at the 11th character in the record. The following command creates an embedded-key index named `customers` for the file `sales`.

```
create_index sales customers 11,16 -collation alphabetic
```

The embedded key has one component; it begins at character (byte) 11 in a record and has a length of 16. The collating sequence is `alphabetic`, so the index sorts the keys in alphabetical order.

**Example 2.**
The following command creates an item index named `dictionary` for a file named `empty` in the current directory.

```
create_index empty dictionary -type item -collation alphabetic -no_duplicates
```

The keys in the index are dictionary words, and each item gives the valid positions for hyphenation of the associated word. The index can only be filled by using the subroutine `s$add_item`.
create_index

Example 3.
The following is a sample file created by specifying the `-duplicate_path` argument.

```plaintext
cmd: %foo#m5>dir_1>dir_2>dir_3>huge_seq  
index: index_1

duplicate key:
00000000 494E4445 585F315F 4B45595F 37 | INDEX_1_KEY_7 |
ignored in record number 4417 [1141x] at position 4417 [1141x] (block 2 [2x] offset 320 [140x]):
00000000 494E4445 585F315F 4B45595F 37202020 | INDEX_1_KEY_7 |
00000010 20202049 4E444558 5F325F4B 45595F37 | INDEX_2_KEY_7|
00000020 30202020 20202049 4E444558 5F335F4B | INDEX_3_K |
00000030 45595F37 30202020 20202020 | EY_7 |
existing key locates record 385 [181x] at position 385 [181x] (block 1 [1x] offset 384 [180x]):
00000000 494E4445 585F315F 4B45595F 37202020 | INDEX_1_KEY_7 |
00000010 20202049 4E444558 5F325F4B 45595F37 | INDEX_2_KEY_7|
00000020 20202020 20202049 4E444558 5F335F4B | INDEX_3_K |
00000030 45595F37 30202020 20202020 | EY_7 |
```

In the preceding example, the command first identifies the file and then identifies the index. Next, the command identifies each invalid duplicate key with the following information.

- A dump of the duplicate key
- The position of the record where the duplicate key was found
- A dump of the record containing the invalid duplicate key
- The position of the record currently identified by the key
- A dump of the record currently identified by the key

Related Information

See the OpenVOS Subroutines manuals for descriptions of types of keys and indexes. See the description of the `create_file` command for information about DAE and SAE files.
create_record_index

**Purpose**
This command keeps a list of all record numbers in a file, as well as the actual location of the records in the file.

**Display Form**
```
----------------------------- create_record_index -----------------------------
path_name:                   
```

**Command Line Form**
```
create_record_index path_name
```

**Arguments**
- `path_name` **Required**
The file for which a record index is to be maintained.

**Explanation**
The `create_record_index` command enables the operating system to reuse file space.

The record index manages file space by tracking the record numbers and locations of each record of the file, and reusing the space made available by deletions. File I/O operations are unchanged but may occur more slowly, since the operating system must refer to the record index each time a record is accessed. File size is kept to a minimum, however, because space in the file is used efficiently.

The `create_record_index` command tracks the numbers assigned to each record written to the file. Generally, the numbers ascend sequentially, beginning with 1 (although the increment and first record number are dependent on the file organization). The file space of deleted records is reused, but the record numbers of deleted records are not (in contrast to what happens with a deleted record index where the record numbers can be reused). Thus, the writing of a new record occurs logically at the end of the file, but the actual storage of the record may be somewhere else.

Once a file has a record index, the index is updated for the life of the file. The index itself cannot be deleted.

Use this command when you want records stored efficiently, but also want the record numbers of new records to reflect their being written at the logical (not necessarily the physical) end of the file. It is most useful for files from which many records are deleted.
create_record_index

**Note:** The record index increments the record number regardless of a file’s physical space; therefore, when using the record index, you should use the locate_large_files command to identify how full a file is, rather than checking the number of blocks allocated to the file. See the description of the locate_large_files command for more information.

You cannot specify this command for a 64-bit stream file.

**Examples**

Consider a relative file named `%s1#d02>sales>Jones>discounts`, with 10 records. Create a record index by issuing this command.

```
create_record_index %s1#d02>sales>Jones>discounts
```

Suppose you subsequently delete record number 2 from discounts. The next record written to the file could be written at position 2, reusing that space. However, the new record would become record number 11; attempts to read record number 2 would report e$deleted_record (1269).

**Related Information**

See the description of the create_deleted_record_index command for information about another way to save file space.
create_tape_volumes

Purpose
This command initializes one or more tapes.

Display Form

```
create_tape_volumes

Purpose
This command initializes one or more tapes.

Display Form

create_tape_volumes

Arguments

- tape_device_or_port_name
  Required
  The name of the tape device or port attached to a tape drive.

- first_volume_id
  The volume ID of the first tape to be initialized. The tape facility disregards this argument when the tape is unlabeled, although it is printed as part of the mount requests message. By default, the command creates tapes with blank volume IDs.

- number_of_tapes
  The number of tapes to be initialized.

Command Line Form

create_tape_volumes tape_device_or_port_name
  [first_volume_id]
  [number_of_tapes]
  [-tape_format tape_format]
  [-first_cartridge_no cartridge_number]
  [-owner_id owner_id]
  [-message message]
  [-overwrite]
  [-no_unattended]
create_tape_volumes

- tape_format tape_format
  Specifies the default format of tape volumes created on this tape drive. The possible values for tape_format are ansi for ANSI-labeled tapes, ibm for IBM OS/VS-labeled tapes, ibm_mvs for tapes to be used on MVS/RACF systems, unlabeled for unlabeled tapes, and unix for tapes that have UNIX tar, cpio, or cpio formats. The operating system also sets the default translation mode according to the tape format you specify. When you choose the format ansi or unix, the default translation mode is ascii; when you choose the format ibm or ibm_mvs, the default translation mode is ebcdic; and when you choose the value unlabeled, the default translation mode is binary. If you do not specify a value, the system initializes the tape as ANSI format.

- first_cartridge_no cartridge_number
  The cartridge number to position the tape drive to before creating the first volume. By default, the tape drive does not position to a different cartridge before initializing the first tape.

- owner_id owner_id
  Specifies an owner ID for tape volumes to be created on this tape drive. When you attach a port to a tape drive, the operating system sets the default value of this parameter to your person name.

- message message
  Specifies a default message to be sent to the operator when you mount or dismount a tape volume. When a port is attached to a tape drive, the operating system sets the default value of this parameter to an empty string. If you reset this value, the operating system issues operator messages when volumes are mounted and dismounted. By default, the command does not issue operator messages.

- overwrite
  Specifies that any existing tape labels are to be overwritten. By default, the command does not overwrite the tape.

- no_unattended
  Specifies that an operator be prompted before creating each tape volume. By default, the command does not prompt the operator. This argument has no effect on tape devices for ftServer modules.

Explanation

The create_tape_volumes command initializes a tape or series of tapes by performing a mount_tape -create_volume, and dismount_tape on each tape.

The first_volume_id is the volume ID assigned to the first tape. The volume IDs for the second and succeeding tapes are automatically incremented. For example, if you give an ID of 001 for the first tape, the second tape will automatically be numbered 002. If you give an ID of az00 for the first tape, the second tape will automatically be numbered az01. Finally, if you give an ID of zz98, the second tape will automatically be zz99, and the third and fourth tapes will automatically be numbered aaa00, and aaa01, respectively.
If the tape to be initialized has already been initialized, the system displays the current volume ID and format and asks if you want to overwrite the tape. If you have specified `-overwrite`, the system displays the current volume ID and format and then overwrites the tape without asking for confirmation.

**Related Information**

See the manual *OpenVOS System Administration: Disk and Tape Administration* (R284) and the commands `mount_tape` and `dismount_tape`. 
**Purpose**

This command converts fixed files to stream files.

**Display Form**

```
--- cvt_fixed_to_stream ---
in_path: ...
```

**Command Line Form**

```
cvt_fixed_to_stream in_path
```

**Arguments**

- `in_path` **Required**
  
  The path name of a fixed file to be converted. The file name can be a star name. There are no restrictions on the name of the file.

**Explanation**

The `cvt_fixed_to_stream` command converts a fixed file of any record size to a stream file. The command overwrites the existing fixed file with the converted stream file.

If you attempt to convert a file that is not a fixed file, the operating system displays the following message:

```
Skipped path_name. Not a fixed file.
```

**Access Requirements**

To convert a file, you need modify access to the directory containing it, and write access to the file itself.

**Examples**

The following example converts a fixed file to a stream file:

```
cvt_fixed_to_stream perl.pm
```

The following example converts all `.profile` files to stream files:

```
cvt_fixed_to_stream *.profile
```
cvt_stream_to_fixed

Purpose

This command converts stream files to fixed files.

Display Form

```
--- cvt_stream_to_fixed ---
in_path:
-record_size:
-force: no
```

Command Line Form

```
cvt_stream_to_fixed in_path
   [-record_size record_size]
   [-force]
```

Arguments

- **in_path**
  Required
  The path name of a stream file to be converted. The file name can be a star name. If you specify the -record_size argument, the stream file to be converted does not require a suffix. If you do not specify the -record_size argument, the stream file to be converted must have one of the following suffixes:

  - .obj
  - .pm
  - .tto
  - .dump

- **-record_size record_size**
  Specifies the record size. The value must be between 1 and 32767.

- **-force**
  Overwrites the default record size. By default (no), the command does not overwrite the default record size.

Explanation

The cvt_stream_to_fixed command converts a stream file to a fixed file. The command overwrites the existing stream file with the converted fixed file.
If the path name has the suffix .pm, .tto, or .dump, the default record size is 4096. If the path name has the suffix .obj, the default record size is 1024. You can specify a file without any suffix, but if you do, you must specify the -record_size argument. If the path name contains one of the previously mentioned suffixes and you specified a value for record_size other than the default value for that suffix type, you can specify the -force argument to overwrite the default value.

If you attempt to convert a file that is not a stream file, the operating system displays the following message:

    Skipped path_name. Not a stream file

If you attempt to convert a stream file without a suffix and you did not specify a value for record_size, the system displays the following message:

    Skipped path_name. Suffix unknown

Access Requirements
To convert a file, you need modify access to the directory containing it, and write access to the file itself.

Examples
The following example converts a stream file to a fixed file:

    cvt_stream_to_fixed update_database.pm

The following example converts all the stream files with .tto suffixes to fixed files:

    cvt_stream_to_fixed *.tto
**debug**

**Purpose**
This command invokes the debugger from both command level and break level.

**Display Form**
There is no display form for the `debug` command.

**Command Line Form**
```
design
```
From command level:

```
design program
```

From break level:

```
design
```

**Arguments**
- `program`
  A command, program module, or saved image. You can use a path name or a file name to name a program module. The path name can be a full path name or a relative path name. If you use a file name, the debugger searches the current directory and the
  command library paths for a `.pm` file matching the file name.

If `program` expects arguments, supply them after `program`, as shown in the following example:

```
design program arg1 arg2 arg3 ...
```

When you are in the debugger, you can start program execution under control of the debugger with the `start` request.

If `program` is a saved executable image (keep module), you must include the suffix `.kp`. A *keep module* is a program that was interrupted and stored in its interrupted state in a file. If the system automatically generates a keep module, the file name has the suffix `.process_id.kp` (*process_id* is the program’s process ID, or PID). If you manually create a keep module (for example, via the debugger’s command line), the file name usually has the suffix `.kp`. When you specify a `.kp` module, the debugger uses both the `.kp` file and the corresponding `.pm` file that was executing when the keep
module was created. Therefore, the .pm file must exist, unmodified, in the same directory that it was in when the .kp file was created.

**Note:** When the system automatically generates a keep module, it creates a new keep module each time, rather than overwriting the old files. To prevent the disk from filling up with old keep modules, a system administrator should be aware of this issue and develop a plan for deleting these files.

You can use the debugger to examine a keep module. The debugger disregards any attempt to start or execute a keep module.

You must give a program argument when you invoke the debugger from command level. You cannot supply an argument when you invoke the debugger from break level.

Note that program cannot be a first token abbreviation. For example, if you have an abbreviation first pqr by print_quarterly_report, you cannot use debug pqr. Since the program name is not the first token of the command line, the abbreviation pqr is not expanded.

**Explanation**

The `debug` command invokes the debugger. You can invoke the debugger in two ways.

From command level, you can invoke the debugger to control program execution or to examine a saved image. The operating system loads the named program module or saved image and puts your process at debugger request level.

From break level, you can invoke the debugger by typing `debug`. Your process goes to debugger request level. At debugger request level, you can issue the requests described in the “Source-Mode Requests” section and the “Machine-Mode Requests” section later in this command description.

You can use the debugger on programs written in OpenVOS COBOL, OpenVOS PL/I, OpenVOS FORTRAN, OpenVOS Pascal, OpenVOS C, OpenVOS Standard C, or assembly language.

You can do any of the following in the debugger:

- start running a program
- set breakpoints in the program
- see the values of program variables and the contents of stack frames
- set the values of variables
- step through the program
- call procedures, subroutines, or OpenVOS C or OpenVOS Standard C functions with arguments

Debugger requests or the program being debugged can modify variables that are shared between processes.

One request, the `quit` request, returns your process to command level. The debugger discards the image of the executing program and all frames on the stack.
If you press the `<CTRL><BREAK>` keys while your process is in the debugger, the operating system puts you at break level. From break level, you can issue one of the break-level requests. When you go to break level from within the debugger, you can continue debugging the same program by issuing the `debug` request.

You can use either commas or spaces to separate a request’s arguments. A debugger request line cannot extend over more than one line; the maximum line length is 300 characters. Use semicolons to separate more than one debugger request in a request line. The following example contains two debugger requests, `set` and `continue`.

```
set d_amt = 4005; continue
```

Press the `RETURN` key to terminate a request line.

The debugger replaces first token abbreviations in your debugger requests if you enable abbreviation replacement in your process. However, the debugger uses only the first word of the output of the abbreviation directive. You can thus use abbreviations in debugger requests only to abbreviate the names of the debugger requests. See the *OpenVOS Commands User’s Guide* (R089) for more information on using abbreviations.

In a `#replace` statement, the debugger knows how to resolve the synonym for a literal constant or declared name. However, in an OpenVOS C or OpenVOS Standard C `#define` compiler directive, the debugger does not know how to resolve the token name or macro.

While your process is in the debugger, you can issue operating system internal commands as if you were at command level. To issue an internal command from debugger request level, you enter the name of the command preceded by two periods. For example, enter `..list` to issue the `list` command. The set of internal commands may change in subsequent releases of the operating system. To display a list of internal commands, enter the following command.

```
..help -type internal
```

Note that you can use abbreviations for internal commands (for example, `..1` for `list`).

A command line or command macro can contain multiple operating system commands. The following command line contains two commands.

```
dump print_reports; display_error (command_status)
```

From command level, commands on the same command line or in a command macro after the `debug` command may or may not execute, depending on whether the program being debugged terminates abnormally or normally.

- If the program being debugged terminates abnormally, commands following the `debug` command do **not execute**. For example, if you issue a `quit` request before the program terminates normally, the program has, in effect, terminated abnormally. Therefore, any commands on the same command line or in a command macro after `debug` do not execute.

- If the program being debugged terminates normally, commands following the `debug` command do **execute**. For example, if you issue a `quit` request after the program being
debugged terminates normally, any commands on the same command line or in a
command macro after debug do execute.

It is possible to use the debugger to debug programs running in tasking mode. When an error
causes such a program to go to break level, the break message appears on the process
terminal. You can enter the debugger and debug the program from the process terminal. Other
terminal I/O is still handled through the terminals that belong to specific tasks.

When you enter the debugger to debug a program running in tasking mode, the environment
of the debugger is set up to debug the task that invoked the debugger. To switch the
environment of the debugger to a different task, enter env -task task_id, where task_id
is a task ID number designating a different task. A task ID of 0 designates the task invoking
the debugger. A task ID of 1 designates the master task.

When you set a breakpoint in a program running in tasking mode, any task may hit the
breakpoint. It is not possible to force the debugger to allow only one task to stop executing at
the breakpoint.

Stopping at a breakpoint will cause the debugger to reset the terminal. One result is that in
programs with raw terminal input, the program’s input buffer is flushed.

### Debugging Optimized Code

Optimized code is difficult to debug. You might encounter various problems if you attempt to
debug code compiled at optimization levels 3 and 4.

When you compile using the -table argument, the optimization level defaults to 1 and
cannot be overridden. Therefore, the -table argument always ensures that you have optimal
use of all of the debugger’s capabilities.

When you compile using the -production_table argument, the optimization level is not
affected. The optimization level is the same as if you compiled using the -no_table
argument.

If you experience problems associated with optimization, try to isolate the object module that
contains the problem, and then recompile that module using a lower optimization. You can
bind objects that have different optimization levels together into the same .pm file. Refer to
the appropriate compile command description in this manual for more information on the
arguments that affect optimization levels. (The compile commands are c, cc, cobol,
fortran, pascal, and pl1.)

The following list describes some of the effects that optimized code might have on debugging.

- Optimization causes some variables to be saved in registers, instead of in memory
  locations. When variables are saved in registers, the set and display requests cannot
  read or change the current values of the variables. Those requests access memory
  locations, not registers.

  When you are debugging optimized code, the debugger issues one warning message
telling you that expressions may not evaluate correctly, to remind you that current
values might be saved in registers.
Optimization removes useless statements, such as those that perform operations on local variables that are never referenced again. Removed statements are not executable.

Optimization rearranges code into more efficient sequences. For example, optimization moves a statement out of a loop if the statement only needs to execute once. As a result, it might be difficult to relate source code to execution sequences. If code that was moved from a loop causes a fault, error messages indicate that the fault occurred in the first statement of the loop rather than in the statement the code originally came from.

Optimization rearranges machine instructions, making it difficult for the debugger to know which source statement is being executed.

Source-Mode Debugging

The source modes of the debugger correspond to the OpenVOS high-level languages OpenVOS COBOL, OpenVOS FORTRAN, OpenVOS Pascal, OpenVOS C, OpenVOS Standard C, and OpenVOS PL/I. You can debug a program written in one of these languages in the corresponding debugger mode. The requests that change the mode of the debugger to the source modes are `cobol`, `fortran`, `pascal`, `c`, and `pl1`. Use the `c` mode for both OpenVOS C and OpenVOS Standard C.

The mode determines how the debugger interprets your requests and how it displays information about the program you are debugging. For example, when the debugger displays the data type of a variable, the form is in the language associated with the mode.

To debug a program in a source mode, you must have compiled the program with a symbol table. The symbol table contains information about the names of variables and their locations in the compiled program. Use the compiler’s `table` or `production_table` argument to incorporate a symbol table into the program module.

Terminology

Throughout the explanation of the `debug` command and its requests, the following terms are used.

The block is the program unit currently in execution. The `current block environment` or `current environment` is the latest activation of the block that is executing. The current environment has an associated stack frame as well as source code in a program block. Table 2-14 lists the possible environments for each mode of the debugger.

<table>
<thead>
<tr>
<th>Debugging Mode</th>
<th>Current Block Environment</th>
</tr>
</thead>
<tbody>
<tr>
<td>c</td>
<td>Function or block</td>
</tr>
<tr>
<td>cobol</td>
<td>Run unit or compile unit</td>
</tr>
<tr>
<td>fortran</td>
<td>Main program, a subprogram, or a statement function</td>
</tr>
<tr>
<td>pascal</td>
<td>Program, procedure, or function</td>
</tr>
<tr>
<td>pl1</td>
<td>Begin block or a procedure</td>
</tr>
</tbody>
</table>
In all source modes, the debugger maintains the value of the current block environment and the current source line. The debugger also maintains the current code address, used in machine mode.

The debugger may reset the current environment when you issue a request that resets the current line. For example, when you issue the position or source request and enter a block of code other than the current block, the debugger resets the current environment to the new block. If several stack frames for a block are active on the stack, the debugger uses the most recent block activation. In addition, the debugger uses the block and stack pointer where the current line resides.

The current line is a line in the source module. The debugger resets the current line when it reaches a breakpoint while executing the program, or when you issue one of the following debugger requests: env, position, source, or step. The debugger resets the current line as follows:

- When it stops for a breakpoint, the debugger resets the current line to the source line of the breakpoint.
- When you issue an env request, the debugger resets the current line in the following manner.
  - When you issue an env request and specify a block environment that has not started or has ended, the debugger resets the current line to the first line in the specified block.
  - When you issue an env request and specify the block environment that is executing, the debugger resets the current line to the line following the last line executed in the current block.
  - When you issue an env request and specify a block environment where execution is suspended because program control was transferred to a called block, the debugger resets the current line to the last line executed in the suspended block.
- When you issue a position request, the debugger resets the current line to the new position.
- When you issue a source request, the debugger resets the current line to the last source line it displays.
- When you issue a step request, the debugger resets the current line to the line containing the statement following the last statement executed.

Note: The env, source, and position requests do not affect program execution. However, the step request does affect program execution.
The current line can be a line in an OpenVOS FORTRAN, OpenVOS Pascal, OpenVOS C, OpenVOS Standard C, or OpenVOS PL/I include file or an OpenVOS COBOL copy file. Every source line, including the current line, has a line number that is determined by its position in its source module. The number of a line in an include or copy file is in the following form.

\[ \text{file\_number} \text{- line\_number} \]

The `file_number` value is the number of the include or copy file. The `line_number` value is the line number within the include or copy file. For example, the first line in a source module is 1. The first line in the first include file is 1-1. The second line in the first include file is 1-2. The first line in the second include file is 2-1, and so forth.

The `current statement` is a source statement, a `then` clause of an `if` statement, or an `else` clause of an `if` statement. The debugger usually determines the current statement from the current line. The current statement is the first source statement, `then` clause, or `else` clause that starts on the current line. When such a statement does not start on the current line, the current statement is the first source statement, `then` clause, or `else` clause on the nearest line preceding the current line that contains the start of such a statement or clause.

The first and last lines in a program module are always possible current statements, although they may not be defined as executable statements. In all cases, the current statement is a source statement for which the compiler has generated some code. In all cases, the debugger informs you of the line number of the current statement when it stops for a breakpoint.

When a source line contains more than one statement, it is usually difficult to designate any but the first statement on the line as the current statement. For instance, although you can specify a statement in a `position` request that is not the first statement on a line, the debugger sets the current statement to the first statement on the line.

You can use the `step` request to set the current statement to any statement for which the compiler generates code, or to any source statement by setting a breakpoint on the statement in machine mode and then hitting the breakpoint while in source mode. To avoid the difficulties of setting breakpoints on a statement, you should write your programs so that no line contains more than one statement, `then` clause, or `else` clause.

The debugger updates the current statement only when execution stops at a breakpoint or when you issue one of the following requests: `env`, `position`, `source`, or `step`. The debugger always updates the current block environment when it updates the current statement. Therefore, the current block environment always contains the current statement.

The `current task` is the task you are currently debugging. The debugger updates the current task when you issue the `env` request with the `-task` argument, or when execution stops at a breakpoint.

**Frequently-Used Arguments**

When you enter the source-mode requests, certain arguments are frequently used as part of the requests. This section describes the following arguments.

- `number`
- `substring`
The number argument is an unsigned integer constant.

The substring argument is a reference to the OpenVOS PL/I or OpenVOS Pascal `substr` built-in function, to the OpenVOS C or OpenVOS Standard C `substr` operator, or to an OpenVOS FORTRAN variable reference followed by `(expression : expression)`.

The request_list argument is one or more debugger requests separated by semicolons and enclosed in parentheses. For example:

```
(source 1; display report(4).header; set report(4).header=0)
```

This request displays the current line, displays the value of the variable `report(4).header`, and assigns the value 0 to that variable.

You can give a request_list in a then clause or an else clause of an if request or in a break request. A request_list cannot contain any of the language-mode requests: machine, cobol, fortran, pascal, c, and pl1.

The line argument is a source statement line number. It must be the number of a source line on which a statement begins, on which a then clause begins, or on which an else clause begins. Thus, if a statement (other than an if statement) extends over more than one line, only the number of the first line can be specified as a value of line. For a source statement in an include or copy file, the line argument is in the form `file_number - line_number`.

The label argument is any valid entry point constant or label constant. To distinguish statement labels from line numbers, you must append a colon (:) to a statement label. For example, `loop : 100 :`, and `100` are valid values for label and line. The value `100 :` is a label; however, the value `100` is a line number. In fortran mode, a statement label must be preceded by a dollar sign ($).

Any label argument can have a colon appended to it to act as a terminator for the reference. The use of a colon as the explicit terminator for label is optional in the following requests: with a break request that does not include a request_list, or with the clear, continue, and disassemble requests.

Notes:

1. With the break request, you must use a colon to separate a label from a request_list.

2. Do not use a colon as the terminator for line. In this case, using a colon causes the debugger to interpret the number as a label, not as a line number.

The variable argument stands for a variable reference as defined by the language that corresponds to the debugger mode. The debugger uses the same rules the languages use for interpreting partially qualified variable references. Thus, the interpretation of variable
depends on the current environment. For example, the OpenVOS PL/I variable reference `report(4).header` refers to the member `header` in the fourth element of the array of structures `report`. Often, `variable` will be only a simple variable name.

The `variable` argument can be a variable that is shared between processes. If a variable is shared between processes, using the `set` request to assign a value to a variable changes the value for all processes.

**Use of the expression Argument in the High-Level Language Modes**

The `expression` argument stands for an expression. The following paragraphs describe how you specify this argument in each high-level language mode.

**Use of the expression Argument in c Mode**

In `c` mode, `expression` can be a constant, a variable reference, the OpenVOS C and OpenVOS Standard C operator `substr`, or the OpenVOS PL/I built-in function `length`. It can also be the OpenVOS C and OpenVOS Standard C library functions `memcpy`, `memmove`, `memset`, or `strlen`, a function reference known in the current environment, or a combination of these constructed by using any OpenVOS C and OpenVOS Standard C operators other than the following: increment and decrement (`++` and `--`), comma (`,`), conditional (`?:`), and assignment (`=`, `+=`, `-=`), etc.

The logical operators `&&` and `||` can be used, but both operands are always evaluated (in contrast to the way such constructs are handled in the C language, where the second operand is conditionally evaluated).

Casts are supported to a degree. Given a cast in the form `(typename declarator)`, `typename` must be a name identifying either an OpenVOS C or OpenVOS Standard C fundamental or derived data type or a type defined using `typedef`. It can also be a name in the form `struct tag, union tag, or enum tag`, where `tag` is defined in the source program. The optional `declarator` is a string of zero or more asterisks (`*`), which specify that the cast operand will be temporarily converted to a pointer type.

String constants, considered arrays (and thus addresses) in the C language, are interpreted as character strings when used in the debugger. For example, "abc" is treated as identical to `substr("abc")`. An exception to this interpretation occurs when a string constant is used as an argument in a function reference, in which case its address is passed.

**Use of the expression Argument in cobol Mode**

In `cobol` mode, `expression` can be a non-reference modified identifier, a constant, or a conditional expression consisting of the comparison of two identifiers or constants. (A `reference modifier` is a parenthesized list of one or two colon-separated values following an identifier. For example, `a(2:3)`. Its purpose is to indicate that the program is to access only a portion of the data.) For example, `100, city-code, base >= item, and amount >= 100` are valid `cobol` mode expressions. The permitted relational operators in `cobol` mode are the OpenVOS COBOL operators `<`, `<=`, `=`, `>=`, `^<`, `^=`, and `^>`, `^=`.

**Note:** In order to expand the types of expressions that you can use when debugging OpenVOS COBOL programs, you can debug OpenVOS COBOL programs in `pl1` mode. However, debugging OpenVOS COBOL programs in `pl1` mode has several drawbacks. If your COBOL variable names contain hyphens, for example, you cannot...
set or display them in pl1 mode. In addition, you cannot set breaks on non-unique paragraph names that are qualified by section names.

Use of the expression Argument in fortran Mode
In fortran mode, expression can be a constant, a variable reference, a substring reference, one of the OpenVOS PL/I built-in functions null, byte, rank, substr, length, and addr, or one of the OpenVOS FORTRAN intrinsic functions char,  ichar, and len. It can also be a function reference known in the current environment, or a combination of these constructed using the OpenVOS PL/I or OpenVOS FORTRAN arithmetical, logical, relational, and string operators.

Use of the expression Argument in pascal Mode
In pascal mode, expression can be a constant, a variable reference, one of the OpenVOS PL/I built-in functions byte, substr, length, and addr, or one of the OpenVOS Pascal predefined functions chr, eof, eoln, or ord. It can also be a function reference known in the current environment, or a combination of these constructed using the OpenVOS Pascal arithmetical, boolean, relational, set, and string operators.

Use of the expression Argument in pl1 Mode
In pl1 mode, expression can be a constant, the string pseudovariable, a variable reference, or one of the OpenVOS PL/I built-in functions null, byte, rank, string, substr, length, addr, and addr, a function known in the current environment. It can also be a combination of these constructed by using the OpenVOS PL/I arithmetical, relational, and string operators.

Note that pl1 debugging mode supports pointer constants even though they are not supported in OpenVOS PL/I. A pointer constant is an unsigned hexadecimal number starting with a decimal digit and having X or x as its final character or characters. It can appear as the left most pointer qualifier of a variable reference, and wherever a pointer-type expression can appear. For example, in pl1 debugging mode, the following is a valid statement where 11088x is a pointer constant.

    db? display 11088x->based_int

The string pseudovariable can appear on the left-hand side of a set request in pl1 mode.

Source-Mode Requests
From debugger request level, you can enter any of the source-mode requests. In addition, you can use the help request to display online information on all the source-mode requests or more information on one request.

- **args**
  Displays the values of all the arguments passed to the block of the current environment.

- **break label[:]
line
  [(request_list)] [-every number]**

  Sets a breakpoint and allows you to give a list of requests that the debugger executes after stopping for a breakpoint. The breakpoint is set so that program execution stops before the statement or clause on the specified line is executed. The breakpoint remains in the program until you clear it with a clear request.
**Note:** You cannot examine or set breakpoints in a shared library that has not yet been loaded.

When you omit `label` and `line`, the `break` request sets a breakpoint on the current statement. When you include `label` or `line`, the debugger determines the statement or clause that would be the current statement if the specified line were the current line, and sets a breakpoint on that statement or clause. See the “Terminology” section earlier in this discussion for information on how the debugger determines a current statement from a current line.

If you supply a `request_list` argument, the debugger executes the requests in the list every time execution stops at the breakpoint. If you supply an `-every` argument and specify a `number` value, the debugger stops program execution at the breakpoint only after reaching the statement the specified `number` of times.

If you omit a `continue` request from `request_list`, the debugger displays information about the state of the program at the breakpoint and goes to debugger request level. From this level, you can issue any debugger request, continue program execution with a `continue` request, or stop the debugging session with a `quit` request.

- **call**: `call procedure [(argument...)]`
- **call**: `call 'procedure' [using argument...]` (in cobol mode)

Calls the program with the entry name `procedure`, passing it arguments if given. An `argument` can be an expression in the language of the debugger mode when the mode is `fortran`, `pascal`, `c`, or `pl1`. When the mode is `cobol`, an `argument` must be a variable reference.

Use the first form of the `call` request when the debugger is in `fortran`, `pascal`, `c`, or `pl1` mode. If you use the first form with more than one argument, separate the arguments with commas. Use the second form in `cobol` mode.

The subprogram `procedure` must be accessible from the current environment. If the subprogram `procedure` returns, your process is at debugger request level. To display the value of a function, use the `display` request.

- **clear**: `clear [label [:] line [ -all ]]

Clears one or more breakpoints. When you omit an argument, the `clear` request clears the current breakpoint. The current breakpoint is the one that most recently stopped program execution, placing your process at debugger request level.

The `-all` argument clears all breakpoints. When you supply `label` or `line`, the `clear` request clears the breakpoint on the statement at the specified line. You cannot supply both `-all` and either `label` or `line`. 
clearw

Clears one or more watchpoints.

The `clearw reference` form is valid in any source mode except machine mode. It clears any watchpoint that maps exactly onto the storage used by `reference`. The `reference` value can be a variable, an array element, a pointer-qualified variable, or a substring. It cannot be a structure or an array.

The `clearw memory_reference num_bytes` form is valid in machine mode. It clears any watchpoint that maps exactly onto the storage whose address is designated by `memory_reference`, and whose length in bytes is given by `num_bytes`. If you omit `num_bytes`, the request clears watched storage up to the next mod4 boundary in memory. The `memory_reference` value must not be a register name or register range.

The `clearw -id n` form clears the watchpoint that is numbered `n`.

The `clearw -all` form clears all watchpoints.

continue

Resumes program execution after the debugger stops for a breakpoint. When you omit `label` and `line`, the `continue` request continues executing the program with the statement following the most recently executed statement.

If you supply `label` or `line`, the request transfers control to the associated statement and starts execution at that point. The statement specified must be accessible from the current statement. If the source module was compiled with the `-production_table` argument, supplying a `line` argument may produce unpredictable results.

When a `continue` request in a `request_list` is executed, any requests following the `continue` request are not executed.

disassemble

Displays the instructions generated for the statement specified by `label` or `line` or for the current statement if you do not include an argument. The debugger displays the instructions in assembly language code.

display expression

Displays the value of `expression`. The expression must be valid for the mode of the debugger. In general, it can be a computational expression that evaluates to a scalar, or it can be a reference to any of the following types of variables.

- a scalar
- a structure
In addition to the usual method of referencing an array in the particular language, the debugger allows you to reference an array using two other methods. With the first method, you use the `low:high` construct to specify a range of elements for that dimension of the array. The specifiers `low` and `high` are integer values indicating the low and high end of the range. With the second method, you use an asterisk (*) to represent all possible subscript values for that dimension.

The `low:high` construct or the asterisk appears in the position where a subscript value would normally appear. For example, the following `display` request uses the `low:high` construct to show elements 7 through 9 of the array `id_numbers`.

```
    display id_numbers
```

In `c` mode, you can use an empty set of brackets () to display all the characters of a variable up to but not including the null character (`\0`).

The `display` request displays whatever is currently in memory. If you are debugging optimized code, the most current variable values might be saved in registers instead of in memory. However, if you compile using the `-table` argument, the generated code always stores the current value of variables in memory, and you can safely use the `display` request.

If more than one invocation of a block exists on a stack (as in a recursive procedure), the `display` request changes the current environment in an undefined manner.

**Notes:**

1. In `fortran` mode, the `low:high` construct is not allowed.

2. In `c` mode, the `low:high` construct must be used to reference more than one element of an array defined with unspecified extents.

3. In `cobol` mode, `expression` must be a variable reference, enclosed in parentheses, as shown in the following:

   ```
   display array (expression_1:expression_2)
   ```

4. If you compile with the `-production_table` argument instead of the `-table` argument, you may not be able to use the `display` request if the optimization level is 3 or 4.

```
  ▶ dump variable [number]
```

Displays the value of `variable` as a hexadecimal number and as an ASCII character string. If you include the `number` argument, the debugger displays the specified number of bytes of data.
If you omit the *number* argument, the debugger displays a minimum of 16 bytes of data. However, if the specified variable (for example, an array) contains more than 16 bytes, the debugger uses the variable’s type to determine how much data to dump.

In Fortran mode, you must use a colon as a delimiter between the *variable* and *number* arguments because FORTRAN parsing eliminates spaces inside variable names. For example, issue the request `dump var1:8` instead of `dump var1 8` to distinguish `var1` from `var18`.

```plaintext
(env)
```

Sets the current environment to the environment you specify. The mode changes to the language of the new environment. The debugger tells you the new mode whenever it changes the mode.

When you issue an `env` request, the debugger resets the current line. See the “Terminology” section earlier in this discussion for more information on how the debugger resets the current line after you issue an `env` request.

You can specify one of the following arguments with the `env` request.

The *procedure* argument is the name of an active procedure or block with a frame on the stack. With this argument, the current environment is set to the most recent activation of the block named *procedure*. For example, the following request sets the environment to the most recent activation (the closest stack frame) of the procedure named `sort_report`.

```
env sort_report
```

If the procedure is inactive, the debugger searches for a procedure known in the current scope. If such a procedure is not found, the debugger searches for an external procedure having the specified name. If the debugger does not find an external procedure, it searches for an object module with the specified name. If the debugger finds none, it displays an error message.

The *stack_frame* argument can be either of the following:

- an unsigned integer representing the stack frame number. For example, the request `env 3` sets the current environment to the block activation of stack frame 3. For information on displaying stack frames and stack frame numbers, see the description of the `trace` request later in this discussion.

- a signed integer. The debugger adds the specified value to the current stack frame number to calculate the stack frame number of the new current environment. For example, if the stack frame number of your current environment is 4 and you issue the request `env -2`, the debugger sets the new current environment to that of stack frame 2.
The **-frameptr** argument specifies a `memory_reference`. With this argument, the debugger sets the current environment to the stack frame located at the address specified by `memory_reference`. For more information on the syntax of `memory_reference`, see the “Machine-Mode Memory Reference Argument” section later in this discussion.

When the stack has been corrupted, the **-frameptr** argument allows you to examine partially damaged stacks. In addition, this argument may also be useful when debugging applications that do not follow operating system stack standards. With this argument, the debugger does not switch tasks if the frame address is on a different stack.

The **-task** argument specifies a `task_id`. With this argument, the debugger sets the current task to that specified by `task_id`, and sets the current environment to the stack frame with the highest number that the `trace` request would display. A `task_id` of 0 identifies the current task.

The `begin.line_number` argument specifies the name of an OpenVOS PL/I begin block. In `pl1` mode, you can set the current environment to a begin block by supplying the name that the OpenVOS PL/I compiler gives the block. The name is in the form `begin.line_number`. The `line_number` is the line number of the source line on which the begin block starts.

**`help`** `[request_name]`

Displays online documentation. If you omit `request_name`, the `help` request displays the names and uses of all debugger requests. If you supply `request_name`, `help` provides information about the particular request.

**`if`** `expression` `then` `(request_list_1)` `else` `(request_list_2)`]

Executes `request_list_1` if the expression specified in `expression` is true. If you include an `else` clause and the expression is false, the `if` request executes `request_list_2`.

The allowed form of `expression` depends on the debugger mode. It must be a logical expression that evaluates to true or false. When the mode is `cobol`, the logical expression must be a comparison, using a relational operator, of two variables or of a variable and a constant. When the mode is `pl1`, `pascal`, or `c`, the logical expression can be any expression that the language allows in an `if` statement. When the mode is `fortran`, the logical expression can be any expression, enclosed in parentheses, that the language allows in a logical `if` statement.

**Note:** The entire `if` request, like all debugger requests, must be on one line.

**`keep`**

Creates a keep module (`.kp`) that you can use later for debugging.
**debug**

- **list**
  Displays the following information about all breakpoints:
  - where you set the breakpoint
  - how many times the debugger encountered the breakpoint
  - what debugger requests you issued at the breakpoint

- **listw [n] [-full]**
  Displays information about watchpoints.
  If you specify \textit{n}, the request displays information only about watchpoint \textit{n}. If you omit \textit{n}, the request displays information about all defined watchpoints.
  If you specify the \texttt{-full} argument, the request displays a memory compare list for any watchpoint whose new value differs from the old value. The \textit{memory compare list} shows memory words that have different values, as well as the values themselves (in hexadecimal). If you omit this argument, the request displays the values using natural data-type formatting. Note that watchpoints set in machine mode have no data type, so the \texttt{-full} argument is forced.

- **position \textit{identifier} \[ -\texttt{include} \]
  - \texttt{-no_include}**
  Resets the current line to the line specified by \textit{identifier}. Resetting the current line resets the current statement and the current environment.
  If \textit{identifier} is a character string, the debugger finds the first occurrence of the string in the source module after the current line, and resets the current line to the line containing the character string. If the character string contains spaces or punctuation marks, you must enclose it in apostrophes.
  If \textit{identifier} is an unsigned integer, the debugger resets the current line to the line with that source line number. If \textit{identifier} is a signed integer, the debugger adds the specified value to the current line number to calculate the new current line.
  If \textit{identifier} is in the form \texttt{file_number-line_number}, where both numbers are unsigned integers, the debugger resets the current line to the line numbered \textit{line_number} in the include or copy file numbered \textit{file_number}. For example, the following position request resets the current line to line number 14 in the third include file that was incorporated into the source file.

  \begin{verbatim}
  position 3-14
  \end{verbatim}

  If you specify the \texttt{-no_include} argument, the debugger \textbf{does not recognize} source code lines from include and copy files when you use the \texttt{position} request to reset the current line and when you use the \texttt{source} request to display source code. If you specify the \texttt{-no_include} argument, all subsequent \texttt{position} and \texttt{source} requests do not use source code from include and copy files until you issue a request specifying the \texttt{-include} argument.

  If you specify the \texttt{-include} argument, the debugger \textbf{does recognize} source code lines from include and copy files when you use the \texttt{position} request to reset the current
line and when you use the source request to display source code. If you specify the
-include argument, all subsequent position and source requests use source code
from include and copy files until you issue a request specifying the -no_include
argument.

Notes:

1. If you never specify either -include or -no_include, the
   -no_include argument is the default.
2. When identifier is in the form file_number-line_number, the
debugger ignores the -no_include argument and does recognize source
code from the specified include or copy file.

Even though the -no_include argument has been specified, the debugger recognizes
the contents of include and copy files if the current line is within the include file. For
example, the following situations cause the debugger to recognize the include file:

- if you change the current line to a line within an include file (for example, if you
  issue the request position 2-45)
- if program execution is suspended at a position within an include file
- if a change in the environment causes the current line to be a line in an include file

In the preceding situations, when you specify -no_include, the debugger does not
recognize other files nested within include files if you issue a position or source
request. It does recognize the main file and the include file containing the current line.

▶ quit
Ends the debugging session and returns your process to command level.

▶ return
Returns your process to break level after you have entered the debugger from break
level.

▶ regs
Displays the current contents of the processor registers. In addition to the contents of
the data and address registers, the regs request displays machine condition
information, such as the contents of the user stack pointer and the program counter. For
more information on registers, refer to the VOS Symbolic Debugger User’s
Guide (R308).

▶ set reference = expression
Assigns the value of expression to reference. The expression must be a scalar.
The reference can be a variable, a member of an array, a pointer-qualified variable, or,
where the language supports it, a string or substring reference. Multiple assignments,
such as a = b = 0, cannot be used.

Optimized code may not reload registers for every statement. As a result, the set
request may produce unpredictable results when you are debugging optimized code.
However, if you compile using the -table argument, the generated code reloads the
processor registers from memory for every statement, and you can safely use the `set` request.

- **source** `[number] [-include | -no_include]

Displays one or more lines of source code. When you omit a `number` argument, the `source` request displays the current line of source code. When you supply a `number` argument, it displays the specified number of source lines, starting with the current line. The debugger resets the current line to the last line displayed.

If you specify the `-no_include` argument, the debugger does not recognize source code lines from include and copy files when you use the `position` request to reset the current line and when you use the `source` request to display source code. If you specify the `-no_include` argument, all subsequent `position` and `source` requests do not use source code from include and copy files until you issue a request specifying the `-include` argument. If you never specify either `-include` or `-no_include`, the `-no_include` argument is the default.

If you specify the `-include` argument, the debugger does recognize source code lines from include and copy files when you use the `position` request to reset the current line, and when you use the `source` request to display source code. If you specify the `-include` argument, all subsequent `position` and `source` requests use source code from include and copy files until you issue a request specifying the `-no_include` argument.

For additional information on the `-no_include` and `-include` arguments, see the description of the `position` request earlier in this discussion.

If more than one invocation of a block exists on a stack (as in a recursive procedure), the `source` request changes the current environment in an undefined manner.

- **source_path** `[path_name] [-file_number number]

Allows you to do the following:

- When you omit the `path_name` argument, the `source_path` request displays the path names of all source modules used in compiling the current program.

- When you supply the `path_name` argument, the `source_path` request specifies the path name that the debugger will use to find the main source module that corresponds to the executing program module. The main source module appears first in the list of files specified with the `bind` command.

- When you supply the `-file_number` argument, the `source_path` request does one of the following:

  **Note:** If you omit the `path_name` argument, it displays the path name of the source module identified by `number`. If you supply the `path_name` argument, it specifies the path name that the debugger will use to find the source module identified by `number`.  

2-258
When you move or rename a source module, the debugger is not able to find the source module file because the program module’s symbol table indicates the wrong path name. To give the debugger the information it needs to find the source module, you use the `source_path` request and include the `path_name` argument and, optionally, the `-file_number` argument.

The `-file_number` argument allows you to display or change any single source module path name. The `number` argument specifies an included file in the current environment. For example, the following `source_path` request tells the debugger to change the path name of the fifth included file in the current environment to file name `sort_records`, located in the directory `version2`.

```
source_path -file_number 5 version2>sort_records
```

If `path_name` contains a file name, the debugger searches for the specified file relative to the current directory. If `path_name` is a directory name, the debugger searches for the source module’s compile-time file name in the specified directory. If `path_name` is an empty string (`''`), the debugger searches for the source module using the path name contained in the program module’s symbol table. The symbol table contains the path name of the source module at compile time.

The information you give in the `source_path` request is saved when the debugger changes the environment. Abbreviations defined using the `subsequent` directive in the abbreviations file are expanded in `path_name`.

**start**

Starts execution of the program specified in a `debug` command issued from command level. For example, if you enter `debug sort_reports` when your process is at command level, the debugger loads the program `sort_reports`. The debugger takes control, placing you at debugger request level. You can then examine the values of static variables, look at source code lines, or set breakpoints before starting the program. Issuing the `start` request then starts execution of the program under the control of the debugger.

**step [number] [-in -no_in]**

Executes one or more program statements. When you omit the `number` argument, the `step` request executes the current statement. When you supply the `number` argument, the debugger executes the specified number of statements, starting at the current statement. The debugger displays information about the executed statement or statements. The debugger resets the current statement to the statement following the last statement executed. It resets the current line to the source line in which the new current statement begins.

If you specify the `-in` argument, the debugger steps into any procedures or functions activated by the program. If you specify the `-no_in` argument, the debugger does not step into procedures or functions, treating procedure and function calls as single statements. The default argument is `-no_in`.
The step request is not suited for use in a tasking environment. When a step request in a request_list is executed, any requests following the step request are not executed.

- **symbol variable**
  Displays the declaration information about variable. The declaration information consists of the name of the block in which the variable was declared and the variable’s base address, data type, and size.

In c mode, you can display information about a structure, union, or enumerated type that has been defined with a tag by specifying `struct tag, union tag, or enum tag`.

- **task_status [task_id][-long][-all]**
  Displays the task ID, terminal port, and state of a specified task. If you specify the -long argument, the debugger also displays the task’s stack base, stack length, static base, static length, and the CPU time and page-fault count (if they are nonzero).

  If you omit the task_id argument, the debugger displays information for the current task. If you supply a value for task_id, the debugger displays information for the specified task. A task_id value of 0 identifies the current task. If you specify the -all argument, the debugger displays the information for all tasks.

- **trace [number][-all][-args][-on_units]**
  Displays information about the environments with frames on the stack. If you do not supply any arguments or you omit the number argument, the debugger displays information about the entire stack. If you include a number argument, it displays information about the specified number of block activations, starting with the most recent block activation.

  If you supply the -all argument, the debugger displays information about run-time language support procedures as well as your program’s procedures. If you supply the -args argument, an args request is executed for each frame on the stack. All frames on the stack are numbered, including frames for the run-time support routines. If you supply the -on_units argument, the debugger displays a stack trace with information about all active condition handlers.

- **watch [reference]**
  Sets a watchpoint.

  The watch reference form is valid in any source mode except machine mode. It sets a watchpoint that maps exactly onto the storage used by reference. The reference value can be a variable, an array element, a pointer-qualified variable, or a substring. It cannot be a structure or an array.

  The watch memory_reference num_bytes form is valid in machine mode. It sets a watchpoint that maps exactly onto the storage whose address is designated by memory_reference, and whose length in bytes is given by num_bytes. If you do not
specify `num_bytes`, the length of the watched storage will be up to the next mod4 boundary in memory. The `num_bytes` value must be equal to or less than 32,767. The `memory_reference` value must not be a register name or register range.

If you set a watchpoint on an automatic variable, make sure that the watchpoint does not survive past the end of the procedure. To accomplish this, set a breakpoint at the first executable statement after the statement that enters a procedure whose `request_list` sets the watchpoint. Then set a breakpoint at all return points from the procedure whose `request_list` clears the watchpoint.

When you watch an unshared variable, you are watching only one task’s instance of that variable. You do not see what occurs to other tasks’ instances of that variable.

**Note:** When you set watchpoints, ensure that the memory you are watching belongs to instantiated data for your program. If the instantiation vanishes (for example, entities on the stack when a subroutine returns), you should remove the watchpoint. If you do not remove it, the debugger’s behavior will be unpredictable.

▶ where

Displays information about the current line, the current statement, the current block environment, and the current task (if the program is running in a tasking environment).

**Machine-Mode Debugging**

You can debug any program when the debugger is in machine mode. Machine mode is also called the object mode of the debugger. In this mode, you can refer to code and data values by address instead of by name, and you can examine the contents of processor registers.

The `machine` request starts machine-mode debugging. If a module in the program you are debugging does not have a symbol table, the debugger resets its mode to machine mode automatically.

With larger applications, space considerations may make it impractical to compile an entire program with the `-table` or `-production_table` argument. These arguments incorporate a symbol table into the program module.

With larger applications, you can use the machine mode of the debugger to identify program blocks that require closer examination. Then, you can use the `-table` or `-production_table` argument to compile the source modules containing these blocks. After binding the application, you can debug the blocks containing errors using a high-level language source mode.

**Machine-Mode Memory Reference Argument**

You can use a `memory_reference` argument in the following machine-mode requests: `break`, `clear`, `continue`, `disassemble`, `display`, `dump`, `if`, and `set`. See Table 2-19 for the syntax of each of these requests. The allowed forms of a `memory_reference` argument are explained in this section.

A `memory_reference` can be an absolute memory address: an unsigned number or the sum or difference of two unsigned numbers. Optionally, you can specify the format of an absolute
memory address by appending a format character to the address. The format characters are as follows:

- o indicates octal
- d indicates decimal
- x indicates hexadecimal

If you do not specify a format character, decimal format is assumed.

A memory_reference can be a register or a sequence of registers. A register reference can be specified using either of the following forms.

```
register [.length]
```

```
register [.length]... register [.length]
```

To examine or affect only a particular part of the register, you can use the length suffix .b to indicate a byte length, .w to indicate a word length, or .l to indicate a long-word length of the register. When register is a data register and you do not specify length, the debugger assumes that you want to examine the entire 32-bit register. When register is an address register, the debugger examines the entire 32-bit register. For more information on registers, refer to the VOS Symbolic Debugger User’s Guide (R308).

In addition to an absolute memory address or register reference, memory_reference can be specified using any one of the following forms.

```
[ name [ (modifier) ] ] [ / region ] [. offset ]
```

```
[ offset ] ( memory_reference )
```

```
* [ offset ]
```

```
. offset
```

If you omit name or region, the debugger uses the current name and region. The current name and region are the last ones that you explicitly specified. The env request changes the current name to the most recent activation of the block you specify.

The following paragraphs explain the components of the memory_reference syntax.

The name can be a programmer-defined identifier specifying an entry point constant, label constant, object module, or external symbol name. The name can consist of the uppercase letters A through Z, the lowercase letters a through z, the digits 0 through 9, and the following characters.

```
$ _ @ [ \ ] ^ ` { } | ~
```
In name, you can use two apostrophes (’’) to represent a single apostrophe (’). If name contains a character other than the characters listed above, use apostrophes to enclose the name. For example, because the following name value contains a reserved character, a hyphen (-), you enclose the name in apostrophes.

'INDEX-MODE'

An optional modifier specifies whether name is a block, object module, or external symbol. Table 2-14 shows the block environments for each of the OpenVOS high-level languages.

If you do not specify a modifier, the debugger resolves name by first looking in the list of block names. You can use the trace request to display the names of active blocks. Second, it looks in the list of object module names in the module map. Third, the debugger looks in the list of external variables in the external variable map.

You can override this search order by specifying a modifier in parentheses after name. For example, if modifier is module, the debugger first looks in the list of object modules in the module map. Table 2-15 lists the valid modifiers and allowed modifier abbreviations.

**Table 2-15. Modifiers and Modifier Abbreviations**

<table>
<thead>
<tr>
<th>Modifier</th>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>stack</td>
<td>s</td>
<td>A block environment</td>
</tr>
<tr>
<td>block</td>
<td>b</td>
<td>A block environment</td>
</tr>
<tr>
<td>module</td>
<td>m</td>
<td>An object module name as specified during binding</td>
</tr>
<tr>
<td>extvar</td>
<td>e</td>
<td>An external variable</td>
</tr>
<tr>
<td>common</td>
<td>c</td>
<td>An external variable</td>
</tr>
</tbody>
</table>

The region specifies the address space region in which the offset is based. When name is an external variable, you do not need to specify region. If you use region without specifying name, do not precede region with a slash character (/). Table 2-16 lists the valid regions and allowed region abbreviations.

**Table 2-16. Regions and Region Abbreviations**

<table>
<thead>
<tr>
<th>Region</th>
<th>Abbreviation</th>
<th>Address Space</th>
</tr>
</thead>
<tbody>
<tr>
<td>code</td>
<td>c</td>
<td>Program code</td>
</tr>
<tr>
<td>symtab or table</td>
<td>t</td>
<td>Symbol table</td>
</tr>
<tr>
<td>static or own</td>
<td>i or o</td>
<td>Static data</td>
</tr>
<tr>
<td>stack</td>
<td>s</td>
<td>Stack data</td>
</tr>
<tr>
<td>ext_static</td>
<td>e</td>
<td>External static data</td>
</tr>
</tbody>
</table>

An offset can be a signed or unsigned number. An offset can also be the sum of or difference between two numbers. In this case, the number to the left of the operator (± or -) can be a signed or unsigned number. The number to the right of the operator is an unsigned number. The syntax for an offset is as follows:

\[
\left[ + , - \right] \text{number } \left[ + , - \right] \text{unsigned_number}
\]
The offset is positive and increments the address if you do not specify a sign or if you specify a plus sign (+) before offset. The offset is negative and decrements the address if you specify a minus sign (-) before offset. Optionally, you can specify the format of offset by appending a format character to the number or numbers in offset. The format characters are as follows:

- o indicates octal
- d indicates decimal
- x indicates hexadecimal

If you do not specify a format character, decimal format is assumed. If you use name without specifying region, do not include a period before offset. An asterisk (*) preceding offset indicates that it is an offset to the last location referenced.

**Note:** To make specifying stack offsets easier, the debugger evaluates an offset in the stack data region as negative if the first number in offset is hexadecimal and does not have an explicit sign.

Table 2-17 gives some examples of machine-mode memory references and a brief explanation of each reference.

### Table 2-17. Example Memory References

<table>
<thead>
<tr>
<th>Memory Reference</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>0E00128x</td>
<td>An absolute memory address: location 0E00128 hexadecimal</td>
</tr>
<tr>
<td>0E00128x+296</td>
<td>An absolute memory address expressed as the sum of two unsigned numbers: 0E00128 hexadecimal and 296 decimal</td>
</tr>
<tr>
<td>calculate(block)/code.174</td>
<td>A reference to an offset of 174 decimal into the code region of the block named calculate</td>
</tr>
<tr>
<td>main(module)/296</td>
<td>A reference to an offset of 296 decimal into the most recently referenced region of the module named main</td>
</tr>
<tr>
<td>c.128x or code.128X</td>
<td>A reference to an offset of 128 hexadecimal into the code region of the most recently referenced name</td>
</tr>
<tr>
<td>.296</td>
<td>A reference to an offset of 296 decimal into the most recently referenced name and region</td>
</tr>
<tr>
<td>global(extvar)/2</td>
<td>A reference to an offset of 2 decimal from the external variable global</td>
</tr>
<tr>
<td>d7</td>
<td>A reference to data register d7</td>
</tr>
<tr>
<td>-80(a6)</td>
<td>A reference to a negative offset of 80 decimal from the current value of address register a6</td>
</tr>
</tbody>
</table>
Many times a block environment has the same name as an object module. In this case, if you specify a block environment name and do not specify a region or offset, the debugger (by default) resolves the memory reference to offset 0 of the code region of the object module.

Other Machine-Mode Arguments
The following arguments used in machine-mode debugging are identical to the arguments used in source-mode debugging: number, request_list, and line. For an explanation of each of these arguments, see the “Frequently-Used Arguments” section earlier in this discussion.

In addition to the memory_reference argument, two other arguments are implemented uniquely in machine-mode debugging.

- relational_expression
- constant

**The relational_expression Argument**
The relational_expression argument is used in the if request. It must be a logical expression that evaluates to true or false. The syntax of relational_expression follows:

\[
\text{memory_reference relational_operator constant}
\]

The memory_reference is any valid memory reference. For more information on this argument, see the “Machine-Mode Memory Reference Argument” section earlier in this discussion.

The valid operators for relational_operator are as follows:

\[
< = > <= ^= >=
\]

**The constant Argument**
The constant can be an integer constant, PL/I character-string constant, or PL/I bit-string constant. The type and extent of the constant determines the type and extent of the logical comparison. In addition to its use in a relational_expression, the constant argument is also used with a machine-mode set request.

**Note:** Although the PL/I syntax is used to make machine-mode requests, the high-level language PL/I has no direct relevance to machine-mode debugging.

When constant is an integer constant, it is in the following form.

\[
\text{integer_constant.suffix}
\]

The rules for forming integer_constant are similar to the rules for forming an offset. An integer_constant can be a signed or unsigned number. An integer_constant can also be the sum of or difference between two numbers. In this case, the number to the left of

---

**Table 2-17. Example Memory References (Continued)**

| *+2          | A reference to an offset of 2 decimal into the most recently referenced location |

---

**OpenVOS Commands Reference Manual (R098) 2-265**
the operator (+ or -) can be a signed or unsigned number. The number to the right of the operator is an unsigned number.

An integer_constant is positive if you do not specify a sign or if you use a plus sign (+). An integer_constant is negative if you specify a minus sign (-). To indicate a number’s format, you can append one of the format characters: o, d, or x. The default format is decimal.

An integer_constant can be followed by an optional suffix. An integer_constant is considered to be a word (short integer) unless you specify a suffix. The allowed values for suffix are as follows:

- b indicates that constant is a byte
- w indicates that constant is a word
- l indicates that constant is a long word

The following examples are integer constants.

1024

FFx.l

When constant is a PL/I character-string constant, it is composed of ASCII characters enclosed in apostrophes. It can range from 0 to 256 characters in length. In a character-string constant, you can use two apostrophes (’’) to represent a single apostrophe (’). The following examples are character-string constants.

'It didn't help.'

'friday'

When constant is a PL/I bit-string constant, it is composed of a string of zeros and ones enclosed in apostrophes, followed by the letter b. It can range from 0 to 32 digits in length. An optional digit following the letter b represents the number of bits each character of the constant represents.

- 1 indicates binary characters: the digits 0 or 1
- 2 indicates base 4 characters: the digits 0 through 3
- 3 indicates octal characters: the digits 0 through 7
- 4 indicates hexadecimal characters: the digits 0 through 9 and the letters a through f

If you do not specify a digit following the letter b, binary characters are assumed. The following examples are bit-string constants.

'1'b

'ffff'b4

See the OpenVOS PL/I Language Manual (R009) for more information on character-string and bit-string constants.

Table 2-18 gives some examples of relational expressions and a brief explanation of each expression.
Machine-Mode Requests

The machine-mode requests include all the source-mode requests except call and symbol. The machine request and the requests that change the mode to that of a high-level language (for example, the pl1 request) are available in both source mode and machine mode.

The machine request also includes the endian_specifier argument. This argument is available only for the machine request. See “The endian_specifier Argument” later in this section for more information.

An object module contains a statement map when you use the assemble command without selecting the -no_statement_map argument. If the program module contains a statement map, the following requests have the same functionality and syntax in machine mode and source mode.

<table>
<thead>
<tr>
<th>env</th>
<th>source</th>
</tr>
</thead>
<tbody>
<tr>
<td>help</td>
<td>source_path</td>
</tr>
<tr>
<td>keep</td>
<td>start</td>
</tr>
<tr>
<td>list</td>
<td>step</td>
</tr>
<tr>
<td>position</td>
<td>task_status</td>
</tr>
<tr>
<td>quit</td>
<td>trace</td>
</tr>
<tr>
<td>regs</td>
<td>where</td>
</tr>
<tr>
<td>return</td>
<td></td>
</tr>
</tbody>
</table>

The remaining debugger requests have modified arguments, functionality, or syntax in machine mode. Table 2-19 lists the machine-mode functionality and syntax for each of these modified requests.

In the break, clear, continue, and disassemble requests, you can use memory_reference as an argument. You can append a colon (:) to the memory_reference argument to act as a terminator for the reference.

Notes:

1. With the break request, you must use a colon to separate a label name or a machine-mode memory reference from a request_list.

Table 2-18. Example Relational Expressions

<table>
<thead>
<tr>
<th>Relational Expression</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>d0 &gt;= 1111x.w</td>
<td>Specifies that the first word of data register d0 is greater than or equal to the integer constant 1111 hexadecimal</td>
</tr>
<tr>
<td>00FD6AD0x = 'L'</td>
<td>Specifies that the contents of memory address 00FD6AD0 hexadecimal are equal to the character-string constant L</td>
</tr>
<tr>
<td>setting1(extvar)/0 ^= '0101'b</td>
<td>Specifies that offset 0 of the external variable setting1 is not equal to the bit-string constant 0101 binary</td>
</tr>
</tbody>
</table>
2. With the `break`, `clear`, `continue`, and `disassemble` requests, do **not** use a colon as the terminator for `line`. Using a colon causes the debugger to interpret the number as a label, not as a line number.

In the `break` request, the debugger uses the colon to differentiate between the label or memory reference and the `request_list`. For example, a colon is required in each of the following `break` requests.

- `break label_name: (display variable_10; continue)`
- `break sublabel_name(5): (continue sublabel(3))`
- `break 00E000CEx: (regs; continue)`
- `break main(module): (display a5; continue)`

The use of a colon as the explicit terminator for a `memory_reference` is optional in the following requests: with a `break` request that does not include a `request_list`, and with the `clear`, `continue`, and `disassemble` requests.

### Table 2-19. Requests That Are Modified in Machine Mode

<table>
<thead>
<tr>
<th>Request</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>args</code></td>
<td>Displays pointers to the arguments for the current block, and for each argument shows four bytes of data in hexadecimal format. The syntax for the <code>args</code> request is as follows: <code>args</code></td>
</tr>
<tr>
<td><code>break</code></td>
<td>Sets a breakpoint at a specified machine address or line number. If you omit <code>memory_reference</code> and <code>line</code>, it sets a breakpoint at the current code address. If you specify the <code>request_list</code> argument, it allows you to give a list of requests that the debugger executes after stopping for the breakpoint. If you specify the <code>-every</code> argument, the debugger stops program execution at the breakpoint only after reaching the address (or line) the specified <code>number</code> of times. The syntax for the <code>break</code> request is as follows: <code>break [line </code>memory_reference<code>[:]</code>request_list<code>] [-every </code>number<code>]</code></td>
</tr>
<tr>
<td><code>clear</code></td>
<td>Clears a breakpoint at a specified machine address or line number. If you omit arguments, it clears the current breakpoint. If you specify the <code>-all</code> argument, it clears all breakpoints. The syntax for the <code>clear</code> request is as follows: <code>clear [line </code>memory_reference<code>[:]</code>-all<code>]</code></td>
</tr>
</tbody>
</table>
### Table 2-19. Requests That Are Modified in Machine Mode  *(Continued)*

<table>
<thead>
<tr>
<th>Request</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>continue</strong></td>
<td>Resumes program execution from the current breakpoint. If you specify the <code>memory_reference</code> or <code>line</code> argument, it resumes program execution from the instruction specified. The syntax for the <code>continue</code> request is as follows: &lt;br&gt; <code>continue line memory_reference[: ]</code></td>
</tr>
<tr>
<td><strong>disassemble</strong></td>
<td>Displays, as a pseudo-assembly-language instruction, the word or line specified by <code>memory_reference</code> or the current code address. If you omit <code>number</code>, the debugger disassembles one instruction and enters display mode. In machine mode, to show the next instruction, you press the [RETURN] key; to exit display mode, you press any key other than the [RETURN] key. If you supply <code>number</code>, the debugger displays the specified number of instructions but does not enter display mode. The syntax for the <code>disassemble</code> request is as follows: &lt;br&gt; <code>disassemble line memory_reference[: ] [number]</code></td>
</tr>
<tr>
<td><strong>display</strong></td>
<td>Displays the value specified by <code>memory_reference</code>. The <code>format</code> for the data shown can be one of the following: <code>d</code> for decimal, <code>b</code> for binary, <code>x</code> for hexadecimal, or <code>a</code> for ASCII. The default is decimal. If <code>memory_reference</code> is an address and you omit <code>number</code>, the debugger displays one line of data. If you supply <code>number</code>, the <code>memory_reference</code> must be an address. In this case, the debugger displays the specified number of bytes of data. The syntax for the <code>display</code> request is as follows: &lt;br&gt; <code>display memory_reference [format] [number]</code></td>
</tr>
<tr>
<td><strong>dump</strong></td>
<td>Displays the value of <code>memory_reference</code> as a hexadecimal number and as an ASCII character string. If you omit <code>number</code>, the debugger displays a minimum of 16 bytes of data. However, if the variable contains more than 16 bytes, the debugger uses the type of <code>memory_reference</code> to determine how much data to dump. If you supply <code>number</code>, the debugger displays the specified number of bytes of data. The syntax for the <code>dump</code> request is as follows: &lt;br&gt; <code>dump memory_reference [number]</code></td>
</tr>
</tbody>
</table>
The endian_specifier Argument

If you specify one of the optional endian_specifier arguments, the debugger assumes that all machine-mode memory references should be treated as either big-endian or little-endian.

- A big-endian memory reference is one in which the byte ordering is such that the lowest-addressed byte contains the high-order bits of a binary item.

- A little-endian memory reference is one in which the byte ordering is such that the highest-addressed byte contains the high-order bits of a binary item.

If the current environment is in an object module of the program module being debugged, the debugger will continue to use this assumption whenever it enters that object module in machine mode. Otherwise, the debugger uses the assumption only while in the current environment.

Table 2-20. Values for the endian_specifier Argument

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>big_endian</td>
<td>The debugger treats all machine-mode memory requests as big-endian.</td>
</tr>
<tr>
<td>big_endian</td>
<td></td>
</tr>
<tr>
<td>le</td>
<td></td>
</tr>
<tr>
<td>little-endian</td>
<td>The debugger treats all machine-mode memory requests as little-endian.</td>
</tr>
<tr>
<td>little_endian</td>
<td></td>
</tr>
</tbody>
</table>
Related Information

See also the description of the `mp_debug` command.
decode_vos_file

Purpose

This command reverses the effects of the encode_vos_file command on an encapsulated and/or encoded file. You can also use it to decode uuencoded or MIME base64 encoded files created on other systems.

Display Form

```
----------------------------------- decode_vos_file -----------------------------------
source_file:                       
destination_dir:                   
-overwrite: yes                     
-tell: no                           
```

Command Line Form

```
decode_vos_file source_file 
[destination_dir]
[-no_overwrite] 
[-tell] 
```

Arguments

- **source_file** Required
  The path name of a file to decapsulate/decode. You can specify only one file at a time (star names are not allowed).

Typically, *source_file* was created by encode_vos_file. However, decode_vos_file decapsulates/decodes uuencoded or MIME base64 encoded files created on other systems (for example, files received as email).

The -encode (uuencode) and -base64 (MIME) arguments of the encode_vos_file command are compatible with encoding done on UNIX systems, PCs, or mail-reading programs. The decode_vos_file command can usually handle files from these systems without pre-editing, provided that the message includes only one encoded block.

Email messages usually contain a free-text section followed by one or more encoded sections. This program extracts the first uuencoded or MIME base64 encoded section only. To extract subsequent encoded sections, or if errors are reported when processing an email file, edit the encoded section into a separate file and process that file.
decode_vos_file

**destination_dir**
The path name of the directory to which you want to write the file. If you do not specify destination_dir, the command writes the file to the current directory. The name of the destination file is the name of the original file that you encoded with encode_vos_file and is not related to the name of source file (which may have been renamed after the encoding occurred).

For MIME base64 encoded files from other systems, decode_vos_file uses the first file name found in a valid MIME header. If no file name is present, the command creates one by dropping the last suffix from the encoded file’s name and adding the suffix .DECODE to it.

If destination_dir is #null, the command does not perform any decoding, but it still performs the header-interpretation logic, including the -tell argument (if specified). This allows a macro that has successfully decoded a file to get the name and type of the extracted file by running this command again with -tell and the attach_default_output command.

**-no_overwrite**
Specifies that the command should not overwrite existing files that have the same names as those being decapsulated/decoded. By default (the value yes), the command silently overwrites these existing files.

**-tell**
Specifies whether to display the name of the destination file that is being created in the destination directory. The destination-file name is the name of the file that was originally encoded by encode_vos_file or by another system. The decode_vos_file command extracts this name from headers in the source file and is not related to the name of the source file itself, since encoded files can be renamed. By default (no), the destination-file name is not displayed.

The OpenVOS header, uuencode header, or MIME header is also displayed on a separate line after the destination-file name. An OpenVOS header is present if the file being decoded was created by encode_vos_file. Foreign files do not have an OpenVOS header, but they do have a uuencode header (that is, begin...) or MIME header (that is, Content-type...).

**Explanation**
The decode_vos_file command reverses the effects of the encode_vos_file command. It decapsulates (if necessary) and optionally decodes an OpenVOS file, back to the sequential, relative, or fixed file format.

**Related Information**
See the description of the encode_vos_file command.
decrypt

Purpose
This command converts ciphertext data into cleartext data.

Display Form
----------------------------------- decrypt ----------------------------------
input_file:                       decrypt                      output_file: [output_file]
-password:                        [ -password string
-delete: no                      [-delete]
-s suppress_password: no

Command Line Form
decrypt input_file
[output_file]
-password string
[ -delete]
[-suppress_password]

Arguments

► input_file  Required
The name of the file to be decrypted. This file can have any file organization
(sequential, relative, fixed, or stream), and it can have the pipe attribute. However, it
cannot have any indexes.

► output_file
The name of the file to contain the decrypted data. This file has the same file
organization as input_file. If input_file has the pipe attribute, output_file
has it, too. If you specify neither output_file nor the -delete argument, the
command asks if you want to replace input_file with an output_file of the same
name. Type y or Y at the prompt if you want to replace it. If you type n or N at the
prompt, input_file is not decrypted.

► -password string  Required
A string that forms the decryption key. The string must have a length of one to eight
characters and can contain any character.

Note: Abbreviations are not expanded.
- **-delete**
  Suppresses the prompt that occurs if you specify the same file for both `input_file` and `output_file`, or if `output_file` already exists. By default (the value `no`), the command displays this prompt.

- **-suppress_password**
  Prevents the command from attempting to use the password saved in `input_file`. By default (the value `no`), the command reads the saved password from `input_file`.

### Explanation

The `decrypt` command converts ciphertext data into cleartext data. The command reads the input file, decrypts the contents of the input file using the Data Encryption Algorithm (DEA) in cipher feedback 8 mode, and writes the decrypted data into the output file. By default, the output file replaces the input file.

By default, the `decrypt` command verifies that the password specified on the command line matches the decrypted value of the saved password. If they match, the `decrypt` command decrypts the file. Otherwise, it displays an error message and terminates. This action avoids decrypting the file with the wrong key, which would produce a useless output file and could result in the deletion of the original, encrypted, input file.

When you encrypt a fixed or stream file using the `-suppress_password` argument, you must also use this argument when decrypting the file.

When you encrypt a sequential or relative file using the `-suppress_password` argument, the initial zero-length record informs the `decrypt` command that no password is available. Therefore, you are not required to specify the `-suppress_password` argument when you are decrypting sequential or relative files.

If you use the `-suppress_password` argument to encrypt a file, and then you accidentally use the incorrect password to decrypt the file, and finally, you overwrite the input file with the (incorrectly) decrypted output, you must re-encrypt the file with the incorrect password, and then decrypt it with the correct password.

The encryption and decryption algorithm is entirely dependent on the user-specified password; no other information is used during the encryption or decryption process.

### Access Requirements

You need read permission on the input file, and you need modify access and default write permission on the directory containing the output file.

### Related Information

See the `encrypt` command. Also, for more information about DEA support, see the following files:

- `system>doc>dea.doc`
- `system>doc>tdea.doc`
delete_dir

Purpose
This command deletes a directory and all of its contents.

Display Form

```
---------------------------------- delete_dir -----------------------------------
directory_names: ...
-ask: yes
-expired_only: yes
-brief: no
```

Command Line Form
```
delete_dir directory_names...
   [-no_ask]
   [-no_expired_only]
   [-brief]
```

Arguments
- **directory_names** Required
  One or more names or star names of directories to be deleted.

- **-no_ask**
  Suppresses the prompt asking whether to delete a directory. By default, the command asks before deleting a directory, unless the directory is empty.

- **-no_expired_only**
  Deletes all files, including those that have an expiration date some time in the future. By default, the command deletes only those files whose expiration dates have passed or for which no expiration date is set.

- **-brief**
  Suppresses the display of each directory name that matches a star name before the directory is deleted. By default, the command displays the names.

Explanation
The `delete_dir` command deletes the target directory if a `directory_names` term matches a link whose target is a directory. The link remains, even though it has no target.

If the `delete_dir` command encounters a file that it cannot delete, the file is not deleted. If such a file is contained in a subdirectory, that subdirectory is not deleted. If a directory
contains any object that cannot be deleted, the command deletes all files and subdirectories that can be deleted, but it does not delete the directory itself or any links it contains. Normally, the command returns an error message for each file and subdirectory that cannot be deleted.

Examples of files that cannot be deleted are files with the safety switch set, hidden files, and files whose expiration date has not passed unless you have specified the -no_expired_only argument. A directory containing hidden objects is never deleted, and e$dir_not_empty (1095) is returned regardless of the release the module is running. However, if the module is running a release that supports extended names, an error message is reported for each hidden object. See Using OpenVOS Extended Names (R631) for more information about hidden files.

Access Requirements
You need modify access to the directory immediately containing a directory you delete.

Examples

Example 1.
To delete the directory >east>Jones>customers, use this command.

    delete_dir >east>Jones>customers

Example 2.
Suppose your current directory contains the directories reports.jan90, reports.feb90, reports.mar90, and reports.apr90 among others. You want to delete all the reports directories except reports.apr90. The following command prompts you as to whether you want to delete each directory with a name that matches the star name reports.*.

    delete_dir reports.*

If you answer yes to each prompt except the one for reports.apr90, the command deletes all but that directory.

Example 3.
Suppose your current directory contains these directories.

    month.7
    month.8
    month.9

The directories month.7 and month.8 are empty, but month.9 contains the following directories.

    week.1
    week.2
    week.3
    week.4

Now suppose that you issue this command.

    delete_dir month.*
The command deletes the directories `month.7` and `month.8` without asking because they are empty, but asks you before deleting `month.9`, since it is not empty.

**Related Information**

See also the description of the `set_expiration_date` command. For more information about directory management, see the command descriptions of `change_current_dir`, `compare_dirs`, `copy_dir`, `create_dir`, `display_current_dir`, and `move_dir`. For information about removing a link, see the `unlink` command.
**delete_file**

**Purpose**
This command deletes files from the directory hierarchy.

**Display Form**

```
-------------------------- delete_file --------------------------
| file_names: [**---**]   |
| -ask:    yes           |
| -brief:  no            |
| -force:  no            |
```

**Command Line Form**

```
delete_file file_names...
    [-no_ask]
    [-brief]
    [-force]
```

**Arguments**

- **file_names**  
  One or more names or star names of files to be deleted.

- **-no_ask**  
  Suppresses the prompt, when a file name is a star name, asking whether to delete a file with a matching name. By default, the command asks before deleting a file with a matching name.

- **-brief**  
  Suppresses the display of each file name that matches a star name before the file is deleted. By default, the command displays the name.

- **-force**  
  Deletes a file not currently in use and marks for deletion a file that is in use. In the second case, the command deletes the file after all users have closed it. By default, the command ignores files that are in use.

**Explanation**

The `delete_file` command deletes the target file if a `file_names` term matches a link whose target is a file. The link remains, although it has no target.
If you mistakenly delete an important file, your only recourse is to restore the file from your latest backup tape. The OpenVOS file system does not have “un-delete” capabilities. You can prevent the deletion of important files by performing one or more of the following actions.

- Create an abbreviation for the `delete_file` command that contains the `-ask` argument. For example:

  ```
  first del by delete_file -ask
  ```

  When you specify the `-ask` argument, the command prompts you before deleting the file.

  **Note:** When `file_names` is a star name, unless you specify `-no_ask`, the command prompts you before deleting a file with a matching name. When you answer prompts about a set of files, no files are deleted until you have responded to prompts for the entire set. If you cancel the command before answering prompts about all files in the set, no files in the set are deleted. After you have answered prompts for all the names that match one `file_names` term, however, the command deletes the files in that set before asking you about the next set of names.

- Create an abbreviation for `delete_file` which moves the file rather than deletes it. If you move it to your `(process_dir)` directory in

  ```
  (master_disk)>system>process_dir_dir
  ```

  OpenVOS deletes the file when you logout, but it is available before then in case you need it. For example:

  ```
  first del by move_file &1& (process_dir)>&1&
  ```

- If using the `process_dir_dir` directory is too risky, create an abbreviation that moves the deleted file to a directory where it will remain when you logout.

- Use the `set_safety_switch` command to set a file’s safety switch to on. If you set the safety switch for a file and then you attempt to delete it, OpenVOS displays the following message:

  delete_file: Operation invalid while file is protected by the safety switch. `path_name`

  To delete a file that has the safety switch turned on, use the `set_safety_switch` command to turn off the safety switch.

- Use the `set_expiration_date` command to prevent a file’s deletion before a specific date and time. If you set the expiration date for a file and then you attempt to delete it, OpenVOS displays the following message:

  delete_file: The object may not be deleted before its expiration date. `path_name`

  To delete the file, use the `set_expiration_date` command to reset the expiration date to today’s date.

The `-brief` argument suppresses messages telling you which files are being deleted.
Access Requirements
You need modify access to the directory immediately containing any file that is to be deleted.

Examples
Suppose clark_report is a link in the current directory to a file, and the current directory contains the files week90-02-04, week90-02-11, week90-02-18, week90-02-25, and week90-03-03. The following command deletes some of these files.

```
delete_file week90-02-* clark_report
```

The command deletes the files in the current directory whose names match the given star name and the target of the link clark_report. The link remains in the directory. The deleted files are week90-02-04, week90-02-11, week90-02-18, week90-02-25, and the target of clark_report.

Related Information
See also the command descriptions of compare_files, copy_file, create_file, display_file_status, dump_file, locate_files, move_file, set_file_allocation, and truncate_file. For information about removing a link, see the unlink command.
**delete_index**

**Purpose**
This command deletes a set of indexes to a file.

**Display Form**

```
--------------------------------- delete_index ---------------------------------
file_name:                   index_names:               -ask: [yes]
```

**Command Line Form**

```
delete_index file_name index_names ... [-no_ask]
```

**Arguments**

- **file_name**
  
  Required
  
  The path name of a file.

- **index_names**
  
  Required
  
  One or more names or star names of indexes to be deleted. The command deletes all indexes with matching names.

- **-no_ask**
  
  Suppressed the prompt that asks you, when an index name is a star name, whether to delete an index with a matching name. By default, the command asks before deleting an index with a matching name.

**Explanation**

The `delete_index` command deletes one or all indexes to the file `file_name`.

When you are responding to prompts about a set of indexes whose names match `index_names`, the `delete_index` command does not delete any index in the set until it has asked you about all of them. If you cancel the command before answering prompts about all indexes in the set, no indexes in the set are deleted.

**Access Requirements**

You must have write access to the file `file_name` and modify access to the directory containing the file.
Examples

Example 1.
To delete all the indexes to the file `reports>this_week`, issue this command.

    delete_index reports>this_week *

Example 2.
To delete the indexes `name` and `zip_code` to the file `new`, issue this command.

    delete_index <Smith>joness_customers>new name zip_code

Related Information

See also the descriptions of the `create_index` and `truncate_file` commands. When you truncate a file, you truncate, but do not delete, all indexes to the file.
delete_library_path

Purpose
This command removes a path name from the list of directories that define the specified library.

Display Form

Command Line Form

```
delete_library_path [library_name]
```

Arguments

- `library_name`
  Specifies one of the following library names.
  - include
  - object
  - command
  - message

By default, the command looks for the `library_path_name` in the include library.

- `library_path_name`
  Required
  The path name of a directory. The path name can include the command functions (current_dir) or (home_dir); if `library_name` is message, the path name can also include (referencing_dir) and (language_name). If enclosed in apostrophes, the command functions are evaluated when the path name is used.

Explanation
The delete_library_path command allows you to remove a single path name from the list of directories that define the specified library.
A library is one or more directories that the operating system searches for objects of a particular type. Each module has the following libraries.

- include library
- object library
- command library
- message library

The compilers search the include library for include files; the binder searches the object library for object modules; and the command processor searches the command library for commands and the message library for messages associated with individual commands.

For each library, the search for an object begins in the first directory on the library list. If the object is not in that directory, the search proceeds to the second directory on the list, then to the third, and so on. The module's default list of directories for each library serves as a guide to where to find objects. The delete_library_path command enables you to delete any path name in the list for a specified library, so you can control which directories the operating system searches for an object.

The path name of the directory library_path_name can include the command functions (current_dir) or (home_dir); if library_name is message, the path name can also include (referencing_dir) and (language_name). (Note that you must enclose a command function in apostrophes in order to prevent its evaluation by the command processor at the time you give it to the delete_library_path command.)

The list of libraries modified by the delete_library_path command remains in effect only for the life of your process.

**Examples**

Suppose you use the list_library_paths command to list your include library and you get the following output.

```
include library directories:
  (current dir)
  %s1#d03>Sales>incl
  %s1#d03>Sales>Smith>work
  %s1#d04>system>include_library
```

To delete the directory named work, execute the following command.

```
delete_library_path include work
```

Now if you list your include library, you get the following output.

```
include library directories:
  (current dir)
  %s1#d03>Sales>incl
  %s1#d04>system>include_library
```
Related Information

See also the command descriptions of `add_library_path`, `list_library_paths`, and `set_library_paths`. See the OpenVOS Commands User’s Guide (R089) for information about the conventions for searching libraries. The descriptions of the `delete_default_library_path` and `set_default_library_paths` commands, in the OpenVOS System Administration: Administering and Customizing a System (R281), provide additional information about a module’s default directories for libraries.
**detach_default_output**

**Purpose**

This command detaches your `default_output` port and reattaches it to its previous attachment.

**Display Form**

```
detach_default_output
```

No arguments required. Press ENTER to continue.

**Command Line Form**

```
detach_default_output
```

**Explanation**

The `detach_default_output` command saves the identity of the attachments, in sequential order, of the `default_output` port so that `detach_default_output` can reattach the port to its previous attachment. If no previous attachment exists, the operating system disregards a `detach_default_output` command, since it cannot reattach the port to an object that does not exist.

**Examples**

Suppose the `default_output` port for your process is attached to your terminal. Consider the following sequence of commands.

```
attach_default_output table_of_contents
list -all
detach_default_output
```

These commands attach the `default_output` port to a file named `table_of_contents`, list the current directory, and then reattach the port to your terminal. The file `table_of_contents` now contains listings of all the files, subdirectories, and links in the current directory.

**Related Information**

See the description of the `attach_default_output` command for information about how to change the default attachment of the `default_output` port. See also the descriptions of the `attach_port`, `detach_port`, `start_logging`, and `stop_logging` commands.
**detach_port**

**Purpose**
This command detaches an attached port from a file or I/O device.

**Display Form**

```
--------------- detach_port ---------------
port_name: [port name]
```

**Command Line Form**

```
detach_port port_name
```

**Arguments**

- **port_name**
  - Required
  - The name of the port to be detached.

**Explanation**

The `detach_port` command **cannot** be used to detach your process’s `default_input`, `terminal_output`, `command_input`, `default_output`, or `terminal` port, or your process’s unique string port to the system message file.

**Examples**

To detach the port `this_week` from the file or I/O device to which it is attached, use this command.

```
detach_port this_week
```

**Related Information**

See the description of the `attach_port` command for information on how to attach a port to a file or I/O device. See also the descriptions of the `attach_default_output`, `detach_default_output`, and `list_devices` commands.
dismount_tape

Purpose

This command dismounts a tape mounted on the specified tape drive or the tape drive to which the specified port is attached.

Display Form

```
-------------------------------- dismount_tape --------------------------------
tape_device_or_port_name:    [message]
-message:                  [message]
-unload:                   yes
```

Command Line Form

```
dismount_tape tape_device_or_port_name
   [-message message]
   [-no_unload]
```

Arguments

- **tape_device_or_port_name** 
  The name of the tape device, or the name of the port attached to the tape drive, holding the tape to be dismounted.

- **-message message**
  Displays a message on an operator’s terminal or on your terminal. The message is added to the standard message about dismounting the tape. By default, the default message defined by the `set_tape_drive_params` command is displayed.

- **-no_unload**
  Dismounts the tape without unloading it. By default, once a tape is mounted with `mount_tape`, it remains mounted until you dismount it with `dismount_tape`.

Explanation

The `dismount_tape` command dismounts a tape volume from a tape drive, unloads the tape by default, and sends a message to the operator.

If you explicitly mount a tape with `mount_tape`, you must explicitly dismount the tape with `dismount_tape`. If `mount_tape` implicitly attaches a port, `dismount_tape` implicitly detaches the port. The `dismount_tape` command unloads the tape unless you specify `-no_unload`. 
If `write_tape`, `read_tape`, or `list_tape` implicitly attaches a port and implicitly mounts a tape, that is, with no prompts, the command implicitly dismounts the tape and detaches the port. Likewise, if `save_object`, `restore_object`, or `list_save_tape` implicitly attaches a port and automatically mounts a tape, that is, with the prompts for `mount_tape`, the command implicitly detaches the port and unloads the tape. In these cases, you do not need to use the `dismount_tape` or `detach_port` commands.

However, if a command that automatically mounts a tape does so after you attach a port with `attach_port`, you must dismount the tape with either `dismount_tape` or `detach_port`. The `dismount_tape` command dismounts the tape and unloads the tape, unless you specify `-no_unload`. The `detach_port` command detaches the port, forcing the tape to be unloaded. For more information, see Explanation in the `mount_tape` command.

**Related Information**

See also the command descriptions of `copy_tape`, `create_tape_volumes`, `display_tape_params`, `dump_tape`, `list_save_tape`, `mount_tape`, `position_tape`, `read_tape`, `restore_object`, `set_second_tape`, `set_tape_drive_params`, `set_tape_mount_params`, `set_tape_file_params`, `verify_save`, and `write_tape`. 
display

Purpose
This command writes a file to the output device or file attached to your default_output port, which is usually your terminal, or to the device or file specified in the command.

Display Form

```
file_names:            yes
-caseless:            no
-index:               no
-match:               no
-output_path:         no
-first_line:          no
-last_line:           no
-line_numbers:        no
-header:              yes
-raw:                 no
-line_length:         default
-slave_printer:       no
-slave_page_length:   60
-min_lines:           1
-interpret_tabs:      no
-file_separator:      lf
-match_status:        no
```
Command Line Form

```
display file_names...
   [-no_caseless]
   [-index index_name]
   [-match character_string]
   [-output_path output_path_name]
       [-slave_printer]
   [-first_line first_line_number]
   [-last_line last_line_number]
   [-line_numbers]
   [-no_header]
   [-line_length]
   [-raw]
   [-interpret_tabs start_column, spacing]
   [-slave_page_length number_of_lines]
   [-min_lines number_of_lines]
   [-file_separator]
   [-match_status]
```

Arguments

- **file_names**: One or more names or star names of files to be displayed. You can specify any type of file. For IBM® Document Content Architecture (DCA) files and WordPerfect® files, use the `display_file` command, instead.

- **-no_caseless**: Specifies that when you select `-match`, the specified match of `character_string` is case sensitive. If you do not use `-no_caseless` and use `-match`, the matching disregards the case of `character_string`. If you use `-no_caseless` without also using `-match`, the command disregards this argument.

- **-index index_name**: Specifies an index to control the order in which records in the file are displayed. If you use `-index`, you can give only one file path name, and `index_name` must be an index to that file.

- **-match character_string**: Displays only the lines in the files that contain the character string `character_string`. The command disregards the case of the alphabetical characters. If you specify `-no_caseless`, the match is case sensitive. By default, the operating system displays all the lines in the range defined by `-first_line` and `-last_line`.

- **-output_path output_path_name**: Directs the output to the output device or file `output_path_name`. By default, the command directs the output to your default output port. You cannot specify both `-output_path` and `-slave_printer` in the same command.
-first_line first_line_number
Displays each specified file beginning at the line numbered first_line_number. If there are fewer than first_line_number records in the file, the command displays an error message informing you of that, and returns you to command level. By default, the display begins at the first line in each file.

-last_line last_line_number
Displays each specified file through the line numbered last_line_number. By default, the display is through the end of the file.

-line_numbers
Includes line numbers in the display. By default, the display omits the line numbers.

-no_header
Suppresses the display of the names of the specified files. By default, the command displays the file name of each specified file just before its contents.

-raw
Displays the files without interpreting embedded word processor controls. Control sequences that do not normally appear on the screen are replaced with the ASCII digits representing the hexadecimal value of the bytes. You cannot specify -raw and -interpret_tabs in the same command.

-line_length
Specifies the number of columns displayed before the line wraps. For terminal devices, the default line length is the width of the output device; for nonterminal output devices (for example, output to a file), the default line length is 132 columns. The default line length for slave printers and for terminals is the same (generally, 80 columns).

-slave_printer
Directs the output from the command to a printer attached to the terminal. If you do not use -slave_printer or -output_path, the command directs the output to the default_output port. You cannot specify both -slave_printer and -output_path in the same command.

-slave_page_length number_of_lines
Prints a specified number of lines per page of output, if you also specify -slave_printer. The minimum number of lines you can specify is 10. By default, the value is 60. If you specify -slave_page_length and do not specify -slave_printer, the command disregards this argument.

-min_lines number_of_lines
Specifies the minimum number of lines to be displayed following each line containing the characters specified in -match. If you specify a value of n for number_of_lines, the command displays the line containing the specified string and the n-1 subsequent lines, if any. By default, the value of number_of_lines is 1.

-interpret_tabs start_column, spacing
Interprets occurrences of the ASCII tab character. You must give the column number start_column of the first tab stop and the number spacing of positions between tab stops. A comma must separate the two numbers. You cannot specify both -interpret_tabs and -raw in the same command.
**display**

- **-file_separator**
  Controls the appearance of the output when the -output_path argument is used. When using the -output_path argument, you can separate the output from multiple files by specifying the ff (form-feed) value. By default, the -file_separator argument uses the lf (line-feed) value.

- **-match_status**
  If you specify this argument with the -match argument, the command sets the (command_status) command function to one of the following:
  
  - e$no_match(7857) if the command finds no matches
  - 0 if the command finds a match

  This argument has no effect unless you also specify the -match argument.

**Explanation**

The display command displays any type of file, with the following exceptions:

- Pipes, message queues, and server queues. If you specify only one file and it is a pipe, message queue, or server queue, the command returns an error. If you specify more than one file, either explicitly or via a star name, the command does not display any file that is a pipe, message queue, or server queue, and the command status is set to zero. If you specify -match in this case, files that cannot be displayed are treated as if they contain no matches.

- IBM DCA files and WordPerfect files. Use the display_file command to display these files.

The display command can display stream files that have been opened using region locking (including files that are open and in use by POSIX programs).

If you specify the argument -index, the displayed records are sorted using the index index_name. You can specify this argument only when displaying a single file.

The -match argument allows you to display only the lines containing the string character_string. If character_string contains spaces, you must enclose the string in apostrophes. This argument is convenient for displaying only the portions of a file that contain a particular string and for identifying all files that contain it.

With the -output_path argument, you can direct the output to a file or device different from the current attachment of your default_output port. If output_path_name is an existing file, the display command truncates the file before writing to it. If -output_path is not an existing file, the command creates a sequential file with that name and writes the output to it. If you specify this argument and specify more than one file, the command appends the output from each file to the output from the preceding files.

The -file_separator lf argument writes a blank line after each file if the display command produces any output while processing a file. For example, if any characters in a line matched the -match character_string argument, then a blank line is written after all of the matching lines have been displayed. If multiple files are specified and the ff character is
used in the -file_separator argument, then a form-feed character is written to separate
the output.

The display command opens an input file for dirty input if the input file is locked using
record locking or if it is exclusively locked by another process and you do not specify the
-index argument. In these situations, the command displays the following messages:

For exclusive locking:

    display:  File is in use; using dirty input mode.
               Some data may not be visible yet. object_name

For record locking:

    display:  File is in use; using dirty input mode.
               Some records may not be visible yet. object_name

If a file is already locked using region locking, the command attempts to open the file only if
you did not specify the -index argument.

The -slave_printer argument is supported for both TeleVideo-style (V101, V102, V103)
and ANSI-style (VT100, VT220, VT320, CIT482, V105, V109) terminals. In addition,
OpenVOS supports this argument on any terminal whose terminal-type definition (.ttp) file
defines the sequences aux-printer-on and aux-printer-off. The following standard
ANSI .ttp files define the aux-printer-x output sequences: cit482, vt100, vt220,

Access Requirements
You need read access to a file to display it.

Examples

Example 1.
The following command displays the path name and contents of the file this_week in the
current directory.

    display this_week -index date_changed

The lines in the file are displayed in the order determined by the index date_changed.

Example 2.
To display the lines containing the character string city-code in all OpenVOS COBOL
source modules in your current directory, use this command.

    display *.cobol -match city-code -line_numbers

This command also displays the path names of the specified files and the line numbers of the
lines that contain the string city-code.

Related Information
See also the description of the display_file command.
Purpose
This command displays a user’s access to files, directories, and devices.

Display Form

Command Line Form

display_access path_names...
  [-user user_name]
  [-all]

Arguments

- path_names Required
  One or more names or star names of files, directories, and device access lists. The path name for device access lists is /system/acl/access_list_name. The command displays the user’s access to each matching file, directory, or device access list. By default, the command uses your current directory.

- -user user_name
  Specifies a user name whose access is to be displayed. By default, the command uses your user name.

- -all CYCLE
  Displays the access of all users with explicitly defined access to each file, directory, or device. If you use -all and -user in the same command, the command disregards -user. By default, the command uses your user name.

Explanation
The display_access command displays your access, the access of another user, or the access of all users to each file, directory, and device you specify in the path_names argument.

To determine the access to a file, the display_access command uses the access control list of the file and the default access control list of its containing directory. To determine the
access to a directory, the command uses the access control list of the directory. To determine the access to a device, the command uses the access control list of the device.

The `path_names` argument for device access lists contains the file name `access_list_name`. This file name matches the value for the `access_list_name` field for each device in a set in the `devices.tin` file. The `access_list_name` file contains the access control list (ACL) for a device or set of devices. The operating system creates the ACL from the `devices.tin` file. A user can then display access to the devices by issuing the `display_access` command. If the `access_list_name` field for a device does not have a value, no ACL is created, and all users can access that device.

**Access Requirements**

To see the access to an object, you need status access to the directory containing it.

**Examples**

Example 1.
The following command displays your access to the OpenVOS COBOL source modules in `>Sales>Jones>reports`.

```
    display_access >Sales>Jones>reports>*.cobol
```

Example 2.
To display the access of all users to all objects in the current directory, use this command.

```
    display_access * -all
```

The output of this command might look like the following example, for each object in the current directory.

```
%s1#d02>Marketing>Smith>weekly_report

    r  Backup.Backup (D)
    w  Clark.Marketing
    w  Smith.* (D)
    w  *.SysAdmin (D)
    w  *.System (D)
    r  *.Marketing (D)
    r  *.Services (D)
    r  *.Strategy (D)
    r  *.Operator (D)
    n  *.* (D)
```

The command derives the access from the access control list of the file and the default access control list of the directory `%s1#d02>Marketing>Smith`. An entry in the access control list of a file takes precedence over an entry in the default access control list for the same user name.

The symbol (D) indicates that this term comes from the default access control list of the directory.
Example 3.
The following command displays the access of all users of the device $s1#d02$, which has the access list name of `exclude_service_acl`.

```bash
display_access $s1#d02>system>acl>exclude_service_acl -all
```

The output of this command might look like the following example.

```bash
%s1#d02>system>acl>exclude_service_acl
w  *.Installer
w  *.SysAdmin
w  *.System
r  *.Marketing
n  *.Services
r  *.*
```

Related Information

For more information about access, see the command descriptions of `display_access_list`, `display_default_access_list`, `give_default_access`, `propagate_access`, `remove_access`, and `remove_default_access`. For a detailed discussion of access, see *OpenVOS Commands User’s Guide* (R089) and *OpenVOS System Administration: Registration and Security* (R283).
display_access_list

Purpose
This command displays the access control lists of files, directories, and devices.

Display Form

Path_names: display_access_list

Command Line Form

display_access_list path_names...

Arguments

- path_names
  
  Required
  
  One or more names or star names of files, directories, and device access lists. The path name for device access lists is >system>acl>access_list_name. The command displays the access control list of each matching file, directory, or device access list. By default, the command uses your current directory.

Explanation

The display_access_list command displays the access control lists of the files, directories, and devices you specify in the path_names argument.

The path_names argument for device access lists contains the file name access_list_name. This file name matches the value for the access_list_name field for each device in a set in the devices.tin file. The access_list_name file contains the access control list (ACL) for a device or set of devices. The operating system creates the ACL from the devices.tin file. A user can then display the ACL of devices by issuing the display_access_list command. If the access_list_name field for a device does not have a value, no ACL is created, and all users can access that device.

Access Requirements

To display the access control list of a file, directory, or device, you need status access to the containing directory.
Examples
The following command displays the access control lists of all OpenVOS PL/I and OpenVOS COBOL source modules in the current directory.

   display_access_list *.pl1 *.cobol

Related Information
For more information about access control, see the command descriptions of display_access, display_default_access_list, give_default_access, propagate_access, remove_access, and remove_default_access.
**Purpose**

This command displays the status of the batch queue on a specified module.

**Display Form**

```
- module: [current_module]
```

**Command Line Form**

```
display_batch_status [-module module_names]
```

**Arguments**

- `-module module_names`
  One or more names or star names of modules, for which you want batch queue information. The default is the current module.

**Explanation**

The `display_batch_status` command displays information about the status of batch jobs in the queues for a module. The following information is provided:

- the name of the queue or queues in the module
- the state of the queue, which may be running, suspended, or stopped
- the maximum number of users allowed to have batch jobs in the queue at the same time (a limit set by `batch_admin`)
- the number of jobs running for each user
**display_batch_status**

**Examples**

The following example illustrates the output for the `display_batch_status` command.

Batch queues for %s1#d02

<table>
<thead>
<tr>
<th>QUEUE</th>
<th>STATE</th>
<th>MAX</th>
<th>RUNNING JOBS</th>
</tr>
</thead>
<tbody>
<tr>
<td>r1b</td>
<td>run</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>bk (m2)</td>
<td>run</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>normal (m2)</td>
<td>run</td>
<td>1</td>
<td>02 Smith (weekly_receivables)</td>
</tr>
<tr>
<td>normal (m2)</td>
<td>run</td>
<td>1</td>
<td>02 Clark (pass3)</td>
</tr>
</tbody>
</table>

**Related Information**

See the command descriptions of `batch`, `cancel_batch_requests`, `cancel_device_reservation`, `reserve_device`, `list_batch_requests`, `move_device_reservation`, and `update_batch_requests`. 
display_current_dir

Purpose

This command displays the full path name of your current directory.

Display Form

```
----------------------------- display_current_dir -----------------------------
No arguments required. Press ENTER to continue.
```

Command Line Form

```
display_current_dir
```

Explanation

The `display_current_dir` command displays the full path name of your current directory.

Related Information

For more information about directory management, see the command descriptions of `change_current_dir`, `compare_dirs`, `copy_dir`, `create_dir`, `delete_dir`, and `move_dir`. 
display_current_module

Purpose

This command displays the full name of the processing module running your process.

Display Form

--- display_current_module ---
No arguments required. Press ENTER to continue.

Command Line Form

display_current_module

Explanation

The `display_current_module` command displays the full name of your current module. The current module is the processing module executing your process.

The full name of a module is a percent sign followed by a system name, then a number sign, then a module name. For example, `%s1#m2` is the full module name of the module m2.
**display_date_time**

**Purpose**
This command displays the current date and time.

**Display Form**

```
----------------------------- display_date_time -----------------------------
-<CYCLE>-long: 0
```

**Command Line Form**

`display_date_time [-long]`

**Arguments**

- `-long`
  Displays the long form of the date and time. By default, the command displays a short form.

**Explanation**

The `display_date_time` command displays the current date and time on your terminal or the current attachment of `default_output`.

The short form of the date and time is as follows:

`yy-mm-dd hh:mm:ss zzz`

In the date part:

- `yy` is a two-digit code for the year (90 is 1990)
- `mm` is a two-digit code for the month (06 is June)
- `dd` is a two-digit code for the day of the month
display_date_time

In the time part:

\( hh \) is the hour of the day according to a 24-hour clock (14 is 2:00 P.M.)
\( mm \) is the minute of the hour
\( ss \) is the second of the minute
\( zzz \) is a three-character code for the time zone as defined by the user or the system

The long form of the date and time is as follows:

\( day-name, month \ day, \ year \ time \)

In this form:

\( day-name \) is the name of the day of the week
\( month \) is the name of the month
\( day \) is the number of the day of the month
\( year \) is the year
\( time \) is the time of day according to a 12-hour clock

Examples

Example 1.
The following command displays the current date and time in the short form.

\[ \text{display_date_time} \]

This display might appear.

\[ 90-06-15 14:31:21 \text{ est} \]

Example 2.
This command displays the current date and time in the long form.

\[ \text{display_date_time -long} \]

If the current date and time are the same as in the previous example, the following display appears.

\[ \text{Tuesday, June 15, 1990 2:31 pm} \]

Related Information

See also the description of the set_time_zone command and the (date_time) command function.
display_default_access_list

Purpose

This command displays the default access control lists of directories.

Display Form

```
------------------------- display_default_access_list -------------------------
directory_names: current_dir
```

Command Line Form

```
display_default_access_list [directory_names] ...
```

Arguments

- `directory_names`
  One or more names or star names of directories. The command displays the default access control list of each matching directory. By default, the command displays the default access control list of your current directory.

Explanation

The `display_default_access_list` command displays the default access control list of a directory which has the same form as the access control list of a directory or file. It is of user-name access-type pairs. However, unlike access control lists, which are associated with both directories and files, default access control lists are associated only with directories. Furthermore, the access types in a default access control list are access types to files.

A default access control list specifies the access of users to files in the directory when the users are not covered by the files’ access control lists. When checking a user’s access to a file, the operating system first searches the file’s access control list for a matching entry. If the user name does not match any entry in the access control list, the operating system searches the default access control list of the containing directory. If the user name does not match any entry in either list, the user has undefined access to the file. Undefined access is equivalent to null access.

When you create a directory, the operating system gives it the default access control list of its containing directory.
**Access Requirements**

To display the default access control list of a directory, you need status access to the directory.

**Related Information**

See also the command descriptions of `display_access`, `display_access_list`, `give_default_access`, `propagate_access`, `remove_access`, and `remove_default_access`. For a detailed discussion of access, see *OpenVOS Commands User’s Guide* (R089) and *OpenVOS System Administration: Registration and Security* (R283).
display_default_open_options

Purpose
This command displays the default open options for a directory.

Display Form

```
----------------------------- display_default_open_options -----------------------------
directory_names: directory_names...
-brief: no
```

Command Line Form

```
display_default_open_options directory_names...
    [-brief]
```

Arguments

- `directory_names` Required
  One or more names or star names of directories for which default open options are to be displayed.

- `brief` CYCLE
  Displays the default open options for only those directories that do not have standard default open options. By default (the value `no`), the command displays the default open options for all specified path names.

Explanation

This command displays the default open options for all directories that match the specified star names. Files that are located in the directory, or that are created in that directory in the future, will inherit the directory’s default open options unless you explicitly set open options for the file or its indexes. A newly created directory inherits its default open options from its parent directory.

If you set the `-brief` argument to `no` (the default), the command displays each of the path names that the command examines only if the default open options are not default values. The command displays the default open options as canonical strings (these values are also used in the `display_dir_status` command).

For more information about the open options, see the manual OpenVOS System Administration: Administering and Customizing a System (R281). See also the description of the `s$get_default_open_options` subroutine in the OpenVOS Subroutines manuals.
Related Information

See also the descriptions of the `display_open_options`, `set_default_open_options`, and `set_open_options` commands.
**display_device_info**

**Purpose**

This command displays information about a specified device.

**Display Form**

```
----------- display_device_info -----------
device_name: [device_name]
-brief: [no]
```

**Command Line Form**

```
display_device_info device_name
   [-brief]
```

**Arguments**

- **device_name** 
  Required
  The path name of a device. The device can be defined on any module.

- **-brief** 
  CYCLE
  Displays a subset of the information about the specified device. By default, the `display_device_info` command displays full information about the specified device.

**Explanation**

The `display_device_info` command displays the following information about a specified device. (Use the `list_devices` command to determine the `device_name`.)

- **type**—The device type
- **module name**—The name of the module to which the device is connected
- **channel number**—The channel number associated with the device
- **event id**—The event ID associated with the device
- **clone limit**—The maximum number of clone devices that can be created from one clone device. A non-zero value indicates that the device is a clone device; a zero value indicates that the device is not a clone device. If `clone limit` has a zero value and `cloned from` has a non-zero value, the device was cloned from a clone device.
- **clone count**—The current number of devices cloned from the clone device
display_device_info

- **cloned from**—Helps generate the suffix used in clone device names. A non-zero value indicates that the device is a clone device; a zero value indicates that the device is not a clone device. If cloned from has a non-zero value and clone limit has a zero value, the device was cloned from a clone device.

- **device type name**—The device type. If no device driver is loaded, device type name displays the requested device type.

- **streams device**—The device is a STREAMS-based device

- **streams minor number**—A unique identifier for STREAMS devices that are not cloneable

- **streams remote minor number**—Reserved for future use

- **streams remote clone limit**—Reserved for future use

- **streams driver**—The name of the STREAMS driver

- **hardware address**—The hardware address of the device. If the device is not physical, the value is NULL. See the manual *OpenVOS System Administration: Configuring a System* (R287) for more information about this field.

- **Device Flags**—Indicates the following:
  - **local**—Whether the device is located on the module that is executing the `display_device_info` command.
  - **reserved**—Whether the device is reserved for use by a specific process
  - **shareable**—Whether the device can be shared by multiple processes
  - **configured**—Displayed only for compatibility with previous VOS releases
  - **dial out**—Whether the channel can originate outbound telephone calls
  - **login slave**—Whether the device is a login device
  - **force listen**—Whether the channel has been configured to ignore dataset leads. Stratus does not recommend the use of this option.
  - **privileged terminal**—Whether a privileged process can log in on the terminal
  - **ebcdic**—Whether the character code is ASCII or EBCDIC
  - **dial up**—Whether the channel can be externally dialed-up
  - **half duplex**—Whether the line is full-duplex or half-duplex
  - **multiplexed**—Whether the device is multiplexed
  - **sub device**—Whether the device is a subdevice
- spooler—Whether the device is a printer
- alternate speed—Whether the device supports alternate baud rates

- reference count—The reference count indicating the number of processes attached to the device
- person name—The person name of the current user
- primary person name—The person name of the current primary user (displayed only for vterm devices)
- process id—The process identifier of the login process
- time assigned—The time at which the device was assigned to the process
- baud—The initial baud rate for the device
- stop bits—The number of stop bits associated with each character
- parity—The parity assumed for the channel
- terminal type name—The model name of the terminal or printer attached to the device
- parent device name—The primary device of the subdevice
- mpx number—The control unit address for the subdevice
- sub device number—The device address of the subdevice
- firmware type—Displayed only for compatibility with previous VOS releases
- parameters—Any device-specific parameters
- More Users—Provides information about additional users
- Secondary Users—Provides information about additional users (displayed only for vterm devices)
Examples

Example 1.
Device information for a window term device

display_device_info %sys2#tli_log.m105_1

Device Information for %sys2#tli_log.m105_1

type:                         window_term
module name:                  %sys2#m105
channel number:               0
event id:                     2
clone limit:                  0
clone count:                  2
cloned from:                  27
device type name:             window_term
streams device:               0
hardware address:             NULL

Device Flags:
local:                        1
reserved:                     0
shareable:                    1
configured:                   1
dial out:                     0
login slave:                  1
force listen:                 0
privileged terminal:          1
ebcdic:                       0
dial up:                      0
half duplex:                  0
multiplexed:                  0
sub device:                   0
spooler:                      0
alternate speed:              0

reference count:              10
person name:                  Rogerson
process id:                   0719805A
time assigned:                04-03-01 19:54:02 est
baud:                         ext
stop bits:                    0?
parity:                       even
terminal type name:           ascii
parent device name:           
mpx number:                   0
sub device number:            0
firmware type:                default
parameters:                   -access_layer tli_al

(Continued on next page)
(Continued)

More Users:

person name: Rogerson  
process id: 07178040  
time assigned: 04-03-01 20:25:59 est

person name: Rogerson  
process id: 07178040 (4 attachments)  
time assigned: 04-03-01 20:25:48 est

person name: Rogerson  
process id: 0719805A  
time assigned: 04-03-01 19:54:15 est

person name: Rogerson  
process id: 0719805A (3 attachments)  
time assigned: 04-03-01 19:54:03 est

Example 2.
Device information for a STREAMS device

display_device_info #enet.m105.11.3 -brief

Device Information for %sys2#enet.m105.11.3

type: streams_pci  
module name: %sys2#m105  
slot number: 11  
event id: 3  
clone limit: 32  
clone count: 0  
cloned from: 6  
streams driver: genet  
hardware address: 11/3  
Device Flags: streams device, local, shareable, +configured  
reference count: 0  
parameters: -partner #enet.m105.10.3 -sdlmux +#sdlmuxA.m105.10.

Note: In the preceding example, the plus-sign character (+) indicates that the line has wrapped.
**Example 3.**
Device information for a tape device

```bash
display_device_info %sys2#tape.fibre
```

Device Information for %sys2#tape.fibre

- **type:** tape
- **module name:** %sys2#m103
- **slot number:** 10
- **channel number:** 0
- **event id:** 2
- **clone limit:** 0
- **clone count:** 0
- **cloned from:** 0
- **device type name:** tape
- **streams device:** 0
- **hardware address:** 10/2/1/0/1

**Device Flags:**
- **local:** 1
- **reserved:** 0
- **shareable:** 0
- **configured:** 1
- **dial out:** 0
- **login slave:** 0
- **force listen:** 0
- **privileged terminal:** 0
- **ebcdic:** 0
- **dial up:** 0
- **half duplex:** 0
- **multiplexed:** 0
- **sub device:** 0
- **spooler:** 0
- **alternate speed:** 0
- **reference count:** 0
- **baud:** ext
- **stop bits:** 0?
- **parity:** even
- **terminal type name:**
- **parent device name:**
- **mpx number:** 0
- **sub device number:** 0
- **firmware type:** default
- **parameters:**
Example 4.
Device information for a vterm device

display_device_info vterm1.1

Device Information for %sys2#vterm1.1

type: vterm
module name: %sys2#m100
channel number: 1
event id: 2
clone limit: 0
clone count: 0
cloned from: 0
device type name: vterm
streams device: 0
hardware address: NULL

Device Flags:
local: 1
reserved: 0
shareable: 1
configured: 1
dial out: 0
login slave: 1
force listen: 0
privileged terminal: 0
ebcdic: 0
dial up: 1
half duplex: 0
multiplexed: 0
sub device: 0
spooler: 0
alternate speed: 0
reference count: 3
primary person name: Rogerson
process id: 07148088
time assigned: 04-02-25 14:17:23 est
baud: 9600
stop bits: 0
parity: space
terminal type name: v103
parent device name:
mpx number: 0
sub device number: 0
firmware type: default
parameters:

(Continued on next page)
Secondary Users:
person name:                  Rogerson
process id:                   07148088
time assigned:                04-02-25 14:17:31 est

person name:                  Rogerson
process id:                   07148088
time assigned:                04-02-25 14:17:27 est

Related Information
For a complete description of the parameters displayed, see the manual *OpenVOS System Administration: Configuring a System* (R287). See also the description of the *list_devices* command.
display_dir_status

Purpose
This command displays information about a specified directory.

Display Form

```
--- display_dir_status ---
directory_names: ...
-show_limits: no
```

Command Line Form

```
display_dir_status directory_names...
   [-show_limits]
```

Arguments

- `directory_names` **Required**
  One or more names or star names of directories. The command displays information about all directories with matching names. By default, the command uses your current directory.

- `-show_limits`  
  Provides information about a directory’s entry and block limits as well as the current number of entries for expandable directories.

Explanation

The `display_dir_status` command displays the following information about the specified directory:

- the full path name
- the date and time the directory was last used
- the date and time the directory was last modified
- the date and time the directory was last saved
- the date and time the directory was created
- the number of disk blocks used to store the directory
- the access mode of the user for the directory
- the author of the directory
- the default open options
- the author of a directory is the user who created the directory
- the entry limits setting, if the directory is expandable
**display_dir_status**

- the values of block limit, if the limits are non-default
- the value of limit-related data

For more information about the open options, see the description of the **display_open_options** command.

**Access Requirements**

To obtain information about a directory, you need status or modify access to it.

**Examples**

The following command displays the status of the directory *Sales* that is a subdirectory of the current directory.

```
display_dir_status Sales -show_limits
```

This display might result.

```
name:               %s1#d02>Sales>east>Jones
last used at: 14-06-15 14:31:40 est
last modified at: 14-06-15 14:31:40 est
last saved at: never
time created: 14-06-15 14:31:40 est
blocks used: 6
mode:               m
author:             Smith.SysAdmin
entries allowed 32700
current entries: 25142
blocks allowed: 8720 (default 500)
current blocks: 8720
```

**Related Information**

See also the command descriptions of **compare_dirs, create_dir, delete_dir, display_file_status, display_open_options**, and **move_dir**.
display_disk_info

Purpose
This command displays information about the disks in a specified module.

Display Form

```
<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>disk_name:</td>
<td>The logical volume of which the</td>
</tr>
<tr>
<td>member:</td>
<td>disk is a member. The command</td>
</tr>
<tr>
<td>partner:</td>
<td>displays information about the</td>
</tr>
<tr>
<td>-module:</td>
<td>specified logical volume in the</td>
</tr>
<tr>
<td>-long:</td>
<td>current module. You cannot</td>
</tr>
<tr>
<td>-partitioned:</td>
<td>specify both disk_name and -module. By default, the command displays information about all of the member disks in the specified modules.</td>
</tr>
<tr>
<td>-show_non_native:</td>
<td>The command displays information about the disks in a specified module.</td>
</tr>
</tbody>
</table>
```

Command Line Form

```
display_disk_info [ disk_name ] ... [ -module module_name ] ...

[ member ]
[ partner ]
[ -module module_name ]
[ -long ]
[ -partitioned ]
[ -show_non_native ]
```

Arguments

- **disk_name**
  The logical volume of which the disk is a member. The command displays information about the specified logical volume in the current module. You cannot specify both disk_name and -module. By default, the command displays information about all of the member disks in the specified modules.

- **member**
  A number, between 0 and 9 inclusive, that refers to the member disk’s location in the logical volume. By default, the command displays disk information about all members of the specified logical volume.

- **partner**
  The partner of the member disk or disks specified by member. The possible values are primary, secondary, and either one (blank). The default setting is either one. If you
do not select either the primary or the secondary disk, the computer selects whichever one is available. If both disks are available, the computer selects the primary disk. Use the default setting if it does not matter which partner of the duplex disk is used, or if you are displaying the label of a nonduplex disk.

- **-module module_name**
  Specifies a module name or star name. You cannot specify both disk_name and -module. By default, the command displays information about all of the member disks in the specified modules.

- **-long**
  Displays a long report on the disks, including the size of the log partition file. By default, the command displays a short report, and does not report the size of the log partition file.

- **-partitioned**
  Displays partition-related information for a disk. By default, the command does not display this information.

- **-show_non_native**
  Displays information about any non-native file system devices (for example, iso9660 file system devices such as DVDs). By default, the command does not display this information.

  **Note:** Open StrataLINK does not support non-native file system devices. Because Open StrataLINK considers them to be devices for local use only, the -show_non_native argument has no effect if you specify it with the -module argument.

**Explanation**

The display_disk_info command displays the following information for each disk, specified in alphabetical order by disk name.

- the name of the module containing the disk
- the total number of disk blocks in the file partition, if the disk is mounted
- the number of disk blocks being used in the file partition, if the disk is mounted
- the number and percentage of disk blocks remaining in the file partition, if the disk is mounted

The system level I/O subprograms file_io, cache_manager, and tp_overseer reserve blocks for their use. These reserved blocks are counted as free by display_disk_info. Commands and subroutines that perform I/O call these subprograms, so you may get out-of-disk errors, even though display_disk_info shows as many as 200 to 300 free blocks.

If you specify the -long argument, the command displays, in alphabetical order by disk name, the following information:

- the total, used, and free disk block information for the file, paging, and log partitions for each disk (if the disk is mounted)
• the disk type
• the number of disk reads or writes since the last time the system was booted
• the non-default per-volume attributes related to optimization and allocation (see
  OpenVOS System Administration: Administering and Customizing a System (R281) for
  more information about per-volume attributes)

Note: If your system administrator issued the add_paging_file command (instead
of the update_disk_label command) to add paging space to your module’s disks,
this space is not shown in the output of the display_disk_info command. If you
want to see the actual paging space in use for a module, issue the
display_paging_usage command. For more information on the
add_paging_file and display_paging_usage commands, see the manual
OpenVOS System Administration: Registration and Security (R283).

The display_disk_info command displays information about non-native file system
devices in the following situations:

• if you specify the -show_non_native argument

• if you issue the command for a specific non-native disk, regardless of whether you
  specify -show_non_native or not

Note that you cannot specify the disk_name argument with the -module argument, nor can
you specify the -show_non_native argument with the -module argument.

Examples

Example 1.
Sample output of the display_disk_info -long command for %s1#m2 follows.

Module %s1#m2

%s1#dsa9
Partition Size Used Free Left
File 14999960 6839887 8160073 54.40%
Member Pri Sec Attributes
0 10/9/0/3/3 RAID,nonduplex,non-verify,serial
Volume Reads Writes Attributes
471 3 mounted

%s1#m2_mas
Partition Size Used Free Left
Paging 120000 7425 112575 93.81%
File 14879864 13305525 1574339 10.58%
Member Pri Sec Attributes
0 10/9/0/1/0 10/9/0/3/0 RAID.duplex,verify,serial
Volume Reads Writes Attributes
18582 7176 mounted

(Continued on next page)
### display_disk_info

#### %s1#raid0-1lun

<table>
<thead>
<tr>
<th>Partition</th>
<th>Size</th>
<th>Used</th>
<th>Free</th>
<th>Left</th>
</tr>
</thead>
<tbody>
<tr>
<td>File</td>
<td>63476568</td>
<td>11737092</td>
<td>51739476</td>
<td>81.50%</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Member</th>
<th>Pri</th>
<th>Sec</th>
<th>Attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>10/9/0/2/1</td>
<td>10/9/0/4/1</td>
<td>RAID, duplex, non-verify, serial</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Volume Reads</th>
<th>Writes</th>
<th>Attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>1822</td>
<td>2</td>
<td>mounted</td>
</tr>
</tbody>
</table>

#### %s1#raid1

<table>
<thead>
<tr>
<th>Partition</th>
<th>Size</th>
<th>Used</th>
<th>Free</th>
<th>Left</th>
</tr>
</thead>
<tbody>
<tr>
<td>Paging</td>
<td>120000</td>
<td>7399</td>
<td>112601</td>
<td>93.83%</td>
</tr>
<tr>
<td>File</td>
<td>14879960</td>
<td>10668386</td>
<td>4211574</td>
<td>28.30%</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Member</th>
<th>Pri</th>
<th>Sec</th>
<th>Attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>10/9/0/1/1</td>
<td>10/9/0/3/1</td>
<td>RAID, duplex, verify, serial</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Volume Reads</th>
<th>Writes</th>
<th>Attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>103486</td>
<td>109646187</td>
<td>mounted, throughput, access</td>
</tr>
</tbody>
</table>

#### %s1#raid10-1lun

<table>
<thead>
<tr>
<th>Partition</th>
<th>Size</th>
<th>Used</th>
<th>Free</th>
<th>Left</th>
</tr>
</thead>
<tbody>
<tr>
<td>File</td>
<td>63476568</td>
<td>12986434</td>
<td>50490134</td>
<td>79.54%</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Member</th>
<th>Pri</th>
<th>Sec</th>
<th>Attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>10/9/0/1/3</td>
<td>10/9/0/3/4</td>
<td>RAID, duplex, non-verify, serial</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Volume Reads</th>
<th>Writes</th>
<th>Attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>1822</td>
<td>2</td>
<td>mounted</td>
</tr>
</tbody>
</table>

#### %s1#raid2

<table>
<thead>
<tr>
<th>Partition</th>
<th>Size</th>
<th>Used</th>
<th>Free</th>
<th>Left</th>
</tr>
</thead>
<tbody>
<tr>
<td>Paging</td>
<td>120000</td>
<td>7353</td>
<td>112647</td>
<td>93.87%</td>
</tr>
<tr>
<td>File</td>
<td>14879960</td>
<td>7085273</td>
<td>7794687</td>
<td>52.38%</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Member</th>
<th>Pri</th>
<th>Sec</th>
<th>Attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>10/9/0/1/2</td>
<td>10/9/0/3/2</td>
<td>RAID, duplex, verify, serial</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Volume Reads</th>
<th>Writes</th>
<th>Attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>97413</td>
<td>105292762</td>
<td>mounted, access</td>
</tr>
</tbody>
</table>

#### %s1#raid3

<table>
<thead>
<tr>
<th>Partition</th>
<th>Size</th>
<th>Used</th>
<th>Free</th>
<th>Left</th>
</tr>
</thead>
<tbody>
<tr>
<td>File</td>
<td>30517560</td>
<td>2594317</td>
<td>27923243</td>
<td>91.49%</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Member</th>
<th>Pri</th>
<th>Sec</th>
<th>Attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>10/9/0/2/5</td>
<td>10/9/0/4/5</td>
<td>RAID, duplex, verify, serial</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Volume Reads</th>
<th>Writes</th>
<th>Attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>383426</td>
<td>355255175</td>
<td>mounted, response, access</td>
</tr>
</tbody>
</table>

#### %s1#raid4

<table>
<thead>
<tr>
<th>Partition</th>
<th>Size</th>
<th>Used</th>
<th>Free</th>
<th>Left</th>
</tr>
</thead>
<tbody>
<tr>
<td>File</td>
<td>31738264</td>
<td>31724147</td>
<td>14117</td>
<td>0.04%</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Member</th>
<th>Pri</th>
<th>Sec</th>
<th>Attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>10/9/0/2/6</td>
<td>10/9/0/4/6</td>
<td>RAID, duplex, verify, serial</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Volume Reads</th>
<th>Writes</th>
<th>Attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>852</td>
<td>2</td>
<td>mounted, throughput, access</td>
</tr>
</tbody>
</table>
Example 2.
If a system has several modules, you can give the -module argument with a module star name or an asterisk to retrieve information about disks on the system. If a disk volume exists but is not complete, the command lists that disk with the attribute member missing. The command display_disk_info -module * would produce output similar to this.

```
Module %s1#m2
Size     Used       Left
#d02  202576  194328  8248  4.0% duplex,verify,serial,mounted
#newdisk 0       0       0  0.0% member missing,not mounted
```

```
Module %s1#m3
Size     Used       Left
#d03    419264  409856  9408  2.2% duplex,verify,parallel,mounted
#d07    368584  364084  4500  1.2% duplex,verify,parallel,mounted
#d08    105344  100026  5318  5.0% duplex,verify,parallel,mounted
```

```
Module %s1#m4
Size     Used       Left
#d04    178680  121678  57002  31.9% duplex,verify,serial,mounted
```

```
Module %s1#m5
Size     Used       Left
#d05    174680  137587  37093  21.2% duplex,verify,parallel,mounted
#d06    367584  341509  26075  7.0% duplex,verify,serial,mounted
```

```
Module %s1#m6
Size     Used       Left
#d09    206576  179820  26756  12.9% duplex,non-verify,serial,mounted
#d10     77376   60504   16872  21.8% duplex,verify,serial,mounted
```

Example 3.
The following example uses the -partitioned argument to provide information about partitioned disks on the module.

```
Module %sys5#m16
%sys5#dd_target
Partition        Size      Used      Free      Left
File          14648368  362970    14285398  97.52%
Member Pri Sec Attributes
0  08/1/1/2/03 D913,nonduplex,non-verify,serial
     (partitioned: primary)       <-----
1  08/1/1/2/04 D913,nonduplex,non-verify,serial
     (partitioned: primary)       <-----
Volume Reads Writes Attributes
500   12 mounted
```

(Continued on next page)
display_disk_info

%sys5#m16_mas

Partition       Size      Used      Free      Left
Paging         6000      2648      3352      55.86%
File           1094648   895430    199218   18.19%
Member Pri     Sec      Attributes
                0  04/01/01/01 04/02/01/03  D705,duplex,verify,parallel
                (not partitioned)        <-----
Volume Reads   Writes   Attributes
              80781     31276     mounted

In the preceding example, the extra line of attributes indicates whether or not the partner disks in the member are partitioned. If neither member is partitioned, the command displays the following:

  (not partitioned)

If one partner is partitioned, the command displays the following:

  (partitioned: primary) OR
  (partitioned: secondary)

If both partners are partitioned, the command displays the following:

  (partitioned: primary, secondary)

Example 4.
The following example uses the -show_non_native argument to provide information about non-native file system devices.

display_disk_info -show_non_native

Module %sys1#m122

Size      Used       Left
%m122_mas 14849864  8794333  6055531  40.7% duplex,non-verify, serial,mounted
#titanium1 10986216  3020402  7965814  72.5% duplex,non-verify, parallel,mounted
#m122_dvd               0            0            0  0.0% ISO file system for DVD/CD
**display_disk_usage**

**Purpose**
This command displays the number of disk blocks occupied by a directory and all of its contents.

**Display Form**

```
------------------------ display_disk_usage ------------------------
directory_name: current_dir
-depth: 1
```

**Command Line Form**

```
display_disk_usage [directory_name]
    [-depth depth]
```

**Arguments**

- **directory_name**
  The path name of a directory. The command displays the number of disk blocks used by the directory and all its contents. It also displays the total number of blocks used by each directory in the hierarchy beneath `directory_name` down to a specified level of directories `depth` below `directory_name`. By default, the command displays the disk usage for your current directory and all of its contents.

- **-depth depth**
  Specifies the number of levels in the directory hierarchy about which the command displays information. By default, the command displays the number of disk blocks used by the directory `directory_name` and all its subdirectories.

**Explanation**

The `display_disk_usage` command displays the number of disk blocks occupied by a directory and all of its contents, the number of blocks occupied by its subdirectories and all their contents, and so forth down to a specified level in the hierarchy.

If your process is privileged, you can read the size of any directory, so that a privileged process always obtains an accurate report on disk usage. When your process is unprivileged, executing the `display_disk_usage` command updates the date-time-used attribute of the directory. If your process is privileged, the operating system does not update the date-time-used attribute.
Access Requirements

To display disk usage, you need status access to a specified directory and to all of its subdirectories. If you do not have status access to a directory, the `display_disk_usage` command reports that the directory occupies no disk blocks.

Examples

The following command displays the disk usage for the specified directory and two levels of its subdirectories.

```
  display_disk_usage >Sales>east>Jones -depth 2
```

This display might appear.

<table>
<thead>
<tr>
<th>BLOCKS USED</th>
<th>DIRECTORY</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>%s1#d02&gt;sales&gt;east&gt;jones&gt;reports&gt;weekly_old</td>
</tr>
<tr>
<td>17</td>
<td>%s1#d02&gt;sales&gt;east&gt;jones&gt;reports&gt;monthly_old</td>
</tr>
<tr>
<td>22</td>
<td>%s1#d02&gt;sales&gt;east&gt;jones&gt;reports</td>
</tr>
<tr>
<td>17</td>
<td>%s1#d02&gt;sales&gt;east&gt;jones&gt;customers</td>
</tr>
<tr>
<td>60</td>
<td>%s1#d02&gt;sales&gt;east&gt;jones</td>
</tr>
</tbody>
</table>
display_error

Purpose
This command displays the OpenVOS error message corresponding to a specified error code.

Display Form

```
-------------------------------- display_error --------------------------------
error_code:  [blank]
```

Command Line Form

```
display_error error_code
```

Arguments

- `error_code`  Required
  A standard status code name or number.

Explanation

The `display_error` command returns the status code name and message corresponding to the specified status code number if `error_code` is an OpenVOS status code number. If `error_code` is an OpenVOS status code name, the command returns the status code number and message that corresponds to the specified status code name. If `error_code` is not an existing status code number or status code name, the command displays the following message.

```
display_error: Invalid error code. error_code.
```

The `display_error` command displays markers in the message text that inform the user where the substitutions should be made. For example, if you display the error associated with `m$file_implicitly_locked_by`, this display appears.

```
Code 1024. Object is implicitly locked by &a1& on module &a2& executing &a3&.
```

All OpenVOS status code names begin with `e$, m$, q$, or r$, and all OpenVOS status code numbers are in the range 1001 to 14000. The range reserved for user-defined status code numbers is 16000 to 18000.
Examples

Example 1.
The following command displays the name and message associated with the specified error status code number.

\[ \text{display_error 1112} \]

This display appears.

\[ \text{e$record_not_found. The given key does not locate a record.} \]

Example 2.
By comparison, the following command displays the code number and message associated with the specified error status code name.

\[ \text{display_error e$record_not_found} \]

This display appears.

\[ \text{Code 1112. The given key does not locate a record.} \]

Related Information
See also the description of the \texttt{use_message_file} command. For additional information about status codes, see the OpenVOS Subroutines manuals.
**display_file**

**Purpose**

This command writes a file to the output device or file attached to your `default_output` port, which is usually your terminal, or to the device or file you specify in the command.

**Display Form**

```
--------------------------------- display_file ---------------------------------
file_names:                       yes
-caseless:                        yes
-index:                           no
-match:                           yes
-output_path:                     no
-first_line:                      no
-last_line:                       no
-line_numbers:                    no
-header:                          yes
-raw:                             no
-line_length:                     default
-slave_printer:                   no
-slave_page_length:               60
-min_lines:                       1
-interpret_tabs:                  yes
-file_separator:                  lf
-match_status:                    no
```
Command Line Form

display_file file_names...

- no_caseless
- index index_name
- match character_string
- output_path output_path_name
  - slave_printer
  - first_line first_line_number
  - last_line last_line_number
  - line_numbers
  - no_header
  - line_length
  - raw
  - interpret_tabs start_column, spacing
  - slave_page_length number_of_lines
  - min_lines number_of_lines
  - file_separator
  - match_status

Arguments

- **file_names** Required
  One or more names or star names of files to be displayed. You can specify any type of file (except for transaction files), including IBM® Revisable-Form-Text DCA files and Release 4.2 WordPerfect files.

- **-no_caseless**
  Specifies that when you select -match, the specified match for character_string is case sensitive. If you do not use -no_caseless and use -match, the matching disregards the case of character_string. If you use -no_caseless without also using -match, the command disregards this argument.

- **-index index_name**
  Specifies the name of an index to control the order in which records in the file are displayed. If you use -index, you can give only one file path name, and index_name must be an index to that file.

- **-match character_string**
  Displays only the lines in the files that contain the character string character_string. If you also use -no_caseless, the match is case sensitive. Otherwise, the command disregards the case of the alphabetical characters. By default, the command displays all the lines in the range defined by -first_line and -last_line.

- **-output_path output_path_name**
  Directs the output from the command to the output device or file output_path_name. By default, the command directs the output to your default_output port. You cannot specify both -output_path and -slave_printer in the same command.
-first_line first_line_number
Displays each specified file starting with the line numbered first_line_number. If there are fewer than first_line_number records in the file, the command displays an error informing you of that, and returns you to command level. By default, the display begins at the first line in each file.

-last_line last_line_number
Displays each specified file through the line numbered last_line_number. By default, the display is through the end of the file.

-line_numbers [CYCLE]
Includes line numbers in the display. By default, the command omits line numbers.

-no_header [CYCLE]
Suppresses the display of the names of the specified files. By default, the command displays the file name of each specified file just before its contents.

-raw [CYCLE]
Displays the specified file without interpreting embedded word processor controls, including those from IBM Revisable-Form-Text DCA or Release 4.2 WordPerfect files. Control sequences that do not normally appear on the screen are replaced with the ASCII digits representing the hexadecimal value of the bytes. You cannot specify -raw and -interpret_tabs in the same command.

-line_length
Specifies the number of columns displayed before the line wraps. For terminal devices, the default line length is the width of the output device; for non-terminal output devices (for example, output to a file), the default line length is 132 columns. The default line length for slave printers and for terminals is the same (generally, 80 columns).

-slave_printer [CYCLE]
Directs the output from the command to a printer attached to the terminal. By default, the command directs the output to the default_output port. You cannot specify both -slave_printer and -output_path in the same command.

-slave_page_length number_of_lines
Prints a specified number of lines per page of output, if you also specify -slave_printer. The minimum number of lines you can specify is 10. If you use -slave_printer but omit -slave_page_length, the value of number_of_lines is 60. If you specify -slave_page_length and do not specify -slave_printer, the command disregards this argument.

-min_lines number_of_lines
Specifies the minimum number of lines to be displayed following each line containing the characters specified in the -match argument. If you specify a value of n for number_of_lines, the command displays the line containing the specified string and the n-1 subsequent lines, if any. By default, the value of number_of_lines is 1.

-interpret_tabs start_column, spacing
Interprets occurrences of the ASCII tab character. You must give the column number start_column of the first tab stop and the number spacing of positions between tab
display_file

stops. A comma must separate the two numbers. You cannot specify both -interpret_tabs and -raw in the same command.

- **-file_separator**
  Controls the appearance of the output when the -output_path argument is used. When using the -output_path argument, you can separate the output from multiple files by specifying the ff (form-feed) value. By default, the -file_separator argument uses the lf (line-feed) value.

- **-match_status**
  If you specify this argument with the -match argument, the command sets the (command_status) command function to one of the following:
  
  - e$no_match (7857) if the command finds no matches
  - 0 if the command finds a match

  This argument has no effect unless you also specify the -match argument.

**Explanation**

The display_file command writes the file file_names to the file or device attached to your default_output port or to the file or device specified in an -output_path argument. Normally, the port is attached to your terminal, so the contents of the files are displayed on your terminal screen.

The display_file command displays any type of file, except for pipes, message queues, and server queues. If you specify only one file and it is a pipe, message queue, or server queue, the command returns an error. If you specify more than one file, either explicitly or via a star name, the command does not display any file that is a pipe, message queue, or server queue, and the command status is set to zero. If you specify -match in this case, files that cannot be displayed are treated as if they contain no matches.

The display_file command displays stream files that have been opened using region locking (this includes files that are open and in use by POSIX programs), IBM Revisable-Form-Text DCA file, and Release 4.2 WordPerfect files. To display a WordPerfect file, the file name must be in the form filename.wpf. Only those files whose names have the suffix .wpf are processed as WordPerfect documents.

The -match argument allows you to display only the lines containing the string character_string. If character_string contains spaces, you must enclose the string in apostrophes. This argument is convenient for displaying only the portions of a file that contain a particular string and for identifying all files that contain the string.

With the -output_path argument, you can direct the output to a file or device different from the current attachment of your default_output port. If output_path_name is an existing file, the display_file command truncates the file before writing to it. If -output_path is not an existing file, the command creates a sequential file with that name and writes the output to it. If you specify this argument and specify more than one file, the command appends the output from each file to the output from the preceding files.
The -file_separator lf argument writes a blank line after each file if the display command produces any output while processing a file. For example, if any characters in a line matched the -match character_string argument, then a blank line is written after all of the matching lines have been displayed. If multiple files are specified and the ff character is used in the -file_separator argument, then a form-feed character is written to separate the output.

The display_file command opens an input file for dirty input if the input file is locked using record locking, or if it is exclusively locked by another process and you do not specify the -index argument. In these situations, the command displays the following messages:

For exclusive locking:

```
   display_file:  File is in use; using dirty input mode.
   Some data may not be visible yet. object_name
```

For record locking:

```
   display_file:  File is in use; using dirty input mode.
   Some records may not be visible yet. object_name
```

If a file is already locked using region locking, the command attempts to open the file only if you did not specify the -index argument.

The -slave_printer argument is supported for both TeleVideo-style (V101, V102, V103) and ANSI-style (VT100, VT220, VT320, CIT482, V105, V109) terminals. In addition, OpenVOS supports this argument on any terminal whose terminal-type definition (.ttt) file defines the sequences aux-printer-on and aux-printer-off. The following standard ANSI ttt files define the aux-printer-x output sequences: cit482, vt100, vt220, vt320, v105, v105_ansi, v105_epc, and v109.

Access Requirements
You need read access to a file to display it.

Examples

Example 1.
The following command displays the path name and contents of the IBM Revisable-Form-Text DCA file this_week in the current directory.

```
   display_file this_week
```

Example 2.
To display the lines containing the character string city-code in all OpenVOS COBOL source modules in your current directory, use this command.

```
   display_file *.cobol -match city-code -line_numbers
```

This command also displays the path names of the specified files and the line numbers of the lines that contain the string city-code.
Related Information

See also the description of the `display` command.
**display_file_status**

**Purpose**
This command displays information about a set of files you specify.

**Display Form**

```
----------------------------- display_file_status -----------------------------
file_names: [mask]
-count_keys: no
-index_name:
```

**Command Line Form**

```
display_file_status file_names...
```

**Arguments**

- **file_names**  
  The path name of one or more names or star names of files. The command displays information about each file with a matching name.

- **-count_keys**  
  Counts the keys for an index and displays the number of keys counted.

- **-index_name**  
  Specifies the index for which information should be returned.

**Explanation**

The `display_file_status` command displays some or all of the following, depending primarily on the type of file:

- the full path name
- the file organization (sequential, relative or stream)
- the date and time the file was last used
- the date and time the file was last modified
- the date and time the file was last saved by a `save` command with `-backup` specified
- the date and time the file was created
- the expiration date of the file
- whether the file is a transaction file
- whether the file is set for implicit locking
- whether region locks are enforced for the file
- whether the file is a pipe file
display_file_status

- whether a safety switch is set for the file
- the extent type for the file and for each index associated with the file
- the size of extents
- the open options
- the number of disk blocks to be allocated to the file when more disk space is required (note that the value default is displayed for all files for which allocation size has not been explicitly set to a value other than 0)
- the record size
- the next byte to be written
- the largest record number of the records written
- the number of disk blocks used to store the file
- the number of indexes to the file
- the number of disk blocks to be added each time the file needs more space
- your access to the file
- the author (last user to modify)
- whether the file has RAM usage
- the default character set
- the shift mode
- the maximum queue depth for server queues and one-way server queues
- the names of the indexes to the file
- information about each index, including type, collation, key components, and open options
- the maximum key length, in bytes, that is allowed in the file
- the size of the file:
  - for non-transaction-protected fixed or relative files, the number of records in the file
  - for other file types, the number of bytes in the file.

If you do not specify either the -count_keys or the -index_names argument, the default behavior of the command is to report on all indexes in the file and not count keys. Otherwise, the command behaves as follows:

- If you specify -count_keys but do not specify -index_name, display_file_status reports on all indexes in the file and counts keys for each index.

- If you specify -index_name but do not specify -count_keys, display_file_status reports on only the specified index but does not count keys.

- If you specify both arguments, display_file_status reports on only the specified index and counts keys for the index.

In addition, if you specify -count_keys, the display_file_status command’s output for each index includes a field that reports the number of keys. The output follows.

number of keys: number_of_keys

The number_of_keys value represents the number of keys in the index.
If you specify `-count_keys` for an index residing on a remote module that does not support the counting of keys, the output is as follows:

```
number of keys: not supported on remote module
```

**Note:** The `display_file_status` command shows index names in order of the index address inside the file. This order may change if the file is specified as the subject of the `copy_file`, `move_file`, `restore_object`, or `save_object` command.

For more information about the open options, see the description of the `display_open_options` command.

**Access Requirements**

To obtain information about a file, you need execute, read, or write access to the file, or status or modify access to the directory that contains the file.

**Examples**

**Example 1.**
The command `display_file_status this_week` displays the following information about the specified indexed file.

```
name: %s1#d02>Sales>east>Jones>reports>this_week
file organization: fixed file
last used at:  94-09-06 10:18:26 edt
last modified at:  94-09-06 10:15:55 edt
last saved at:  never
time created:  94-09-06 10:15:55 edt
expiration date:  95-01-01 00:00:00 edt
transaction file:  no
implicit locking:  no
pipe file:  no
safety switch:  no
audit:  no
extent size:  1
record size:  126
last record:  0
blocks used:  0
num indexes:  1
allocation size:  default
mode:  w
author:  Jones.East
tag type:  0
tag version:  0
```

(Continued on next page)
(Continued)

<table>
<thead>
<tr>
<th>index name:</th>
<th>prospects</th>
</tr>
</thead>
<tbody>
<tr>
<td>key components:</td>
<td>1, 6</td>
</tr>
<tr>
<td>type:</td>
<td>embedded_key</td>
</tr>
<tr>
<td>collation:</td>
<td>ascii</td>
</tr>
<tr>
<td>data type:</td>
<td>nonvarying string</td>
</tr>
<tr>
<td>ascending:</td>
<td>yes</td>
</tr>
<tr>
<td>duplicates:</td>
<td>yes</td>
</tr>
<tr>
<td>null keys:</td>
<td>no</td>
</tr>
<tr>
<td>extent index:</td>
<td>no</td>
</tr>
<tr>
<td>automatic update:</td>
<td>yes</td>
</tr>
<tr>
<td>extent_size:</td>
<td>1</td>
</tr>
<tr>
<td>blocks:</td>
<td>0</td>
</tr>
</tbody>
</table>

**Example 2.**

The command `display_file_status >Sales>Jones>sq1` displays the following information about the specified link to a one-way server queue:

<table>
<thead>
<tr>
<th>name:</th>
<th>%s1#d02&gt;Sales&gt;Jones&gt;sql</th>
</tr>
</thead>
<tbody>
<tr>
<td>is a link to:</td>
<td>%s1#d02&gt;Sales&gt;Jones&gt;quotas&gt;sql</td>
</tr>
<tr>
<td>file organization:</td>
<td>one way server queue</td>
</tr>
<tr>
<td>last used at:</td>
<td>never</td>
</tr>
<tr>
<td>last modified at:</td>
<td>94-03-31 16:22:19 EST</td>
</tr>
<tr>
<td>last saved at:</td>
<td>never</td>
</tr>
<tr>
<td>time created:</td>
<td>94-03-31 16:22:19 EST</td>
</tr>
<tr>
<td>safety switch:</td>
<td>no</td>
</tr>
<tr>
<td>audit:</td>
<td>no</td>
</tr>
<tr>
<td>extent size:</td>
<td>1</td>
</tr>
<tr>
<td>blocks used:</td>
<td>0</td>
</tr>
<tr>
<td>allocation size:</td>
<td>default</td>
</tr>
<tr>
<td>mode:</td>
<td>w</td>
</tr>
<tr>
<td>author:</td>
<td>Jones.east</td>
</tr>
<tr>
<td>max_queue_depth:</td>
<td>256</td>
</tr>
<tr>
<td>tag type:</td>
<td>0</td>
</tr>
<tr>
<td>tag version:</td>
<td>0</td>
</tr>
</tbody>
</table>
Example 3.
The command `display_file_status journal` displays the following information about the specified file:

```
name:                       %s1#d02>Sales>east>journal
file organization:          sequential file
last used at:               94-05-11 19:02:48 EDT
last modified at:           94-05-09 09:22:47 EDT
last saved at:              94-05-09 19:11:45 EDT
time created:               94-05-09 09:22:14 EDT
transaction file:           no
implicit locking:           no
pipe file:                  no
safety_switch:              no
audit:                      no
extent_size:                no
next byte:                  92622
num indexes:                0
allocation size:            default
mode:                       r
author:                     Clark.Sales
character set:              latin_1
shift mode:                 single
tag type:                   0
tag version:                0
```

Example 4.
The command `display_file_status file1` displays the following information about the specified DAE file:

```
name: %s1#d02>Sales>east>Jones>file1
file organization:         relative file
last used at:              03-09-11 08:24:43 edt
last modified at:          03-09-11 08:24:43 edt
last saved at:             never
time created:              03-09-11 08:22:37 edt
transaction file:          no
implicit locking:          no
pipe file:                 no
safety switch:             no
audit:                     no
dynamic extents:           yes
extent size:               8
record size:               12
last record:               1
blocks used:               8
num indexes:               0
allocation size:           default
```

(Continued on next page)
Example 5.
In the following example of a simple file with nonextent indexes, the file is a zero-length file.

- **name:** %s1#d02>Sales>Jones>test_file1
- **file organization:** sequential file
- **last used at:** 00-05-03 16:23:42 EDT
- **last modified at:** 00-05-03 16:22:46 EDT
- **last saved at:** never
- **time created:** 00-05-03 16:22:46 EDT
- **transaction file:** no
- **implicit locking:** no
- **pipe file:** no
- **safety switch:** no
- **audit:** no
- **dynamic extents:** no
- **extent size:** 1
- **next byte:** 0
- **blocks used:** 0
- **num indexes:** 2
- **allocation size:** default

mode: w
author: Jones.Sales
tag type: 0
tag version: 0

**Index details:**
- **index name:** test_index_11
- **key components:** 1,2 10,15
- **type:** embedded_key
- **collation:** ascii
- **data type:** nonvarying string
- **ascending:** yes
- **duplicates:** yes
- **null keys:** no
- **extent index:** no
- **dynamic extents:** no
- **automatic update:** yes
- **extent size:** 1
- **blocks:** 0

(Continued on next page)
(Continued)

index name: test_index_12
key components: 4,10 30,5
type: embedded_key
collation: ascii
data type: nonvarying string
ascending: yes
duplicates: yes
null keys: no
extent index: no
dynamic extents: no
automatic update: yes
extent_size: 1
blocks: 0

Example 6.
The following example shows an SAE file with SAE indexes.

name: %s1#d02>Sales>Jones>test_file2
file organization: sequential file
last used at: 99-10-29 16:23:42 EDT
last modified at: 99-10-29 16:22:46 EDT
last saved at: never
time created: 99-10-29 16:22:46 EDT
transaction file: no
implicit locking: no
pipe file: no
safety switch: no
audit: no
dynamic extents: no
extent size: 8
initial allocated size: 32
next byte: 0
blocks used: 32
num indexes: 1
allocation size: default
mode: w
author: Jones.Sales
tag type: 0
tag version: 0

(Continued on next page)
Example 7.
In the following example of a DAE file with DAE indexes, the file was created with the `-num_records` argument set to 3.

<table>
<thead>
<tr>
<th>name:</th>
<th>%s1#d02&gt;Sales&gt;Jones&gt;test_file3</th>
</tr>
</thead>
<tbody>
<tr>
<td>file organization:</td>
<td>sequential file</td>
</tr>
<tr>
<td>last used at:</td>
<td>00-06-03 16:23:42 EDT</td>
</tr>
<tr>
<td>last modified at:</td>
<td>00-06-03 16:22:46 EDT</td>
</tr>
<tr>
<td>last saved at:</td>
<td>never</td>
</tr>
<tr>
<td>time created:</td>
<td>00-06-03 16:22:46 EDT</td>
</tr>
<tr>
<td>transaction file:</td>
<td>no</td>
</tr>
<tr>
<td>implicit locking:</td>
<td>no</td>
</tr>
<tr>
<td>pipe file:</td>
<td>no</td>
</tr>
<tr>
<td>safety switch:</td>
<td>no</td>
</tr>
<tr>
<td>audit:</td>
<td>no</td>
</tr>
<tr>
<td>dynamic extents:</td>
<td>yes</td>
</tr>
<tr>
<td>extent size:</td>
<td>64</td>
</tr>
<tr>
<td>next byte:</td>
<td>0</td>
</tr>
<tr>
<td>blocks used:</td>
<td>64</td>
</tr>
<tr>
<td>num indexes:</td>
<td>2</td>
</tr>
<tr>
<td>allocation size:</td>
<td>default</td>
</tr>
<tr>
<td>mode:</td>
<td>w</td>
</tr>
<tr>
<td>author:</td>
<td>Jones.Sales</td>
</tr>
<tr>
<td>tag type:</td>
<td>0</td>
</tr>
<tr>
<td>tag version:</td>
<td>0</td>
</tr>
</tbody>
</table>

(Continued on next page)
Example 8.
The following example illustrates how an extended sequential file is represented in the display_file_status output.

name: %s1#d02>Sales>Jones>test_file4
file organization: sequential file [4]
next byte: 1024

In this example, the [4] following the file organization indicates the number of record offset units (in this case, four). This bracketed value appears in the output only for extended sequential files.
Example 9.
The following example shows output for a file with non-default open options for itself and its index.

name: %s1#d02>system>error_codes.text  
is a link to: %s1#d02>r15.2_system>error_codes.text  
file organization: sequential file

  .
  .
  .
tag type: 0
  tag version: 0
  record count: 0
  data byte count: 0
  index name: number
  type: separate_key
  collation: numeric
  data type: varying string
  ascending: yes
  duplicates: no
  null keys: no
  extent index: no
  dynamic extents: no
  extent_size: 1
  open options: full preread_extents
  blocks: 18
Example 10.

The following example shows output for a file with RAM usage.

```
name: %s1#d02>Sales>east>journal2
file organization: sequential file
last used at: never
last modified at: 10-05-20 09:07:24 edt
last saved at: never
time created: 10-05-20 09:07:24 edt
transaction file: no
implicit locking: no
pipe file: no
safety switch: no
audit: no
dynamic extents: no
extent size: 1
next byte: 0
blocks used: 0
num indexes: 0
allocation size: default
mode: w
author: Clark.Sales
tag type: 0
tag version: 0
usage: ram
record count: 0
data byte count: 0
```

Example 11.

The following example shows output for a 64-bit stream file that contains two non-zero bytes, one at offset 0 and another at offset 34,000,000,000. Such a file may be sparse, depending on how it was built, and if so, it occupies fewer disk blocks than an identical normal stream file.

```
name: %s1#d02>Sales>Jones>filex
file organization: stream file (64-bit/rstr)
last used at: 13-02-15 15:50:50 est
.
.
.
dynamic extents: yes
extent size: 16
next byte: 34000000001
blocks used: 34
sparse: yes
mode: w
```

The designation (64-bit/rstr) means that filex is restricted (in this case, due to both size and sparseness). If the file were not sparsely allocated, it would occupy 8,300,781 disk blocks; instead, it occupies only the minimum required: enough blocks for the 16-block extent holding the first and last non-zero bytes and the supporting file map blocks.
Example 12.
The following example shows output for a flex file.

```
display_file_status bigflex
name: %s1#d02>Sales>Jones>bigflex
file organization: stream file (64-bit/rstr)
last used at: 13-09-02 16:51:16 edt
last modified at: 13-09-02 16:44:40 edt
last saved at: never
time created: 13-09-02 16:44:39 edt
transaction file: no
implicit locking: no
region locks reqd: no
pipe file: no
safety switch: no
audit: no
dynamic extents: yes
extent size: flex
owner access: none
next byte: 540142534656
blocks used: 616
num indexes: 0
allocation size: 1
sparse: yes
mode: w
author: Jones.Sales
tag type: 0
tag version: 0
```

In this example, the file represents the largest flex file supported (just under 512 GB in size). This file is both greater than 2 GB and sparse, and it is therefore restricted. Flex files are not necessarily restricted, but they are always 64-bit stream files.

If you issue `display_file_status` on a module that does not support flex files, the `extent size` field is -1. If you issue `display_file_status` on a module that does not support 64-bit stream files, the `file organization` field is `stream file`.

Related Information
See the description of the `create_index` command for further information on indexes. See also the command descriptions of `compare_files`, `copy_file`, `create_file`, `delete_file`, `dump_file`, `locate_files`, `move_file`, `set_file_allocation`, `set_pipe_file`, `set_safety_switch`, and `truncate_file`. 
**display_line**

**Purpose**
This command displays the literal text of the argument that follows the command name on the command line.

**Display Form**

```
------------------------ display_line ------------------------
text: [ ]
```

**Command Line Form**

```
display_line [text]
```

**Arguments**

- **text**
  The text to be displayed. The text can contain words, abbreviations to be replaced, quoted strings, and command functions to be evaluated.

**Explanation**

The `display_line` command replaces abbreviations and evaluates command functions in the text before displaying it. This command allows you to view the results of replacing abbreviations and evaluating command functions. You can also use it in command macros to display messages.

**Examples**

**Example 1.**
Suppose that this is the home directory of the current user.

```
%S1#d02>Sales>Smith
```

Suppose also that the following directive appears in the user’s abbreviations file.

```
all mr by make_reports
```

Now consider this command.

```
display_line (home_dir)>mr
```
The command evaluates the command function \(\text{home\_dir}\), expands the abbreviation \(mr\), and generates this display.

\[
\text{%s1#d02>Sales>Smith>make\_report}
\]

**Example 2.**
The following command evaluates the command function \(\text{time}\), then displays the current time followed by the specified date string.

\[
\text{display\_line (time) Tuesday, January 22, 1990}
\]
**display_notices**

**Purpose**
This command displays system-wide notices.

**Display Form**
```
---------------------------------- display_notices ----------------------------------
No arguments required. Press ENTER to continue. 
```

**Command Line Form**
display_notices

**Explanation**
The `display_notices` command displays, in particular, all files named 
(master_disk)>system>notices*.

**Examples**
The following example illustrates the output of the `display_notices` command.

```plaintext
Welcome to %s1#m2

90-04-22  Congratulations to Bill Smith on a major sale in the western region.

90-04-22  NOTICE: The fiscal year ends June 30. Please plan accordingly.
```
display_object_module_info

Purpose
This command returns selected information from an object module or modules.

Display Form

```
-------------------------- display_object_module_info --------------------------
  object_module_name:     
  -header:       yes
  -source_files:  no
  -internal_entries:  no
  -external_entries:  no
  -output_path:
```

Command Line Form

display_object_module_info object_module_name
  [-no_header]
  [-source_files]
  [-internal_entries]
  [-external_entries]
  [-output_path path_name]

Arguments

- **object_module_name**  
  The names or star names of the object modules about which information is to be displayed.

- **-no_header**  
  Supresses header information. By default, the command displays information about how the object module was compiled.

- **-source_files**  
  Displays source file information. By default, the command does not display information about the source files that were used in the compilation of the object module.

- **-internal_entries**  
  Displays information on internal entries. By default, the command does not display information about the internal entries that occurred in the source module that was compiled to create the object module.
- **-external_entries**
  Displays information on external entries. By default, the command does not display information about the external entries declared in the source module that was compiled to create the object module.

- **-output_path path_name**
  Directs the output of the command to a file or device. By default, the command directs the output to the user’s terminal.

**Explanation**

The `display_object_module_info` command returns information from an object module file. The optional arguments determine the information that the command extracts.

Unless you select `-no_header`, the command extracts information about how the object module was compiled. Compilation information includes the following:

- **source module path**
  The path name of the compiled source module.

- **compiler**
  The name of the compiler used to compile the object module. The possible values for compiler are `c`, `cc`, `cobol`, `edit_form`, `fortran`, `pascal`, or `pl1`.

- **os_version**
  The version of the operating system that was running on the processing module when the object module was compiled.

- **user name**
  The user name of the user who compiled the object module.

- **date_time compiled**
  The date and time when the object module was compiled. For example: `99-08-03 19:57:42 EDT`.

- **system name**
  The name of the system on which the object module was compiled.

- **data object file**
  Specifies whether the object module is a data object or a compiled object module.

- **processor**
  The type of CPU on which the source module was compiled. The possible value is `pentium4`. 
options
The compiler options that were selected for the compilation and that are included in the object module. Possible values are as follows:

check
The source module was compiled with check (array subscript reference checking) specified.

optimize
The source module was compiled with optimize specified.

optimization_level
The optimization level that was applied to the compilation of this source module.

table
The source module was compiled with table (full symbol table) specified.

production_table
The source module was compiled with production_table specified.

profile
The source module was compiled with profile (statement execution count) specified.

cpu_profile
The source module was compiled with cpu_profile (statement execution count and CPU time) specified.

system_programming
The source module was compiled with system_programming (sub-level-1 member checking and implicit data-type-conversion checking) specified.

fortran66
The source module was compiled with fortran66 (execute each do-loop at least once) specified.

fixedoverflow
The source module was compiled with fixedoverflow (integer and fixed-point-decimal overflow checking) specified.
full_list

The source module was compiled with full_list (compilation listing with assembly language listing) specified.

xref

The source module was compiled with xref (cross-reference compilation listing) specified.

posix

The source module uses POSIX features.

longmap/check

The source module was compiled with longmap_check specified.

longmap

The source module was compiled with longmap specified.

shortmap/check

The source module was compiled with shortmap_check specified.

minimal_stack_frames

If present, the generated code may not store a back pointer to the previous stack frame. If absent, the generated code always stores a back pointer to the previous stack frame.

untyped_storage_sharing

The source module was compiled with untyped_storage_sharing specified.

typed_storage_sharing

The source module was compiled with no_untyped_storage_sharing specified.

store_args

The source module was compiled with store_args specified.

extended_names_disable

The source module was compiled with extended_names_disable specified.

extended_names_enable

The source module was compiled with extended_names_enable specified.
abi_bitfields

The source module was compiled with the cc command with -compatible_bitfields not specified.

If you select the -source_files argument, the display_object_module_info command extracts source file information from the object module. This includes the path name, date and time created, and date and time modified for the main source module and for all of the include files used in the compilation.

If you select the -internal_entries argument, the command extracts internal entries information from the object module. Internal entries information includes the entry name, the number of arguments the entry takes, and whether the entry is a subroutine or a function.

If you select the -external_entries argument, the command extracts external entries information from the object module. External entries information includes the entry name, the number of arguments the entry takes, and whether the entry is a subroutine or a function.

If you omit all of the arguments, the command displays the path name of the object module.

Access Requirements

You need read access to the object modules about which you want information.

Examples

An example of the display_object_module_info command follows.

```
display_object_module_info testb.obj -source_files
```

```
%sl#d03>Sales>Smith>work>testb.obj
Compilation Information

source module path: %sl#d03>Sales>Smith>work>testb.pl1
compiler:          pl1
os_version:        Release 17.1.0
user name:         Smith.Sales
date_time compiled: 10-05-20 09:58:13 edt
system name:       %s1
data object file:   no
processor:         pentium4
options:
xref
system_programming
profile
table
optimize
optimization_level 1
check
longmap
```

(Continued on next page)
(Continued)

minimal_stack_frames
typed_storage_sharing
store_args

Source Files

<table>
<thead>
<tr>
<th>Source module name</th>
<th>Date-Time Created</th>
<th>Date-Time Modified</th>
</tr>
</thead>
<tbody>
<tr>
<td>$s1#d03&gt;Sales&gt;Smith&gt;work&gt;testb.pl1</td>
<td>10-02-08 11:19:37</td>
<td>10-02-08 12:01:33</td>
</tr>
</tbody>
</table>

Include file name

<table>
<thead>
<tr>
<th>Date-Time Created</th>
<th>Date-Time Modified</th>
</tr>
</thead>
<tbody>
<tr>
<td>10-02-08 11:19:37</td>
<td>10-02-08 12:01:33</td>
</tr>
</tbody>
</table>

No include files.
display_open_options

Purpose

This command displays the open options for one or more files or indexes.

Display Form

```
file_names: ...
-index: ...
-active:  no
-active_only:  no
-brief:  no
```

Command Line Form

```
display_open_options file_names ...
    [-index  index_name]
    [-active]
    [-active_only]
    [-brief]
```

Arguments

- **file_names**

  Required

  One or more names or star names of files for which open options are to be displayed.

- **-index**  \texttt{index_name}

  Specifies the name of one or more indexes for \texttt{file_names} for which open options are to be displayed. The value of \texttt{index_name} is a string. By default, there is no value for \texttt{index_name}. You can specify an asterisk (*) to display all indexes of the file, including system-defined indexes (for example, \_deleted_record_index).

- **-active**

  Displays the open options that are associated with a file’s current activation as well as for the file itself. These open options may or may not be identical. For example, activated file open options reflect the specific default open options when these are inherited from directories (that is, a file’s cache-mode options might be \texttt{default}, whereas the activated file’s cache-mode options may be \texttt{transient}, if that is what the directory default was at the time of activation). Also, a file’s open options may have changed since the file was first activated, and the activated file’s open options may have been explicitly changed from what they inherited from the permanent file attributes, either by use of the \texttt{set_open_options -active} command or by calls to either the \texttt{s$control} or \texttt{s$set_open_options} subroutine for open files.
The activated file’s open options are shown on the line following the file’s open options.

By default (the value no), the command does not display the open options associated with a file’s current activation.

- **-active_only**
  Displays the open options for only those files or indexes that are currently active. By default (the value no), the command displays the open options for all specified files and indexes.

- **-brief**
  Displays the open options for only those files or indexes that do not have default open options. By default (the value no), the command displays the open options for all specified path names.

**Explanation**

This command displays the open options for all files or indexes that match the given star names.

If you set the -brief argument to no (the default), the command displays each of the path names that the command examines only if the open options are not default values. The open options are displayed as canonical strings.

If the command cannot process one of the files, it displays an error message and continues on to the next specified file.

This command uses the following syntax to display open options for files and indexes:

```
display_open_options cache_mode [preread_mode n]
```

An explanation of the syntax follows.

- **cache_mode** is one of the following values:
  - * indicates that the cache mode is in default mode.
  - n indicates that the cache mode is normal.
  - M indicates that the cache mode is in memory-resident mode.
  - T indicates that the cache mode is in transient mode.

- **preread_mode**, which the command displays only if the values are non-default values, is one of the following values:
  - r indicates that the file or index has normal preread extents.
  - N indicates that the file or index has never preread extents.
  - S indicates that the file has preread extents only with sequential access.
  - F indicates that the file or index has full preread extents.

- **n** is the read-ahead value, which is 0 through 127. The command displays the read-ahead value only if it is a non-default value.
display_open_options

For more information about the open options, see the manual OpenVOS System Administration: Administering and Customizing a System (R281). See also the description of the $get_open_options subroutine in the OpenVOS Subroutines manuals.

Examples

The following example illustrates how to use the display_open_options command to display open options for a file.

```
m100: display_open_options charbit
n-F2              %s#m100>disk>charbit
ready 11:03:24
```

In the preceding output, n-F2 indicates the following about the file charbit:

- n indicates that the cache mode is in normal mode.
- F indicates that the file or index has preread full extents.
- 2 indicates that the read-ahead value is 2.

Related Information

See also the descriptions of the display_default_open_options, set_default_open_options, and set_open_options commands.
display_print_defaults

Purpose

This command displays the default values for a specified print queue or queues.

Display Form

--- display_print_defaults ---
-queue: Standard
-module: 
-long: no
-all: no

Command Line Form

display_print_defaults [-queue queue_name]

[ -module module_name]
[-long]
[-all]

Arguments

▶ -queue queue_name
   Specifies the name of a print queue. The command displays the default page lengths and line lengths of queue_name. By default, the command uses the default print queue, either on the module specified in -module or on the current module.

▶ -module module_name
   Specifies the module containing the specified queue. By default, the command uses your current module.

▶ -long
   Displays detailed information about a particular queue. If you omit -queue and specify -long, the command displays extensive information about the standard queue.

▶ -all
   Displays the default values for all print queues. By default, the command displays the default values for the queue specified in -queue.
**Explanation**

The `display_print_defaults` command displays the default values for a given print queue or all print queues. The operating system places print requests in queues that control the spooling and printing order of files.

You can override the default page size and line length values of a print queue by selecting arguments in the `print` command.

**Examples**

**Example 1.**
The following example illustrates the screen output for the `display_print_defaults` command with the `-all` argument.

<table>
<thead>
<tr>
<th>Queue name</th>
<th>Page Size</th>
<th>Line length</th>
</tr>
</thead>
<tbody>
<tr>
<td>single_sheet</td>
<td>66</td>
<td>84</td>
</tr>
<tr>
<td>lqp</td>
<td>66</td>
<td>84</td>
</tr>
<tr>
<td>spinwriter</td>
<td>66</td>
<td>84</td>
</tr>
</tbody>
</table>

**Example 2.**
The following example illustrates the screen output for the `display_print_defaults` command issued with the `-long` argument. Use of the `-queue` argument with `-long` argument supplies more information about any queue.

```plaintext
queue: standard
device_type: line_printer
form_type: standard
page_density: 6_lines_per_inch
line_density: 10_chars_per_inch
channels: 1,1
page_size: 66
line_length: 85
separators: yes
```

**Related Information**

See the descriptions of the `print` command for more information about printers, spoolers, and print queues.
**display_print_status**

**Purpose**
This command displays the status of the printer or printers connected to a module or modules.

**Display Form**
```
---------------------------------- display_print_status ----------------------------------
-module: current_module
-long: yes
```

**Command Line Form**
```
display_print_status [-module module_name]

[-no_long]
```

**Arguments**
- `-module module_name`
  Specifies a module name or star name. By default, the command uses the current module. If there are no devices connected to the current module, you must specify the module or modules to which devices are attached.

- `-no_long`
  Displays detailed information about a printer or printers connected to a module or modules. It gives information about printer names, type, status, and printer process names. It also lists hoppers and queues, indicating which are current, and whether they are mounted. If you omit this argument, the command displays the device name, type, state, and names the queues for all printers connected to the specified module or modules.

**Explanation**
The `display_print_status` command displays the status of the printer or printers connected to a module or modules, including information about the print queues and spoolers.

The command displays the names of any devices currently being spooled on a particular module, the state of the spoolers, and the queues that manage the device or devices. If you give the `-long` argument, it also displays the names of printers as well as their type and status. In addition, it displays printer process names, lists hoppers and queues, and indicates which are current, and which are mounted.
The primary use of the command is to select an appropriate queue for your print request. The command tells you which spoolers are running or stopped and why, and which queues are currently being processed. Use this command to help choose a module to specify when invoking the print command. Use the display_print_status command with the list_print_requests command, to tell how many jobs are waiting on a queue and the size of each job.

The module_name argument can be a module star name. If you issue the -module argument with a module star name, the command displays status information separately for each module. In this way you can determine the module associated with a particular device as well as which queues receive print requests.

Examples

Example 1.
The following example illustrates the output of the display_print_status command for module %s1#m2.

display_print_status -module %s1#m2

(Module %s1#m2)----------------------------------------------------

<table>
<thead>
<tr>
<th>DEVICE</th>
<th>TYPE</th>
<th>STATE</th>
<th>QUEUES</th>
</tr>
</thead>
<tbody>
<tr>
<td>#p.1.0</td>
<td>4590</td>
<td>run</td>
<td>* spinwriter</td>
</tr>
<tr>
<td>#p.1.1</td>
<td>4975</td>
<td>run</td>
<td>* lqp</td>
</tr>
<tr>
<td>#p.1.2</td>
<td>4590</td>
<td>run</td>
<td>* standard on module %s1#m2</td>
</tr>
<tr>
<td>#p.1.3</td>
<td>4590</td>
<td>run</td>
<td>* standard on module %s1#m3</td>
</tr>
</tbody>
</table>

The device is the name of the device as it is known to the system, and the type is the defined device type. The state in this example is run for all devices, but could be any of the states listed below. The queues are those defined for the current module, and the asterisk indicates that the processor is acting on the queue. It is possible to have no active queue for a printer.

Possible states of the spooler are listed below.

- run
- stopped
- offline
- paper feed
- paper jam
- paper out
- process fault
- band fault
- carriage fault
- printer not operational
- stand by
- susp
Most states reflect the status of a particular printer or job; the `susp` status indicates that the activity of an entire queue has been suspended. This can happen when the module of the queue is offline and you are checking the status of that queue from another module.

**Example 2.**
The following example illustrates the output of the `display_print_status` command invoked with the `-long` argument on module `%sales#m27`.

```
display_print_status -long
(Module %sales#m27)--------------------------------------------------
Printer: #p27_b               Type: l306           Status: run
  Process name: spooler
    Hopper 1: (current)      Queue:  wide27 (mounted, current)
    Hopper 2:                Queue:  narrow27 (mounted)

Printer: #spin27              Type: l151           Status: run
  Process name: spooler
    Hopper 1: (current)      Queues: spinwriter (mounted, current)
                               spin27
```

**Related Information**
See also the command descriptions of `display_print_defaults`, `list_print_requests`, and `print`. For more information on the print spooler, see the description of the `spool_admin` command in *VOS System Administration: Administering the Spooler Facility* (R286).
display_program_module

Purpose
This command displays information about the authorship, history, size, and structure of a program module.

Display Form

```
--------------------------- display_program_module --------------------------
program_module_name: no
-dates: no
-disassemble: no
-dump: no
-dynamic_table: no
-dynref_map: no
-dynsym_map: no
-entry_map: no
-external_vars_map: no
-full: no
-hash_map: no
-header: no
-line_numbers: no
-link_map: no
-link_names_map: no
-module_map: no
-object_dirs_map: no
-release: no
-relocation_map: no
-string_pool: no
-page: no
```
Command Line Form

```
display_program_module program_module_name
   [-dates]
   [-disassemble]
   [-dump]
   [-dynamic_table]
   [-dynref_map]
   [-dynsym_map]
   [-entry_map]
   [-external_vars_map]
   [-full]
   [-hash_map]
   [-header]
   [-line_numbers]
   [-link_map]
   [-link_names_map]
   [-module_map]
   [-object_dirs_map]
   [-release]
   [-relocation_map]
   [-string_pool]
   [-page number]
```

Arguments

- `program_module_name` Required
  The path name of the program module file. The program module can have either a fixed or stream file organization. You cannot specify a star name. If you specify only this argument, the command displays only the program module header. For a description of the program module header, see the Sample Output section.

- `[-dates]` Cycle
  Specifies that the command display the file modification date and time and compilation date and time for every object module in the object module map. By default, the command does not display the program module modification and compilation dates.

- `[-disassemble]` Cycle
  Specifies that the command display the program module’s machine code as pseudo-assembly code. If you specify this argument and the `-page` argument, the command displays pseudo-assembly code for the specified page. If you specify this argument and the `-line_numbers` argument, the command ignores the `-disassemble` argument. By default, the command does not display the program module’s machine code as pseudo-assembly code.

- `[-dump]` Cycle
  Specifies that the command dump the program module’s virtual pages. If you specify this argument and the `-page` argument, the command dumps the specified page. By default, the command does not dump the program module’s virtual pages.

- `[-dynamic_table]` Cycle
  Specifies that the command display the contents of the dynamic table. By default, the command does not display the contents of the dynamic table.
**display_program_module**

- **-dynref_map**
  Specifies that the command display the contents of the dynref map. The *dynref map* replaces the link map in shared libraries and dynamically-linked main program modules. Each entry in the map corresponds to an external symbol reference in the executable. By default, the command does not display the contents of the dynref map.

- **-dynsym_map**
  Specifies that the command display the contents of the dynsym map. The *dynsym map* contains information about symbols defined in this program module; it contains only global symbols and is used for dynamic linking. By default, the command does not display the contents of the dynsym map.

- **-entry_map**
  Specifies that the command display the program module’s entry map, which is used by the `s$monitor` and `s$monitor_full` subroutines, and the `analyze_pc_samples` and `harvest_pc_samples` commands. The entry map is created when you bind object modules and specify the `-retain_all` argument. By default, the command does not display the program module’s entry map.

- **-external_vars_map**
  Specifies that the command display the program module’s external and static variables map. **External variables** are variables that can be shared by more than one program. **Static variables** are variables that are used during the execution of a single program. By default, the command does not display the program module’s external and static variables.

- **-full**
  Specifies that the command display the program module header, the dynamic table, and maps, including the entry, dynref, dynsym, external variable, link names, object module, and object directory maps. (This argument does not display the hash map.) For a description of the program module header and maps, see the Sample Output section. By default, the command does not display the program module header, the dynamic table, or maps.

- **-hash_map**
  Specifies that the command display the contents of the hash map. The *hash map* is a data structure that uses a hash function to efficiently map certain identifiers to associated values in the dynsym map. By default, the command does not display the contents of the hash map.

- **-header**
  Specifies that the command display the program module’s header. You cannot specify both this argument and the `-release` argument. By default, the command does not display the program module’s header.

- **-line_numbers**
  Specifies that the command display the program module’s machine code as pseudo-assembly code accompanied by source line numbers and object module boundary information. This machine code listing is similar to the listing produced when you specify the `-full` argument with an OPENVOS compiler. If you specify
-line_numbers, the command ignores the -disassemble argument. By default, the command does not display the program module’s source line numbers.

- **-link_map**
  Specifies that the command display a list of all of the program module’s external cross-references, in contrast with the -link_names_map argument, which displays only a condensed list of link names. These cross-references are resolved at run time. By default, the command does not display the program module’s link map.

- **-link_names_map**
  Specifies that the command display a condensed list of the program module’s external cross-reference names. If a cross-reference name appears more than once in the link map, it appears only once in the link names map. By default, the command does not display the program module’s link names map.

- **-module_map**
  Specifies that the command display a list of all of the object modules in a program module, along with their lengths and offsets. By default, the command does not display the program module’s object module map.

- **-object_dirs_map**
  Specifies that the command display the program module’s object directory map. This map lists the path names of all of the object modules’ directories in the program module. The module map specifies the index into the object directory map for each object module. By default, the command does not display the program module’s object directory map.

- **-release**
  Specifies that the command display the release number from the program module header of a command installed from an OpenVOS release tape. You cannot specify both the -release argument and the -header argument. By default, the command does not display the program module release number.

  **Note:** A release number is not defined in the program modules you have compiled and bound.

- **-relocation_map**
  Specifies that the command display the relocation map of a relocatable program module. Relocatable program modules are used for dynamic tasking and kernel loadable applications. To create a relocatable program module, specify the -relocatable argument with the bind command. By default, the command does not display the program module’s relocation map.

- **-string_pool**
  Specifies that the command display the contents of the string pool. The string pool is a collection of character strings referenced by other maps. By default, the command does not display the contents of the string pool.

- **-page number**
  Specifies that the command display a dumped page of machine code or pseudo-assembly code. The page value can range from 0 to 32,767 and is relative to the load point (not to a virtual memory location). Each page is 4096 bytes long.
Use this argument with the -disassemble or -dump argument. If you specify the -page argument but do not specify -disassemble or -dump, the command uses the -dump argument by default.

Explanation
The display_program_module command displays information about the authorship, history, size, and structure of a program module.

An OpenVOS program module (.pm) consists of a header and a series of maps, some of which are optional, produced by the OpenVOS binder, and one or more object modules produced by an OpenVOS compiler. The header and maps are used by the OpenVOS program loader and debugger to execute the program module. The display_program_module command can display the header, all or selected maps, and pseudo-assembly code listings of object modules in the program module.

The program_module_name value does not require a .pm suffix. For example, you can specify the name of a shared library, which does not have a .pm suffix. The display_program_module command looks first for an input file with a .pm suffix, and if that search fails, it looks for an input file with the name specified in program_module_name.

Using the -disassemble and -line_numbers Arguments
The -disassemble and -line_numbers arguments are mutually exclusive arguments that specify that the command display the program module’s machine code as pseudo-assembly code. However, the arguments display the pseudo-assembly code differently. The -line_numbers argument organizes the display of pseudo-assembly code by object module and provides the corresponding source code line numbers. The -disassemble argument displays pseudo-assembly code in the order it will appear in virtual memory. You may find this output harder to interpret, because it does not follow the same order as the source code. However, you can specify the -disassemble argument with the -page argument, and the command will display pseudo-assembly code for the specified page.

Using the -page Argument
Before using this argument, it is helpful to run the command and specify -full to display the program module header and object module map. The header contains load point information, and the object module map contains the location of the code, symbol table, and static regions relative to the load point. This information enables you to select a page to display.

When you specify the value 0 for the -page argument, you are specifying the load point. All -page values are relative to the load point. In the header, the load point is called the user_boundary. For an example of the user_boundary field, see the sample program module header shown in the Sample Output section.
As noted above, the object module map contains the location of the code region relative to the load point. See the Sample Output section for a sample of the object module map for the `strcpy` object module. To calculate the page location of the beginning of this object module, perform the following steps.

1. Convert the `strcpy` code region address (for example, 00E10DC8) from hexadecimal to decimal notation (14749128).

2. Convert the load point address (for example, 00E00000) from hexadecimal to decimal notation (14680064).

3. Subtract the load point address from the code region address, divide the difference by 4096 bytes per page, and round down the quotient to the nearest integer. In this example, the -page value is 16.

The following is another method for calculating the page location of the `strcpy` object module.

```
!display_line (floor (calc '(' 0E10DC8x - 0E00000x ')' / 4096)) 16
```

**Sample Output**

This section describes the sample output of the `display_program_module` command when it is invoked with the `-full` and `-relocation_map` arguments. The `-full` argument causes the `display_program_module` command to display the program module header, the dynamic table, and the following maps.

- “Object Module Map”
- “Object Directory Map”
- “External Variable Map”
- “Link Names Map”
- “Entry Map”

In addition, the `-hash_map` argument displays the “Hash Map,” and the `-relocation_map` argument displays the “Relocation Map.” The following sections describe the header, the dynamic table, and each of these maps.
Header
The header contains auditing and structure information that the binder adds to the program module. The user_name and date_bound fields may help you determine who last changed a program module. The release_name field contains release information about program modules delivered on OpenVOS release tapes. An example program module header follows.

display_program_module testb

Header:

version:                        1
program_name:            testb
binder_version:          bind, Release 17.1.dev.lb
release_name:            Pre-release
pop_version:                    0
processor:                   1280 (pentium4)
processor_family:               7 (IA32)
data_model:                     0 (32-bit)
binder_options:          -compact, -table, paths
system_name:             sw
user_name:               Sales.Smith
date_bound:              10-05-20 08:58:18 est
main_ep.code_addr:       0000208C
main_ep.static_addr:     00014D58
user_boundary:           00002000
max_heap_size:           00000000
max_program_size:        00000000
max_stack_size:          00000000
stack_fence_size:        00001000
n_modules:                     22
n_external_vars:                9
n_link_names:                  23
n_unsnapped_links:             24
n_entries:                      0
n_vm_pages:                    19
n_header_pages:                 1
n_relocations:                  11
module_map_address:      000132B2
module_map_len:          0000065C
ext_vars_map_address:    0001390E
ext_vars_map_len:        0000018C
link_names_map_address:  00013A9A
link_names_map_len:      0000030E
link_map_address:        00013DA8
link_map_len:            00000090
entry_map_address:       00013E38
entry_map_len:           00000000
header_address:          00013F4C
header_len:              00000B84
relocation_map_address:  00013E38

(Continued on next page)
display_program_module

(Continued)

relocation_map_len: 0000009A
high_water_mark: 00000000
string_pool_address: 00013ED2
string_pool_len: 0000006A
obj_dir_map_address: 00013F3C
obj_dir_map_len: 00000010
section_map_file_address: 00000000
section_map_address: 00000000
section_map_len: 00000000
n_sections: 0
dynamic_table_address: 00000000
dynamic_table_len: 00000000
dynsym_map_address: 00000000
dynsym_map_len: 00000000
n_dynsyms: 0
hash_map_address: 00000000
hash_map_len: 00000000
n_hash_buckets: 0
dynref_map_address: 00000000
dynref_map_len: 00000000
n_dynrefs: 0
flags: profile, private_stack, supports_xfn
n_tasks: 1
stack_len: 00010000
xfn_version: 1

Section Info:

<table>
<thead>
<tr>
<th>Paged</th>
<th>address</th>
<th>length</th>
</tr>
</thead>
<tbody>
<tr>
<td>Code</td>
<td>00002000</td>
<td>00009BC8</td>
</tr>
<tr>
<td>Symtab</td>
<td>0000E000</td>
<td>000052B2</td>
</tr>
<tr>
<td>Static</td>
<td>0000C000</td>
<td>00000DC8</td>
</tr>
<tr>
<td>External</td>
<td>0000D000</td>
<td>00000000</td>
</tr>
</tbody>
</table>

block_map_address: 0000B4F8  block_map_len: 000006D0

task_static_len: 00000DC8  GOT address: 0000CD58

In the preceding example, the output xfn_version displays information about extended-names support, with values of 0, 1, or 2:

- The value 0 indicates that extended-names support is disabled.
- The value 1 indicates that version 1 extended-names support is enabled.
- The value 2 indicates that version 2 extended-names support is enabled.

See the description of the bind command for more information about enabling extended-names support for a program module.
**Dynamic Table**

This section displays the contents of the dynamic table. The dynamic table contains information used in dynamically linked programs, such as the shared libraries needed by the specified program module and where it should look for them, the name of the shared library at runtime, and so on. The following example shows a dynamic table.

```
1 DT_NEEDED                   570 libvosposix.1.so
21 DT_DEBUG                  19842 (00004D82)
30 DT_FLAGS                      1 DF_ORIGIN
0 DT_NULL                       0 (00000000)
```

**Object Module Map**

This map contains a list of all object modules in the program module, and the starting address and length of the code, symbol table, and static regions for each object module. The Scn column indicates which portion of memory the object module occupies. Most object modules occupy section 3, paged memory. The Dir Index column is an index to the DTC column in the “Object Directory Map.” The Date Compiled column indicates the date and time each object module was compiled. The following example shows a portion of a module map for a user program.

```
Module Map:

<table>
<thead>
<tr>
<th>Name</th>
<th>Dir Index</th>
<th>Date Compiled</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Scn</td>
<td>Code</td>
</tr>
<tr>
<td></td>
<td>Start</td>
<td>Length</td>
</tr>
<tr>
<td>test</td>
<td>3</td>
<td>00002000</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>test</td>
<td>2</td>
<td>00002070</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>exit</td>
<td>2</td>
<td>00002760</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>clock</td>
<td>2</td>
<td>00002940</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>s_start_c_program_util</td>
<td>2</td>
<td>00002B50</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```


Object Directory Map
This map lists the directories that are specified using the \texttt{bind} command's \texttt{-search} argument or directive, and the links to the object modules that are chased at bind time. The search (\texttt{-search}) directories are always listed first and the non-search (chased at bind time) directories are listed last. The numbers in the DTC column are used in the directory index in the object module map. The following example shows an object directory map.

Object Directory Map:

- 3 search directories
- 0 non-search directories
- 3 directories in all

DTC Directory Path

1  %s1#d03>sales>smith>work
2  %s1#d03>system_17.1.0>c_object_library
3  %s1#d03>system_17.1.0>object_library

External Variable Map
This map lists all external variables used in the program module. The \texttt{Sx} column indicates which portion of memory the object module occupies. Most object modules occupy section 3, paged memory (section 1 is wired memory, and section 2 is the initial section used by the OpenVOS kernel). The \texttt{Length} column indicates the byte length of each variable in hexadecimal. The following example shows an external variable map.

External Variable Map:

<table>
<thead>
<tr>
<th>Name</th>
<th>Sx</th>
<th>Rx</th>
<th>Address</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>GLOBAL_OFFSET_TABLE</em></td>
<td>3</td>
<td>3</td>
<td>00010848</td>
<td>00000000</td>
</tr>
<tr>
<td>s$p11_first_time</td>
<td>3</td>
<td>4</td>
<td>00009004</td>
<td>00000004</td>
</tr>
<tr>
<td>s$plio_debug</td>
<td>3</td>
<td>4</td>
<td>0000900C</td>
<td>00000002</td>
</tr>
<tr>
<td>s$plio_fcb_chain</td>
<td>3</td>
<td>4</td>
<td>00009000</td>
<td>00000004</td>
</tr>
<tr>
<td>s$plio_in_epilogue</td>
<td>3</td>
<td>4</td>
<td>00009008</td>
<td>00000004</td>
</tr>
<tr>
<td>s$su$plio_cursor</td>
<td>3</td>
<td>3</td>
<td>000081DC</td>
<td>00000002</td>
</tr>
<tr>
<td>s$su$plio_in_epilogue</td>
<td>3</td>
<td>3</td>
<td>00008630</td>
<td>00000004</td>
</tr>
<tr>
<td>sysprint</td>
<td>3</td>
<td>3</td>
<td>00008010</td>
<td>000001B8</td>
</tr>
</tbody>
</table>
**Link Names Map**

This map contains a condensed list of all of the program module's external cross-reference names. If a cross-reference name appears more than once in the link map, it appears only once in the link names map. The following example shows a link names map.

```
#   Name
1  s$convert
2  s$write
3  s$close
4  s$error
5  s$detach_port
6  s$add_epilogue_handler
7  s$add_task_epilogue_handler
8  s$open
9  s$create_index
10 s$create_file
11 s$get_object_type
12 s$delete_file
13 s$control
14 s$attach_port
15 s$expand_path
16 s$get_task_id
17 s$is_file_type_a_terminal
18 s$get_port_attachment
19 s$seq_write
20 s$signal_condition
21 s$seq_write_partial
```
Entry Map
This map lists the program module’s entry map. If the program module was not bound with the `-retain_all` argument or directive, the `display_program_module` command displays a message that the entry map is empty. The following example shows an entry map.

<table>
<thead>
<tr>
<th>Name</th>
<th>Code</th>
<th>Static Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>__do_global_ctors</td>
<td>00004900</td>
<td>00026220</td>
</tr>
<tr>
<td>__do_global_dtors</td>
<td>00004890</td>
<td>00026220</td>
</tr>
<tr>
<td>__main</td>
<td>00004990</td>
<td>00026220</td>
</tr>
<tr>
<td>access_mode_name</td>
<td>00015815</td>
<td>00026220</td>
</tr>
<tr>
<td>are_default_open_options_set</td>
<td>00016174</td>
<td>00026220</td>
</tr>
<tr>
<td>are_open_options_set</td>
<td>000160D5</td>
<td>00026220</td>
</tr>
<tr>
<td>baud_rate_code</td>
<td>00015FCC</td>
<td>00026220</td>
</tr>
<tr>
<td>baud_rate_name</td>
<td>00015E73</td>
<td>00026220</td>
</tr>
<tr>
<td>collating_type_name</td>
<td>00014E30</td>
<td>00026220</td>
</tr>
<tr>
<td>comm_device_type</td>
<td>0001571C</td>
<td>00026220</td>
</tr>
<tr>
<td>comm_list_device_type</td>
<td>00015750</td>
<td>00026220</td>
</tr>
<tr>
<td>default_open_options_name</td>
<td>00016549</td>
<td>00026220</td>
</tr>
<tr>
<td>default_open_options_string</td>
<td>00016CAC</td>
<td>00026220</td>
</tr>
<tr>
<td>device_type_name</td>
<td>0001524A</td>
<td>00026220</td>
</tr>
<tr>
<td>device_type_number</td>
<td>0001545C</td>
<td>00026220</td>
</tr>
<tr>
<td>drq_cancel_ack</td>
<td>0000DBAD</td>
<td>00026220</td>
</tr>
<tr>
<td>drq_close</td>
<td>0000D838</td>
<td>00026220</td>
</tr>
<tr>
<td>drq_create</td>
<td>0000D530</td>
<td>00026220</td>
</tr>
<tr>
<td>drq_delete</td>
<td>0000D648</td>
<td>00026220</td>
</tr>
<tr>
<td>drq_open</td>
<td>0000D6C3</td>
<td>00026220</td>
</tr>
<tr>
<td>drq_open_port</td>
<td>0000D7B9</td>
<td>00026220</td>
</tr>
<tr>
<td>drq_recv</td>
<td>0000D9FE</td>
<td>00026220</td>
</tr>
<tr>
<td>drq_send</td>
<td>0000D8C7</td>
<td>00026220</td>
</tr>
<tr>
<td>drq_set_io_time_limit</td>
<td>0000DC09</td>
<td>00026220</td>
</tr>
<tr>
<td>file_type_name</td>
<td>00014F5E</td>
<td>00026220</td>
</tr>
<tr>
<td>file_type_number</td>
<td>000153D8</td>
<td>00026220</td>
</tr>
<tr>
<td>hold_mode_name</td>
<td>00015A85</td>
<td>00026220</td>
</tr>
<tr>
<td>io_type_name</td>
<td>00015BC5</td>
<td>00026220</td>
</tr>
<tr>
<td>locking_mode_name</td>
<td>0001594D</td>
<td>00026220</td>
</tr>
</tbody>
</table>

(Continued on next page)
display_program_module

(Continued)

login_slave_device_type 000156E5 00026220
main 000046BB 00026220
mpx_device_type 000156AE 00026220
open_options_name 000161CB 00026220
open_options_string 000167D5 00026220
pthread_create 000171D0 00026220
pthread_getspecific 000171D0 00026220
pthread_key_create 000171D0 00026220
pthread_mutex_lock 000171D0 00026220
pthread_mutex_unlock 000171D0 00026220
pthread_once 000171D0 00026220
pthread_setspecific 000171D0 00026220
requester_close_queue 0000CB63 00026220
shareable_device_type 00015784 00026220
short_file_type_name 000151C2 00026220
sync_device_type 00015650 00026220
sync_only_device_type 0001567A 00026220
valid_baud_rate_code 0001607B 00026220
valid_baud_rate_name 000160B5 00026220
valid_device_names 000154FB 00026220
valid_device_type 00015398 00026220
valid_file_type 00015223 00026220
valid_io_type 00015E4C 00026220
wait_mode_name 00015CFE 00026220
Dynamic Reference Map
This map lists the program module’s dynamic reference (dynref) map. If no such map exists for the program module, the `display_program_module` command displays a message that the dynref map is empty. The following example shows a portion of a dynamic reference map.

<table>
<thead>
<tr>
<th>Name</th>
<th>Address</th>
<th>Rtype</th>
<th>Roffset</th>
<th>N_args</th>
<th>Flags</th>
</tr>
</thead>
<tbody>
<tr>
<td>fclose</td>
<td>00003010</td>
<td>0</td>
<td>0</td>
<td>-2</td>
<td>elink, entrypoint</td>
</tr>
<tr>
<td>fopen</td>
<td>00003018</td>
<td>0</td>
<td>0</td>
<td>-2</td>
<td>elink, entrypoint</td>
</tr>
<tr>
<td>__main</td>
<td>00003020</td>
<td>0</td>
<td>0</td>
<td>-2</td>
<td>elink, entrypoint</td>
</tr>
<tr>
<td>strcpy</td>
<td>00003028</td>
<td>0</td>
<td>0</td>
<td>-2</td>
<td>elink, entrypoint</td>
</tr>
<tr>
<td>fgets</td>
<td>00003030</td>
<td>0</td>
<td>0</td>
<td>-2</td>
<td>elink, entrypoint</td>
</tr>
<tr>
<td>sprintf</td>
<td>00003038</td>
<td>0</td>
<td>0</td>
<td>-2</td>
<td>elink, entrypoint</td>
</tr>
<tr>
<td>fputs</td>
<td>00003040</td>
<td>0</td>
<td>0</td>
<td>-2</td>
<td>elink, entrypoint</td>
</tr>
<tr>
<td>strnul</td>
<td>00003048</td>
<td>0</td>
<td>0</td>
<td>-2</td>
<td>elink, entrypoint</td>
</tr>
<tr>
<td>strcmp</td>
<td>00003050</td>
<td>0</td>
<td>0</td>
<td>-2</td>
<td>elink, entrypoint</td>
</tr>
<tr>
<td>fprintf</td>
<td>00003058</td>
<td>0</td>
<td>0</td>
<td>-2</td>
<td>elink, entrypoint</td>
</tr>
<tr>
<td>_iob</td>
<td>000003A3</td>
<td>56</td>
<td>208</td>
<td></td>
<td></td>
</tr>
<tr>
<td>_iob</td>
<td>000003C2</td>
<td>56</td>
<td>208</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Dynamic Symbolic Table Map
This map lists the program module’s dynamic symbolic table (dynsym) map. If no such map exists for the program module, the `display_program_module` command displays a message that the dynsym map is empty. The following example shows a portion of a dynamic symbolic table map.

```
# Name Next Address Stat/len Sx Rx N_args Flags
1  _GLOBAL_OFFSET_TABLE_  0 0000B060 00000000 3 3
2  __DWARF2ABBREV_BEGIN__ 0 0000702E 00000000 3 2
3  __DWARF2ABBREV_END__    0 0000702E 00000000 3 2
4  __DWARF2ARANGES_BEGIN__ 0 00007062 00000000 3 2
5  __DWARF2ARANGES_END__   1 00007062 00000000 3 2
```
**Hash Map**

This map lists the contents of the hash map. The dynamic linker uses the hash map to quickly look up symbol definitions in the entry map and the external variable map. To see the hash map, specify the `-hash_map` argument of the `display_program_module` command. The following example shows a hash map.

```plaintext
hash[0]:          9
hash[1]:         15
hash[2]:          7
hash[4]:         20
hash[6]:         19
hash[7]:         16
hash[9]:         17
hash[10]:        21
hash[11]:          8
hash[12]:        22
hash[14]:         12
hash[16]:          2
```

**Relocation Map**

This map lists those regions where code and data will be relocated in a dynamic tasking or kernel-loadable application. If the program module was not bound with the `bind` command’s `-relocatable` argument or directive, then this map is empty. The following example shows a relocation map.

```plaintext
Relocation Map:

<table>
<thead>
<tr>
<th>Reference Section</th>
<th>Reference Region</th>
<th>Reference Offset</th>
<th>Referent Section</th>
<th>Referent Region</th>
<th>Referent Offset</th>
<th>Reloc Length</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Paged</td>
<td>Static</td>
<td>00000004</td>
<td>Paged</td>
<td>Static</td>
<td>00000010</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Static</td>
<td>000001D4</td>
<td>Paged</td>
<td>Static</td>
<td>000001DC</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Static</td>
<td>00000204</td>
<td>Paged</td>
<td>Static</td>
<td>000001DC</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Static</td>
<td>00000214</td>
<td>Paged</td>
<td>Static</td>
<td>000001DC</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Static</td>
<td>0000060C</td>
<td>Paged</td>
<td>Static</td>
<td>00008848</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Static</td>
<td>00000614</td>
<td>Paged</td>
<td>Static</td>
<td>00008848</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Static</td>
<td>0000061C</td>
<td>Paged</td>
<td>Static</td>
<td>00008848</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Static</td>
<td>0000062C</td>
<td>Paged</td>
<td>Static</td>
<td>0000630</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Static</td>
<td>0000064C</td>
<td>Paged</td>
<td>Static</td>
<td>0000630</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>
```
Related Information

For more information about the \texttt{s$monitor} and \texttt{s$monitor\_full} subroutines and their use of the entry map, see the \textit{VOS Transaction Processing Facility Guide} (R215). For more information about the \texttt{analyze\_pc\_samples} and \texttt{harvest\_pc\_samples} commands and their use of the entry map, see the command descriptions in this manual. For more information about the \texttt{bind} command, see the command description in this manual.
**Purpose**

This command displays information about the current usage of a module or set of modules.

**Display Form**

```plaintext
- long: 0
- module: module
- last: 1
```

**Command Line Form**

```plaintext
display_system_usage [-long]

[-module module_name]
[-last period]
```

**Arguments**

- **-long**
  
  Displays a long usage report.

- **-module module_name**
  
  Specifies a module name or star name. The operating system displays usage information for all modules with matching names. By default, the command uses the current module.

- **-last period**
  
  Specifies the time period for the usage figure. Possible values for `period` are 1, 5, 60, and all, to indicate the number of minutes or the entire time the module has run since bootloader. This argument applies only when you select more than one module with `-module`; the command displays system usage in the specified time period for all modules specified. By default, the command displays the system usage during the past minute of usage for the modules specified.
**Explanation**

The `display_system_usage` command displays the following information about each module:

- the name of the module, with its central processing unit type and OpenVOS release number
- the page fault rate of the module
- the CPU usage rate of the module
- the disk I/O rate of the module
- the percentage of idle CPU time of the module
- the percentage of time handling page faults
- the percentage of time handling interrupts
- the percentage of time that the computational cores are busy (or idle)

If you specify the `-long` argument, the `display_system_usage` command presents the information in more detail, showing how much time and disk I/O is performed by user, operating system, and network processes.

If you specify the `display_system_usage` command from a module running an older release that does not know about the new hardware on a module running a later release, the module returns `UNKN` (unknown) in the `CPU Model` field. See Example 2 later in this subroutine description for an example.

**Note:** If a number in the output is too large to fit in the reserved column of information, the system displays the field as asterisks; for example: ******.

ftServer modules may use Intel® Xeon® processors with Hyper-Threading (HT) Technology. This technology groups multiple logical processors within a computational core. Although the logical processors appear to be independent, they must contend for resources within the core, consuming additional processor time. The **CPU minutes** value measures logical processor time (including contention delays). The **CPU Core Busy** and **CPU Core Idle** values measure computational core activity, a more meaningful measure of how busy (or idle) the system is at any point in time.

The `display_system_usage` command reports the effective number of CPUs for each reporting interval. This line is labeled **N CPUs**. It reports the total of all CPU time used for useful work (100% minus the idle time) as **Total Reported CPU**.
Examples

Example 1.

In the following example, the command displays system usage information for a single ftServer module.

display_system_usage -module %sw#m1
Usage statistics for module %sw#m1, G94330, OpenVOS Release 17.2.0
All 115.3 hours (4.8 days) up.

----CPU----- Last Min Last 5 Min Last Hour All Up Time
CPU minutes 0.87 21.7% 4.38 21.9% 53.62 22.3% 1578.15 5.7%
Min at PF 0.11 2.7% 0.55 2.7% 6.88 2.9% 199.79 0.7%
Idle 2.90 72.5% 14.45 72.2% 171.45 71.4% 25586.46 92.4%
Other 0.01 0.1% 0.03 0.1% 0.41 0.2% 22.30 0.1%
N CPUs 4.00 4.00 4.00 3.99
Total Reported CPU 1.10 27.5% 5.55 27.8% 68.59 28.6% 2099.22 7.6%
CPU Core Busy 1.08 53.9% 5.38 53.8% 65.76 54.8% 2031.10 14.7%
CPU Core Idle 0.92 46.1% 4.62 46.2% 54.25 45.2% 11811.73 85.3%

--I/O Rates--
Page faults,/sec 253766 4229 1277476 4258 1527898 4244 47445572 1142
File IO, /sec 194491 3242 980949 3270 11582035 3217 310157678 747
Disk IO, /sec 195579 3260 987984 3293 11760490 3267 336481731 810

--INT Rates--
Ints, /sec 186375 3106 940000 3133 13331420 3703 1004591432 2419
Int. Time 0.12 2.9% 0.60 3.0% 7.68 3.2% 298.98 1.1%

The output displays statistics for active times (the line CPU Core Busy) and idle times (the lines CPU Core Idle and Idle) as minutes and hours as well as percentages of minutes and hours. The number of hours in the rightmost column represents the time since the system was booted.
Example 2.
In the following example, the `display_system_usage` command is issued from m7, a module running VOS Release 14.7.0, while module m8 is running VOS Release 15.0.0. Because m7 does not recognize the m8 CPU model, `display_system_usage` displays UNKN in the CPU Model field.

```
display_system_usage -module #m*
```

Usage statistics for module %sales#m*, VOS Release 14.7.0am (on %sales#m7)
Data from last minute

<table>
<thead>
<tr>
<th>-----CPU-----</th>
<th>m7</th>
<th>m8</th>
</tr>
</thead>
<tbody>
<tr>
<td>Module Up/hrs</td>
<td>6506</td>
<td>144</td>
</tr>
<tr>
<td>CPU Model</td>
<td>G313</td>
<td>UNKN</td>
</tr>
<tr>
<td>CPU usage %</td>
<td>0.4</td>
<td>1.3</td>
</tr>
<tr>
<td>Time at PF %</td>
<td>0.0</td>
<td>0.1</td>
</tr>
<tr>
<td>Idle %</td>
<td>98.7</td>
<td>98.4</td>
</tr>
<tr>
<td>Other %</td>
<td>0.0</td>
<td>0.0</td>
</tr>
</tbody>
</table>

--I/O RATES--
Page faults/sec 2.1 40.0
File IOs/sec 0.8 10.0
Disk IOs/sec 1.9 22.9

Example 3.
The following example shows the `display_system_usage` output for a single ftServer module.

```
display_system_usage
```

Usage statistics for module %sw#m102, G92200, OpenVOS Release 17.0.0
All 16.3 hours up.

<table>
<thead>
<tr>
<th>-----CPU-----</th>
<th>Last Min</th>
<th>Last 5 Min</th>
<th>Last Hour</th>
<th>All Up Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU minutes</td>
<td>0.01 0.1%</td>
<td>0.04 0.1%</td>
<td>0.55 0.1%</td>
<td>8.52 0.1%</td>
</tr>
<tr>
<td>Min at PF</td>
<td>0.00 0.0%</td>
<td>0.00 0.0%</td>
<td>0.00 0.0%</td>
<td>0.06 0.0%</td>
</tr>
<tr>
<td>Idle</td>
<td>7.97 99.6%</td>
<td>39.85 99.6%</td>
<td>478.10 99.6%</td>
<td>7819.39 99.6%</td>
</tr>
<tr>
<td>Other</td>
<td>0.00 0.0%</td>
<td>0.00 0.0%</td>
<td>0.03 0.0%</td>
<td>0.31 0.0%</td>
</tr>
<tr>
<td>N CPUs</td>
<td>8.00</td>
<td>8.00</td>
<td>8.00</td>
<td>7.99</td>
</tr>
<tr>
<td>Total Reported CPU</td>
<td>0.03 0.4%</td>
<td>0.15 0.4%</td>
<td>1.93 0.4%</td>
<td>30.19 0.4%</td>
</tr>
<tr>
<td>CPU Core Busy</td>
<td>0.04 0.9%</td>
<td>0.19 0.9%</td>
<td>2.33 1.0%</td>
<td>36.76 0.9%</td>
</tr>
<tr>
<td>CPU Core Idle</td>
<td>3.96 99.1%</td>
<td>19.81 99.1%</td>
<td>237.68 99.0%</td>
<td>3888.00 99.1%</td>
</tr>
</tbody>
</table>

--I/O Rates--
Page faults,/sec 0 0.0 0 0.0 1415 0.4 15607 0.3
File IO, /sec 4 0.1 312 1.0 10379 2.9 39080 0.7
Disk IO, /sec 6 0.1 315 1.1 11039 3.1 45895 0.8

(Continued on next page)
Example 4.
The following example shows display_system_usage output for a multi-module system. Note that some modules displayed in the following output do not support the CPU Core Busy or CPU Core Idle values; therefore, those output lines are blank.

!display_system_usage -module *

Usage statistics for module %es#*, VOS Release 17.0.0 (on %es#m3)
Data from last minute.

-----CPU----               m2        m3       m10       m12       m13       m16
Module Up/hrs             172      2359      1631       646       262       173
CPU Model                G334      G748      G332      G334      G321      G334
CPU usage %               0.0       1.1       0.2       0.1       0.0       0.0
Time at PF %              0.0       0.2       0.0       0.0       0.0       0.0
Other %                  99.9      97.8      99.6      99.8      99.8      99.9
Idle %                   99.9      97.8      99.6      99.8      99.8      99.9
N CPUs                    2.0       2.0       1.0       2.0       2.0       2.0
Tot Reported CPU %        0.1       2.2       0.4       0.2       0.2       0.1

--I/O RATES--
Page faults/sec           0.0       5.7       0.0       0.0       0.0       0.0
File IOs/sec              0.0       1.5       0.0       0.2       0.0       0.0
Disk IOs/sec              0.0       1.5       0.0       0.3       0.0       0.0

-----CPU----             m100      m101      m102      m104      m107      m108
Module Up/hrs            67.7       166      66.9       165      93.7       289
CPU Model              G92200    G92900    G92200    G94220    G94340    G94330
CPU usage %               0.0       0.2       0.0       1.4       0.1       0.2
Time at PF %              0.0       0.0       0.0       0.0       0.0       0.0
Other %                  99.5      99.4      99.7      98.1      99.8      99.8
Idle %                   99.5      99.4      99.7      98.0      99.8      99.8
N CPUs                    8.0       4.0       8.0       2.0       8.0       4.0
Tot Reported CPU %        0.5       0.6       0.3      2.0       0.2       0.2
CPU Core Busy %           1.2       1.6       0.9       0.5
CPU Core Idle %          98.8      98.4      99.1      99.5

--I/O RATES--
Page faults/sec           6.7       0.0       0.0       2.8       2.9       0.0
File IOs/sec              5.5       0.0       0.4       2.8       2.8       0.0
Disk IOs/sec              8.3       0.0       0.4       4.1       4.3       0.0

(Continued on next page)
### CPU Usage

<table>
<thead>
<tr>
<th>Category</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Module Up/hrs</td>
<td>67.6</td>
</tr>
<tr>
<td>CPU Model</td>
<td>G94330</td>
</tr>
<tr>
<td>CPU usage %</td>
<td>0.3</td>
</tr>
<tr>
<td>Time at PF %</td>
<td>0.0</td>
</tr>
<tr>
<td>Other %</td>
<td>99.6</td>
</tr>
<tr>
<td>Idle %</td>
<td>99.6</td>
</tr>
<tr>
<td>N CPUs</td>
<td>4.0</td>
</tr>
<tr>
<td>Tot Reported CPU %</td>
<td>0.4</td>
</tr>
</tbody>
</table>

### I/O Rates

<table>
<thead>
<tr>
<th>Category</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Page faults/sec</td>
<td>9.4</td>
</tr>
<tr>
<td>File IOs/sec</td>
<td>10.2</td>
</tr>
<tr>
<td>Disk IOs/sec</td>
<td>14.3</td>
</tr>
</tbody>
</table>

### INT Rates

<table>
<thead>
<tr>
<th>Category</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ints/sec</td>
<td>828</td>
</tr>
<tr>
<td>Int Time %</td>
<td>0.0</td>
</tr>
</tbody>
</table>
display_tape_params

Purpose
This command displays the default, user, and actual parameters of the tape drive, tape mount, and tape file.

Display Form

--- display_tape_params ---

<table>
<thead>
<tr>
<th>tape_device_or_port_name:</th>
<th>display_tape_params</th>
</tr>
</thead>
<tbody>
<tr>
<td>-default:</td>
<td>yes</td>
</tr>
<tr>
<td>-user:</td>
<td>yes</td>
</tr>
<tr>
<td>-actual:</td>
<td>yes</td>
</tr>
</tbody>
</table>

Command Line Form

display_tape_params tape_device_or_port_name
                      [-no_default]
                      [-no_user]
                      [-no_actual]

Arguments

► tape_device_or_port_name  Required
The name of the tape device, or the name of the port attached to the tape drive, for which the default, user, and actual parameters are to be displayed.

► -no_default  CYCLE
Omits displaying the default tape parameter values used by the tape commands. The initial port attachment determines the default tape drive, tape mount, and tape file parameters. By default, the command displays the default tape parameter values.

► -no_user  CYCLE
Omits displaying the user tape parameter values used by the commands that create and write a tape. The set_tape_drive_params, set_tape_mount_params, and set_tape_file_params commands set the user tape drive, tape mount, and tape file parameters. The user tape parameters override the default tape parameters. By default, the command displays the user tape parameter values.

► -no_actual  CYCLE
Omits displaying the actual tape parameter values used by the tape commands. The actual tape drive, mount, and file parameters consist of the user values overlaid on the default values, and the default values that are not overlaid by user values. The actual values of tape mount and tape file parameters can also include values from the tape
label. The user tape parameters override the default tape parameters. The actual values are used for every tape operation. By default, the command displays the actual tape parameter values.

**Explanation**

To display the tape parameters for a tape drive attached to a port, issue the `display_tape_params` command with either the tape device name or port name as an argument.

The `display_tape_params` command does not implicitly attach a port or mount a tape. You must explicitly attach a port with `attach_port` before you use `display_tape_params`. Optionally, you can also mount a tape with `mount_tape` before you use `display_tape_params`. Once the port is attached, you can specify either the tape device or port name for `tape_device_or_port_name`, as convenient.

The output of `display_tape_params` contains data about the attached tape drive, as shown in the Example, which follows.

**Note:** If you specify the `display_tape_params` command for a SCSI tape drive, ignore the Tape Density parameter; it is inaccurate.

The command output contains three sections: `DEFAULT TAPE PARAMETERS`, `USER TAPE PARAMETERS`, and `ACTUAL TAPE PARAMETERS`. You can determine whether any of these sections is displayed by setting the `-no_default`, `-no_user`, or `-no_actual` argument.

The initial port attachment determines the tape parameter values that your terminal screen displays. You can change any of these values by issuing the `set_tape_drive_params`, `set_tape_mount_params`, or `set_tape_file_params` command. If you do not change any initial values with these commands, the output of `display_tape_params` shows the `USER_TAPE_PARAMETERS` as NOT SET or N/S. If the tape is not mounted, the actual tape mount and file parameters are not displayed. If the tape is mounted, but a tape file is not open, the actual file parameters are not displayed.

The tape drive parameters are in effect as long as the port is attached, unless you change the parameters again.

If you use the `set_tape_mount_params` or `mount_tape` command, the mount parameters are in effect only for the next mount.

If you use the `set_tape_file_params` command, the file parameters are in effect for the next open file or files, as specified by the `-file_id` argument of the command.

For more information, see Appendix A, “Setting and Displaying Tape Parameter Values.” For information about tape mounting, see Explanation in the `mount_tape` command description.
Example

To display the tape parameters for a tape drive attached to the port \texttt{t\_port}, type the following command.

\begin{verbatim}
display_tape_params t_port
\end{verbatim}

The output is displayed in the following format.

\begin{verbatim}
*** DEFAULT TAPE PARAMETERS ***

DRIVE: Disposition: reread
Reel Retention: dismount on detach
Multivolume Default: yes
Message: Notify Sys Admin

MOUNT: Volume ID: NOT SET
Owner ID: NOT SET
Tape Format: ansi
Tape Density: 1600
Cartridge No: 0
Access Rights: read/write

FILE: File ID: NOT SET
File Number: NOT SET
Expiration Date: NOT SET
Translation: NOT SET

Open Type: Create Create Create
Fixed Relative Sequential Input Append

File Format: \texttt{vb vb f f}
Block Length: \begin{tabular}{cccc}
80 & 4096 & 4096 & 80 & 80
\end{tabular}
Record Length: \begin{tabular}{cccc}
80 & 4092 & 4092 & 80 & 80
\end{tabular}
Blocking Factor:

*** USER TAPE PARAMETERS ***

DRIVE: Disposition: reread
Reel Retention: NOT SET
Multivolume Default: NOT SET
Message: Notify Sys Admin

MOUNT: Volume ID: NOT SET
Owner ID: NOT SET
Tape Format: NOT SET
Tape Density: NOT SET
Cartridge No: NOT SET
Access Rights: NOT SET

FILE: File ID: NOT SET
File Number: NOT SET
Expiration Date: NOT SET
Translation: NOT SET

(Continued on next page)
\end{verbatim}
(Continued)

<table>
<thead>
<tr>
<th>Open Type:</th>
<th>Create</th>
<th>Create</th>
<th>Create</th>
<th>Fixed</th>
<th>Relative</th>
<th>Sequential</th>
<th>Input</th>
<th>Append</th>
</tr>
</thead>
<tbody>
<tr>
<td>File Format:</td>
<td>N/S</td>
<td>N/S</td>
<td>N/S</td>
<td>N/S</td>
<td>N/S</td>
<td>N/S</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Block Length:</td>
<td>NOT SET</td>
<td>NOT SET</td>
<td>NOT SET</td>
<td>NOT SET</td>
<td>NOT SET</td>
<td>NOT SET</td>
<td>NOT SET</td>
<td>NOT SET</td>
</tr>
<tr>
<td>Record Length:</td>
<td>NOT SET</td>
<td>NOT SET</td>
<td>NOT SET</td>
<td>NOT SET</td>
<td>NOT SET</td>
<td>NOT SET</td>
<td>NOT SET</td>
<td>NOT SET</td>
</tr>
<tr>
<td>Blocking Factor:</td>
<td>NOT SET</td>
<td>NOT SET</td>
<td>NOT SET</td>
<td>NOT SET</td>
<td>NOT SET</td>
<td>NOT SET</td>
<td>NOT SET</td>
<td>NOT SET</td>
</tr>
</tbody>
</table>

*** ACTUAL TAPE PARAMETERS ***

DRIVE: Disposition: reread
Reel Retention: dismount on detach
Multivolume Default: yes
Message: Notify Sys Admin

Tape is not mounted. MOUNT and FILE Parameters are not valid for this state.

Related Information

See also the command descriptions of copy_tape, create_tape_volumes, dismount_tape, dump_tape, list_save_tape, mount_tape, position_tape, read_tape, restore_object, set_second_tape, set_tape_drive_params, set_tape_mount_params, set_tape_file_params, verify_save, and write_tape.
Purpose

This command displays the current parameters for your terminal.

Display Form

```
------------------------- display_terminal_parameters -------------------------
No arguments required. Press ENTER to continue.
```

Command Line Form

display_terminal_parameters

Explanation

The `display_terminal_parameters` command displays the parameters set by the `set_terminal_parameters` command. For example:

```
Modes:interrupt_key_enabled,display_enable,break_enabled,output_flow
Line length:    80
Screen size:    24
Pause lines:    23
Prompt chars:   ''
Continue chars: '+'
Pause chars:    '--PAUSE--'
Escape char:    '
Flow chars:     ''11'13'
Terminal type:  v102a
Tabs:6,11,16,21,26,31,36,41,46,51,56,61,66,71,76,81,86,91,96,101,106
Baud:           19200
Device name:    t1.6
```

Related Information

See the description of the `set_terminal_parameters` command for a complete discussion of these parameters.
display_usb_info

Purpose
This command displays information about a USB device.

Display Form

```
device_name:  no
-long:   no
```

Command Line Form

```
display_usb_info [device_name]
[-long]
```

Arguments

- `device_name`
The name of the USB device for which the information is to be displayed. If you omit this argument, the command displays information about all defined USB devices.

- `[-long]`
  Displays a long report on the device. By default, the command displays a short report.

Explanation
The `display_usb_info` command displays information about a USB device.

Examples
```
display_usb_info
```

```
#usb$ms.2
Vendor name      MATSHITA
Product name     DVD-RAM UJ870BJ
Media Size       Total  Sector  Total Sectors  Size  Bytes
                1704    2048    3489792
```

---

OpenVOS Commands Reference Manual (R098)  2-393
dump_file

Purpose
This command dumps the contents of the specified file for debugging purposes or the contents of an index to the specified file.

Display Form

```
- dump_file ----------------------------------
  file_path_name: ...
  -index:
  -output_path:
  -from: 1
  -to: 2147483647
  -ebcdic: no
  -bcd: no
  -brief: no
  -header: yes
```

Command Line Form

```
dump_file file_path_name
  [-index index_name]
  [-output_path output_path_name]
  [-from first_block]
  [-to largest_block]
  [-ebcdic]
  [-bcd]
  [-brief]
  [-no_header]
```

Arguments

- **file_path_name** Required
  The path name of the file to be dumped.

- **-index index_name**
  Dumps index_name instead of the file.

- **-output_path output_path_name**
  Specifies the file to which the dumped output is to be appended. By default, the command writes the output to the default output port.
dump_file

- **-from first_block**
  Specifies the block number at which the dump is to begin. By default, the dump begins at block 1.

- **-to largest_block**
  Specifies the block number at which the dump is to end. If you omit -to and -from, the dump ends at the largest possible block in the file. If you do not specify the last block of the dump with -to but you specify -from, the command dumps one block.

- **-ebcdic**
  Interprets the data in the file as EBCDIC characters. By default, the data is interpreted as ASCII characters.

- **-bcd**
  Interprets the data in the file as binary coded decimal (BCD) characters. By default, the data is interpreted as ASCII characters.

- **-brief**
  Avoids displaying blocks that contain all 0s (for stream files) or all 1s (for other file types). By default, the command displays all blocks.

- **-no_header**
  Suppresses the normal header, which displays the path name and current date. Use this argument when generating output to compare with expected output, as you may not want to see the current date as a difference. By default (yes), the command displays the normal header.

**Explanation**

The `dump_file` command dumps the contents of the specified file for debugging purposes, in both hexadecimal and ASCII, with the ASCII text in the rightmost column.

**Access Requirements**

You need read access to a file to dump its contents.

**Related Information**

See also the command descriptions of `compare_files`, `copy_file`, `create_file`, `delete_file`, `display_file_status`, `locate_files`, `move_file`, `set_file_allocation`, and `truncate_file`. 

OpenVOS Commands Reference Manual (R098)  2-395
**Purpose**

This command dumps one or more records in a fixed, sequential, relative, or stream file.

**Display Form**

```
--------------------------------- dump_record ----------------------------------
path_name:
-from: 1
-for:
-match:
```

**Command Line Form**

```
dump_record path_name
    [-from first_record_number]
    [-for record_count]
    [-match string]
```

**Arguments**

- **path_name**
  Required
  The path name of the file from which the records are to be dumped.

- **-from first_record_number**
  Specifies the number of the first record to be dumped. By default, the dump begins with
  the first record in the file.

- **-for record_count**
  Specifies the number of records to be dumped. By default, all records are dumped.

- **-match string**
  Specifies a character string in the record. Specifying a string causes only records
  containing the string to be dumped.

**Explanation**

The `dump_record` command dumps one or more records in a fixed, sequential, relative, or
stream file. If you specify `-from` and `-for`, you can specify the first record to be dumped and
the number of subsequent records to dump. If you specify `-match`, you can dump only
records containing the specified string.
dump_tape

Purpose

This command dumps files from a tape.

Display Form

```
---------------------------------- dump_tape -----------------------------------
tape_device_or_port_name: [ -first_file: 1 
- file_count: 32767 
-first_record: 1 
-last_record: 32767 
-ebcdic: no 
-bcd: no 
-initial_rewind: yes 
-stop_on_eot: yes 
-stop_on_error: yes 
-dump: yes 
-hex: yes
```

Command Line Form

```
dump_tape tape_device_or_port_name 
[ -first_file first_file_number ] 
[ -file_count file_count ] 
[ -first_record first_record_number ] 
[ -last_record last_record_number ] 
[ -ebcdic ] 
[ -bcd ] 
[ -no_initial_rewind ] 
[ -no_stop_on_eot ] 
[ -no_stop_on_error ] 
[ -no_dump ] 
[ -no_hex ]
```

Arguments

- **tape_device_or_port_name**  
  Required  
  The name of the tape device, or the name of the port attached to the tape drive, holding the tape from which files are to be dumped.

- **-first_file first_file_number**  
  Specifies the file number of the first file to dump. By default, the dump begins with file number 1.
**dump_tape**

- **-file_count file_count**
  Specifies the number of files to dump. By default, the dump begins with file number 1 or the `first_file_number` and dumps all subsequent files on the tape.

- **-first_record first_record_number**
  Specifies the first record of the selected file to dump. By default, the dump begins with the first record of the file.

- **-last_record last_record_number**
  Specifies the last record of the selected file to dump. By default, the dump continues through the last record of the file.

- **-ebcdic**
  Dumps a tape written in EBCDIC code. By default, the data on the tape is interpreted as ASCII characters.

- **-bcd**
  Dumps a tape written in binary coded decimal (BCD) code. By default, the data is interpreted as ASCII characters.

- **-no_initial_rewind**
  Dumps the tape without first rewinding it. By default, the tape rewinds before the dump begins.

- **-no_stop_on_eot**
  Permits reading a tape past the end-of-tape mark. By default, the dump ends when processing encounters an end-of-tape mark. An end-of-tape mark is two consecutive tape marks.

- **-no_stop_on_error**
  Permits reading a tape past errors. By default, the dump ends when processing encounters an error.

- **-no_dump**
  Suppresses dumping of the data on the tape. If you specify `-no_dump`, the command displays only identifying information for each selected record, including its length.

- **-no_hex**
  Dumps only the character representation of the data, suppressing the accompanying hexadecimal information.

### Explanation

The `dump_tape` command dumps the contents of a tape.

Before using the `dump_tape` command, you must load the tape whose contents are to be dumped. If you do not attach a port with the `attach_port` command, the `dump_tape` command implicitly attaches a port. The `dump_tape` command does not require that a tape be mounted. When execution is completed, if `dump_tape` implicitly attached a port, it implicitly detaches the port.
Access Requirements

By default, you have write access to all devices. If your system administrator restricts access to a device, you need read access to read from tapes, or write access to read from and write to tapes.

Related Information

See also the command descriptions of `copy_tape`, `create_tape_volumes`, `dismount_tape`, `display_tape_params`, `list_save_tape`, `mount_tape`, `position_tape`, `read_tape`, `restore_object`, `set_second_tape`, `set_tape_drive_params`, `set_tape_mount_params`, `set_tape_file_params`, `verify_save`, and `write_tape`. 
Purpose

This command invokes the screen editor.

Display Form

```
------------------------------------- edit -------------------------------------
text_file_names:  
-buffer_names:  
-names_shown:  
-read_only_buffers:  
-num_windows:  
-window_divisions:  
-initial_window:  
-start_at_bottom:  
-dictionary:  
-keystrokes: yes  
-initial_shorthand:  
-shorthand_file: yes  
-backup:  
-backup_name:  
-silent_overwrite: no
```

Command Line Form

```
edit [text_file_names]...  
[-buffer_names buffer_name...]  
[-names_shown name_shown...]  
[-read_only_buffers buffer_number...]  
[-num_windows number_of_windows]  
[-window_divisions division_location...]  
[-initial_window window_number]  
[-start_at_bottom buffer_number...]  
[-dictionary dictionary_name]  
[-no_keystrokes]  
[-initial_shorthand shorthand]  
[-shorthand_file shorthand_file_name]  
[-no_backup]  
[-backup_name backup_file_name]  
[-silent_overwrite]
```
Arguments

- **text_file_names**
  One or more names or star names of text files to be placed in editor buffers. If you specify a file that does not exist, the editor creates an empty buffer with the name of the file you specified. By default, the editor creates an empty buffer. The character set of the file can be either ASCII or Latin alphabet No. 1.

- **-buffer_names buffer_name**
  Specifies a name to be assigned to an editor buffer. You can give as many buffer names as there are files in the `text_file_names` list. The editor assigns the _n_th name in `-buffer_names` to the buffer for the _n_ th file in the list. If you specify fewer buffer names than there are files, the editor assigns the file name of any extra file to its buffer as a buffer name. If you specify more buffer names, the editor reports an error. You can specify the null string (`''`) as a buffer name. In this case, the editor does not assign a buffer name to the buffer for the corresponding file. By default, the editor assigns no name to the buffer for the first file in the list, and assigns the file name for any subsequent file to its buffer as a file name.

- **-names_shown name_shown**
  Specifies a name to be displayed for an editor buffer instead of its full path name. You can give as many shown names as there are files in the `text_file_names` list. The editor displays the _n_ th name in `-names_shown` for the buffer for the _n_ th file in the list. If you specify fewer shown names than there are files, the editor displays the full path name of the buffer for each extra file. If you specify more shown names, the editor reports an error. You can specify the null string (`''`) as a shown name. In this case, the editor displays the full path name of the buffer for the corresponding file. By default, the editor displays the full path name of each buffer.

- **-read_only_buffers buffer_number**
  Specifies one or more buffers that are to contain text accessible for reading only. The editor assigns the buffer number _n_ to the buffer that corresponds to the _n_ th file in the `text_file_names` list. You specify a buffer for read-only access by giving its buffer number. By default, you have write access to all buffers.

- **-num_windows number_of_windows**
  Specifies the number of horizontal windows to open on the terminal screen. The editor assigns buffers to windows in the order of the file names given in the `text_file_names` list. If `number_of_windows` is greater than the number of files, the extra windows are empty. By default, one window is created, which displays the buffer corresponding to the first name in the `text_file_names` list. You cannot specify both `-num_windows` and `-window_divisions` in the same command.
-window_divisions division_location
  Specifies the location of one or more divisions that separate editor screen into windows. A division can be vertical or horizontal. To specify a vertical division, give the window number, the value v or V, and the number of the column where the division is to appear (1v34). To specify a horizontal division, give the window number, the value h or H, and the number of the row where the division is to appear (2h12). Windows are numbered from left to right and from top to bottom. A single screen may be divided into windows both horizontally and vertically. Note that the number of divisions is always one fewer than the number of windows opened. You cannot specify both -window_divisions and -num_windows in the same command.

-initial_window window_number
  Specifies the window_number in which the cursor first appears after you invoke the editor. By default, the cursor first appears in the window of the first file displayed in the text_file_names list.

-start_at_bottom buffer_number
  Specifies the number or numbers of one or more buffers to display, with the cursor starting on the last line rather than the first line of the buffer. The last line of the buffer appears in the middle line of the window.

-dictionary dictionary_name
  Specifies a user dictionary that the editor checks when you issue the “Verify spelling” editor request. You can specify this argument only if the current system has the system dictionary. If you use this argument, the editor checks for misspelled words in the specified user dictionary and then, if necessary, in the system dictionary. By default, the editor checks only the system dictionary.

-no_keystrokes
  Allows editing of a file without creating a keystrokes file. By default, the editor creates or overwrites a keystrokes file in your home directory. The file contains the sequence of editor requests you make and the characters you insert in the edited text. With a keystrokes file, you can usually recover from editing mistakes.

  If you invoke the command from a main process and omit -no_keystrokes, the keystrokes file is named _edit.terminal_name, where terminal_name is the device name of the current terminal. You can interrupt this editing session to enter a subprocess. If you invoke the edit command without -no_keystrokes while you are in the subprocess, the name of the new keystrokes file is the same as the original keystrokes file, except that it acquires the suffix .1. That is, its name is _edit.terminal_name.1. If you create another keystrokes file during a second-level subprocess, its name has the suffix .2; the names of keystrokes files created during further subprocesses have corresponding suffixes.

  If you are not editing from a terminal, the editor does not create a keystrokes file. For example, a keystrokes file is not created when the editor requests come from a command macro.
- initial_shorthand shorthand
  Executes the shorthand shorthand, which must be a single character, before editing
  begins. The shorthand must be in the shorthand file specified by or implied in
  -shorthand_file. By default, the editor does not execute an initial shorthand.

- shorthand_file shorthand_file_name
  Uses the shorthand definitions in the file shorthand_file_name. A shorthand file
  consists of a series of edit requests that will be replaced by a single character. In an
  editing session you can execute this series of requests by pressing the shorthand key and
  the single character that is defined in the shorthand file. By default, the editor uses the
  shorthand definitions in a file in your home directory named
  shorthand_definitions. If shorthand_file_name does not exist or you do not
  specify this argument and shorthand_definitions does not exist, then the editor
  creates a shorthand file with the name.

- no_backup
  Allows editing without creating backup copies of any file the editor reads in to any
  editor buffer and then writes out to the file. By default, the editor creates a backup copy
  of every file that it reads in to a buffer. The backup is created the first time you write
  out the buffer to the file. The path name of the backup copy is the same as the name of
  the original file except that the editor adds the suffix .backup. You cannot specify
  -no_backup and -backup_name in the same command.

- backup_name backup_file_name
  Names a backup copy of the edited files of the text_file_names list with a name in
  -backup_name. You can specify the null string ('') as a backup file name. In this
  case, the editor creates a backup copy of the corresponding file the first time you write
  out the contents of the buffer to that file. The path name of the backup file is the file
  name from the text_file_names list with the suffix .backup appended. By default,
  the editor creates and assigns backup files as if the backup file name specified for each
  file is the null string. You cannot specify -backup_name and -no_backup in the same
  command.

- silent_overwrite
  Overwrites the file silently when you explicitly specify an output path name. By
  default, the editor asks you before writing the file.

Explanation

The edit command calls the screen editor. After you issue the edit command, your process
is at the editor request level. At request level, you can make any of the several requests
described in the VOS Word Processing User’s Guide (R006). One request, the request to quit
the editor, returns your process to command level. To issue the quit request, press the (MENU)
key and then type the letter q.

If you give the path name of a file when you issue the edit command, the editor reads in the
file to the default buffer and displays the first 20 lines of the buffer. Otherwise, the buffer is
empty when you begin.
Before you edit the text file, you must set the shift mode to none and specify a character set of none, ascii, or latin_1. Use the create_file or convert_text_file command to establish the shift mode and character set of the file.

**Note:** If you are working with very large files, there is a limit to the file size that can be read. This limit is determined by the size of the file, the number of format lines it contains, and the size of the program modules that are also being loaded. You must be aware if your file is approaching this limit and break it into smaller files before this occurs; once a file surpasses that size, you cannot access it through the editor.

**Access Requirements**

You need read access to an existing file to read it into an editor buffer and write access to overwrite it with the contents of an editor buffer.

**Related Information**

For a complete description of the OpenVOS screen editor and editor requests, see the *VOS Word Processing User’s Guide* (R006).
edit_form

Purpose
This command invokes the old Forms Editor.

Display Form

```
---------------------------------- edit_form -----------------------------------
input_path:               form_path:
-into:                   no
-prefix:                  no
-library:                accept_field_definitions
-edit:                   yes
-backup:                 yes
-force_write:            no
-basic:                   no
-cobol:                   no
-fortran:                no
-pascal:                 no
-pl1:                    no
-pl1_template:           yes
-c:                      no
-processor:              default
-mapping_rules:          default
-sort_into_by_alignment: yes
-produce_symtab:        yes
```
Command Line Form

```
edit_form input_path
 [form_path]
 [-into]
 [-prefix]
 [library field_definitions_directory_name]
 [-no_edit]
 [-no_backup]
 [-force_write]
 [-basic]
 [-cobol]
 [-fortran]
 [-pascal]
 [-pl1]
 [-no_pl1_template]
 [-c]
 [-processor processor_string]
 [-mapping_rules mapping_rules]
 [-no_sort_into_by_alignment]
 [-no_produce_symtab]
```

Arguments

* **input_path**
  The path name of a form definition file. The path name cannot be an extended name. A .form file is a programming source file. If the file exists, its name must have the suffix .form. You can omit the suffix when specifying the name in the command. If the file does not exist, the Forms Editor behaves as if the file existed but was empty. The name of the form definition file, without the suffix .form, becomes the name of the form. (A form name should not exceed 15 characters; otherwise, the names of some automatically generated include files may exceed 32 characters and be truncated.) You cannot edit or compile a .form file on a release that is earlier than the release on which the .form file was originally created unless the .form file contains only features supported by the earlier release.

Note: Do not give a form the same name as the program that displays it: both a form and its related program require uniquely named object modules.

* **form_path**
  Writes the edited form definition to the file form_path. By default, the Forms Editor writes the form definition file to the file input_path (but in the current directory), when you write out the form. If the specified file does not exist when you write out the form, the editor creates it. The form_path cannot be an extended name.

* **-into**
  Creates a field-values file for each programming language specified by the language arguments. The Forms Editor names the field-values file (an include file) form_name.incl.language and puts the file in the current directory. When revising a form, the command uses the already-specified language’s include files. You can override the existing value with the Forms Editor MENU-S request.
- **prefix**
  Adds a prefix to the name of each field identifier name in any field IDs file that the editor generates. The prefix is the name of the form followed by an underline. The editor also adds the prefix to each variable name in any OpenVOS FORTRAN field-values file that it generates.

- **library field_definitions_directory_name**
  Specifies a directory to be searched for field definition files. The editor searches the directory for field definition files and writes out field definition files to the directory. By default, the editor searches a subdirectory of your current directory named accept_field_definitions. If the directory you specify, either directly or by default, does not exist when the editor is ready to write out a field definition file, it creates the directory.

- **no_edit**
  Creates a new object module and language include files from an existing form definition file without editing the form. (If you specify -force_write as well, the editor writes a new form definition file.) By specifying -no_edit, you can run the Forms Editor in a batch process or started process. By default, the Forms Editor reads the form definition file, displays the form, and lets you edit it.

- **no_backup**
  Suppresses the creation of a backup file for input_path. If you do not use -no_backup, and the input_path and form_path files are in the same directory, the Forms Editor renames the old file and gives it the name of the input_path file (including its suffix .form), with the suffix .backup added. The backup file is created each time you write out the form with the MENU-W request; it replaces a previous backup file of the same name if one exists.

- **force_write**
  Writes a new form definition file (form_name.form) when you invoke with -no_edit. By default, -no_edit produces the object module and specified include files only. Use -force_write with -no_edit to generate a .backup form file or to rename your form without reediting it.

- **basic**
  Creates OpenVOS BASIC versions of the field identifiers file and the field-values file. By default, the editor does not create OpenVOS BASIC versions of the files. When revising a form, the command uses the already-specified language. You can override the existing value with the Forms Editor [MENU]-S request.
- **-cobol** *(CYCLE)*  
  Creates OpenVOS COBOL versions of the field identifiers file and the field-values file. By default, the editor does not create OpenVOS COBOL versions of the files. When revising a form, the command uses the already-specified language. You can override the existing value with the Forms Editor [MENU]-S request.

- **-fortran** *(CYCLE)*  
  Creates OpenVOS FORTRAN versions of the field identifiers file and the field-values file. By default, the editor does not create OpenVOS FORTRAN versions of the files. When revising a form, the command uses the already-specified language. You can override the existing value with the Forms Editor [MENU]-S request.

- **-pascal** *(CYCLE)*  
  Creates OpenVOS Pascal versions of the field identifiers file and the field-values file. By default, the editor does not create OpenVOS Pascal versions of the files. When revising a form, the command uses the already-specified language. You can override the existing value with the Forms Editor [MENU]-S request.

- **-pl1** *(CYCLE)*  
  Creates OpenVOS PL/I versions of the field identifiers file and the field-values file. By default, the editor does not create OpenVOS PL/I versions of the files. When revising a form, the command uses the already-specified language. You can override the existing value with the Forms Editor [MENU]-S request.

- **-no_pl1_template** *(CYCLE)*  
  Specifies that OpenVOS PL/I include files are to be generated as based structures. (See the Explanation section of this command description for details.) When revising a form, the command uses the already-specified language. You can override this argument with the Forms Editor [MENU]-S request.

- **-c** *(CYCLE)*  
  Creates OpenVOS C versions of the field identifiers file and the field-values file. By default, the editor does not create OpenVOS C versions of the files. When revising a form, the command uses the already-specified language. You can override the existing value with the Forms Editor [MENU]-S request.

- **-processor** *processor_string* *(CYCLE)*  
  Specifies the processor for which object code is to be generated. The values of *processor_string* are as follows:

  - default
  - pentium4

  If you are creating a form that is to run on a module using an IA-32 processor, specify the pentium4 value. By default, *processor_string* is the processor type of the current module. The default value is default.
- **-mapping_rules** mapping_rules
  Specifies one of the following data-alignment rules for include files generated by `edit_form`.
  ```text
  • default
  • default/check
  • shortmap
  • shortmap/check
  • longmap
  • longmap/check
  ```

  The default value indicates the system-wide default. The default alignment method is site-settable. By default, the data alignment rules specified by `default` are used. (See the Explanation section of this command description for details.) When revising a form, the command uses the already-specified language. You can override this argument with the Forms Editor [MENU]-S request.

- **-no_sort_into_by_alignment**
  Specifies that the alignment of data items in the field-values file overrides all other sorting criteria. By default, this argument is `yes`. (See the Explanation section of this command description for details.) When revising a form, the command uses the already-specified language. You can override this argument with the Forms Editor [MENU]-S request.

- **-no_produce_symtab**
  Produces forms without a runtime symbol table. Because the forms runtime symbol table is small, use `-no_produce_symtab` only where there is a shortage of virtual memory.

**Explanation**

The `edit_form` command invokes the Forms Editor. After you issue the `edit_form` command, your process is at editor request level. At editor request level, you can make a number of requests. To issue the Quit request and return to command level, press the [MENU] key and then type the letter `q` or `Q`.

If you give the path name of a form definition file when you issue the `edit_form` command, the Forms Editor reads the file and displays the defined form. If you are migrating a Forms-based application from a release earlier than VOS Release 15.0.0, see the manual *Migrating VOS Applications from Continuum Systems* (R607) for more information.

The Forms Editor deletes trailing spaces from all values you enter into the editor’s request form and from all lines you enter into the form you are constructing. It also deletes all empty lines from the bottom of the form. Thus when you write out the form definition file and the other files described previously, the files reflect these deletions.

If you specify `-into`, `-prefix`, `-mapping_rules`, `-sort_into_by_alignment`, or any of the language arguments (`-basic`, `-cobol`, `-fortran`, `-pascal`, `-pl1`, `-pl1_template`, or `-c`) for a particular form, these arguments are saved in the form definition file. You do not have to respecify these arguments when using the Forms Editor requests or in future invocations of the Forms Editor on that form.
If you specify -pl1_template, all PL/I include files are generated as based structures. Field-values structures can then be declared using the PL/I like attribute. You should specify -pl1_template or the Forms Editor [MENU]-S request to prevent possible longmap or shortmap mismatches in the generated include file.

The -mapping_rules argument allows you to specify the data-alignment rules for include files generated by edit_form. The value default indicates the system-wide default. The default is site-settable. The value shortmap specifies that the shortmap alignment rules are to be used for the include files. The value longmap specifies that the longmap alignment rules are to be used for the include files. The values default/check, shortmap/check, and longmap/check are equivalent to default, shortmap, and longmap, respectively.

Note that -mapping_rules affects include files differently, based on the language specified. In OpenVOS C and OpenVOS Pascal, when you specify -mapping_rules, the include files will contain explicit longmap or shortmap keywords in field-values structures. In OpenVOS PL/I, these keywords are generated only if you specify the -pl1_template argument or the Forms Editor [MENU]-S request. In OpenVOS PL/I, if you do not specify -pl1_template or the Forms Editor [MENU]-S request, you must add the longmap and shortmap keywords manually. OpenVOS COBOL, BASIC, and FORTRAN do not allow the definitions of structure templates. For these three languages, the Forms Editor generates an include file for the body of the structure, which you include in your program within the structure definition itself. To do this in OpenVOS COBOL, for example, you would specify the following:

```
  01   structure1.
    copy 'struct_info.incl.cobol'.
```

The -sort_into_by_alignment argument determines whether the alignment of the data items in the field-values file overrides all other sorting criteria. The default is yes. This argument is used to avoid compiler warnings if you compile with the -mapping_rules default/check, shortmap/check, or longmap/check argument. This argument also minimizes the size of the field-values file because no padding will be required.

If you specify the -into argument when you invoke the Forms Editor, but do not specify any of the languages at that time, you can do so using the Forms Editor requests. If you do not specify any of the language arguments, the -into argument is ignored.

Access Requirements

You need read access to a form definition file to read it; you need write access to a form definition file, include file, or object module to write it.

Related Information

See the OpenVOS Forms Management System manuals for a complete description of the Forms Editor requests. Also see the description of the nls_edit_form command.
emacs

Purpose

This command invokes the Emacs text editor.

Display Form

```
emacs
```

```
file_names:  
-start_up_path:  current_start_up_path_name  
-num_windows:  1  
-backup:  no  
-keystrokes:  no  
-keystrokes_dir:  current_directory  
-flow_control:  no  
-nls:  no  
-dictionary:  current_dictionary_path_name  
-organization:  sequential  
-record_size:  
-character_set:  none  
-shift_mode:  none  
-pathname_style:  vos  
-compatibility:  vos  
```

Command Line Form

```
emacs [file_names ...]  

[-start_up_path start_up_path_name]  
[-num_windows number]  
[-backup]  
[-keystrokes]  
[-keystrokes_dir keystrokes_path_name]  
[-flow_control]  
[-nls]  
[-dictionary dictionary_path_name]  
[-organization organization]  
[-record_size record_size]  
[-character_set character_set]  
[-shift_mode shift_mode]  
[-pathname_style style_name]  
[-compatibility compatibility]  
```
Arguments

- **file_names ...**
  One or more names or star names of files to be edited. Copies of the files are read into Emacs buffers in the order in which the file names are specified. By default, Emacs creates an empty buffer for editing. You name the file when you write the file and specify an output path name.

- **-start_up_path start_up_path_name**
  Specifies an Emacs startup file. The name of the startup file must have the suffix .emacs, though you can omit this suffix when you specify the path name. This file tailors the editing environment to your needs. You can use this file to set modes, alter default keystroke sequences for Emacs requests, and assign keystroke sequences to requests that do not currently have them, including your own macros. By default, Emacs looks for a start_up.emacs file in your home directory. If there is no start_up.emacs file in your home directory, Emacs uses built-in default settings for all modes.

- **-num_windows number**
  Specifies the number of horizontal windows you want to open on the terminal screen. The Emacs editor assigns buffers to windows in the order in which the file names were specified in the file_names list. If the value of number is greater than the number of files specified, the extra windows are empty. By default, one window displays the first buffer named in the file_names list. On a standard 24-line screen, the maximum number of windows is five.

- **-backup**
  Creates a backup copy of any file that the Emacs editor reads into any buffer and then writes to the file. The backup copy is created when you write the contents of the buffer to the file. The path name of the backup copy is the same as the name of the original file, with the suffix .backup added. By default, Emacs does not create a backup copy of the file read into a buffer.

- **-keystrokes**
  Creates a keystrokes file. If you specify -keystrokes, Emacs creates or overwrites, in your home directory, a file named _emacs.terminal_name, which in some cases has a numeric suffix following terminal_name. This file contains the sequence of keystrokes you type during an Emacs session. (Keystrokes are control sequences that you issue to give requests and all characters that you insert into a buffer.) You can apply a keystrokes file to a backup file to recover from editing mistakes, or to reinstate changes you made to a file that you were unable to save. By default, Emacs does not create a keystrokes file.

**Note:** When you invoke Emacs with the -keystrokes argument, the keystrokes file for your current Emacs session overwrites the keystrokes file for your previous Emacs session except when the keystrokes file names do not match because of terminal name or suffix differences. Such differences occur when you are in a subprocess, are logged in on a different physical terminal, or have begun a new OpenVOS login session on a system that uses terminal servers or window terminal software. See the VOS Emacs User's Guide (R093) for detailed
information on the situations in which Emacs creates multiple keystrokes files and for information on determining which keystrokes file to apply.

► **-keystrokes_dir keystrokes_path_name**

  Specifies the name of the directory in which the keystrokes files will be stored when you give the -keystrokes argument. By default, Emacs stores keystrokes files in the user’s home directory. You cannot give this argument without also giving the -keystrokes argument.

  Specifying the -keystrokes_dir argument is especially useful in situations in which Emacs creates multiple keystrokes files, since it enables you to specify the name of the directory in which to store the keystrokes file for a particular Emacs session.

► **-flow_control**

  Enables Emacs to recognize the default output flow characters defined for your terminal. For most terminals, these are `CTRL-S` and `CTRL-Q`. By default, Emacs ignores output flow control, making the flow-control characters available for use.

► **-nls**

  Enables an Emacs mode that handles special character sets, such as Chinese, kanji, katakana, or hangul, which use multibyte sequences to represent a single character. When National Language Support (NLS) mode is enabled, Emacs treats these multibyte sequences as an indivisible unit. When NLS mode is disabled, Emacs displays shifted characters in hexadecimal format, and editing requests operate on individual bytes. By default, Emacs checks your process language to determine the appropriate setting. NLS mode is enabled if the character set of the language is `simplified_chinese`, `chinese1`, `chinese2`, `kanji`, `katakana`, `hangul`, or `user_dbcs`. For any other process language, NLS mode is disabled.

► **-dictionary dictionary_path_name**

  Specifies the user dictionary to be used with the check_spelling request. By default, if you give the check_spelling request, Emacs looks for a dictionary with the path name `(home_dir)>user_dictionary`.

► **-organization organization**

  Specifies one of the following types of file organization for the new file.

  • sequential
  • stream
  • relative
  • fixed

  By default, the command creates a sequential file.

► **-record_size record_size**

  Specifies the maximum record size, in bytes, when creating a file. The record_size is a value from 1 to 32,767. The command ignores this value for sequential and stream files.
-character_set  character_set

Specifies one of the following default character sets to be assigned to the file.

- none
- ascii
- latin_1
- latin_9
- kanji
- katakana
- hangul
- simplified_chinese
- chinese1
- chinese2
- user_dbcs

By default, a value of none is assigned to the file, indicating no default character set. Specify a character set only for a fixed, relative, sequential, or extended sequential file.

-shift_mode  shift_mode

Specifies the shift combinations allowed in the new file. Possible values for shift_mode are:

- none
- locking
- single
- all

By default, the command assigns the value none. This argument is ignored when -character_set is none. By default, both single- and locking-shift combinations (all) are allowed. If the shift mode is locking or all, file data is stored as compactly as possible at the expense of execution speed. For more information on shift modes, see the National Language Support User’s Guide (R212).

-pathname_style  style_name

Determines whether Emacs interprets path names as POSIX-style (slash-separated or greater-than-separated) path names or as OpenVOS-style (greater-than-separated) path names. Possible values for style_name are posix or vos. By default (vos), Emacs interprets all path names as OpenVOS-style path names.

This argument applies to all input path names, whether on the command line, given to prompts, or processed by the <ESC><TAB> completion action. It also applies to the path names that are arguments to the -dictionary, -start_up_path, and -keystrokes_dir arguments. This argument has no effect on output path names; Emacs always displays OpenVOS-style path names.

-compatibility  method_name

Determines whether Emacs commands and mode settings are initialized to their “traditional” values (that is, GNU Emacs values) or to OpenVOS-specific values. Possible values are vos or traditional. By default, Emacs initializes its commands and mode settings to OpenVOS-specific values.
**Explanation**

The `emacs` command invokes the Emacs text editor. Once you invoke the `emacs` command, your process is at the Emacs editor request level. At request level, you can issue any of the requests described in this manual. To return your process to command level, issue the `quit` request by pressing the `[ESC]` key and then the `[Q]` key.

When Emacs creates a new file (for example, with the `write_file` or `write_region` request), it uses the values specified by the `-organization`, `-record_size`, `-character_set`, and `-shift_mode` arguments to create the file with the specified attributes.

If the file already exists, Emacs retains the existing attributes of the file. If Emacs cannot create a file of the specified type, it displays an error and prompts you for a response with the following:

```
Invalid shift mode.
Can't make file. Make sequential file? (yes, no)
```

If you type `Yes`, a sequential file with a character set of `none` and a shift mode of `none` is created (the defaults). If you type `no`, the request is aborted.

When Emacs writes out a buffer that contains NLS characters, it overrides a request to create a stream file; instead, it creates a sequential file with a character set of `ascii` and shift mode of `all`.

A fixed file that has a default character set must specify a shift mode of `none`, indicating that no shift characters are allowed.

**Using Emacs on Flow-Controlled Terminals**

Some ASCII terminals generate output flow-control characters to regulate the data-flow rate from the host computer. These terminals send the ASCII characters `DC3 (\^S)` to halt the data flow and `DC1 (\^Q)` to resume the data flow.

If you are using Emacs on a flow-controlled terminal, the default output flow characters can cause unexpected results. If you specify the `-flow_control` argument, do not give editing requests that include your flow-control characters. If you do not specify the `-flow_control` argument and your terminal sends these characters, simply cancel any inappropriate prompts that Emacs issues.

The Emacs text editor accepts `\^S` for the `search` request and `\^Q` for the `quit` request. When you select the `emacs` command `-flow_control` argument, the Stratus hardware interprets these requests as instructions to halt or resume the sending of data. Therefore, to avoid this conflict, issue the `search` request by using the keystroke sequence `[ESC]-S` instead of `\^S`. Similarly, issue the `quit` request by pressing `[ESC]-Q` instead of `\^Q`.

**Access Requirements**

You need read access to a file in order to read it into an Emacs buffer. To write the contents of an Emacs buffer to a file, you need modify access to the directory and write access to the file (which can be specified in the default access list for the directory or the access list for the file).
**Related Information**

See the *VOS Emacs User’s Guide* (R093) for a complete description of Emacs requests. See also the descriptions of the `temacs` and `vemacs` commands.
encode_vos_file

Purpose

This command encapsulates and optionally encodes OpenVOS files for transport to non-OpenVOS systems, possibly over a non-8-bit transport medium, such as the Simple Mail Transfer Protocol (SMTP).

Display Form

<table>
<thead>
<tr>
<th>Argument</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>source_file</td>
<td></td>
</tr>
<tr>
<td>destination</td>
<td>no</td>
</tr>
<tr>
<td>-encode</td>
<td>no</td>
</tr>
<tr>
<td>-base64</td>
<td>no</td>
</tr>
<tr>
<td>-file_is_text</td>
<td>no</td>
</tr>
<tr>
<td>-no_header</td>
<td>no</td>
</tr>
<tr>
<td>-overwrite</td>
<td>yes</td>
</tr>
<tr>
<td>-output_sequential</td>
<td>no</td>
</tr>
</tbody>
</table>

Command Line Form

```
encode_vos_file source_file
[destination]
[-encode]
[-base64]
[-file_is_text]
[-no_header]
[-overwrite]
[-output_sequential]
```

Arguments

- **source_file**  
  The path name of a file to encapsulate/encode. You can specify only one file at a time (star names are not allowed).

- **destination**  
  The path name of the destination. If you do not specify a suffix, the command adds a suffix of .evf to the file. If you do not specify a value for `destination`, the output file will have the same name as `source_file`, but with a .evf suffix. If `destination` is the path name of a directory, the output file will be a file in that directory with the same name as `source_file`, but with a .evf suffix.
encode_vos_file

-encode
Encodes the output file in the UNIX uuencode format. This argument is useful when you must transfer a binary file using a method that does not allow transmission of binary data. By default (no), the command does not encode the output file in the uuencode format.

The output format is a stream file composed of records, each of which contains 61 printable characters. Some Internet mail servers may still corrupt uuencoded files, because the character set, although printable, includes punctuation characters that mail servers rarely filter. Because of this, the base64 encoding scheme is recommended for use with Internet mail applications. Using the uuencode format increases the size of the output file by 38 percent.

-base64
If you specify -base64, the output file is encoded in the MIME base64 format. MIME (Multipurpose Internet Mail Extensions) is a standard developed for Internet mail transmission. Base64 is the MIME standard for transmission of binary data in mail messages.

The output format is a stream file composed of records, each of which contains 72 alphanumeric characters (plus the slash (/), plus-sign (+), and equals-sign (=) characters). (Note that the last line may contain fewer than 72 characters.) These characters are not corrupted by Internet mail servers, even those that perform ASCII-to-EBCDIC conversion. Standard MIME application/octet-stream file attachment headers are used, but encode_vos_file does not produce a full email message (for example, one with subject lines). Base64 encoding increases the size of the output file by 33 percent.

-file_is_text
Specifies that input files are handled like stream files, and does not write encapsulation information. Specifying this argument causes all OpenVOS format information to be lost, but the output file retains the OpenVOS header information.

-no_header
When specified with the -file_is_text argument, converts simple OpenVOS sequential text files to stream format for transfer to non-OpenVOS systems.

-no_overwrite
Specifies that the command should not overwrite existing files that have the same names as those being encapsulated/encoded. By default (the value yes), the command silently overwrites existing files that have the same names as those being encapsulated/encoded.

-output_sequential
Allows you to transmit, via remote_request, a file that was compressed by the gzip utility. Otherwise, remote_request (that is, put_file and get_file) cannot transmit stream files over the RSN when either side is running a release earlier than VOS Release 12.

If you also specify the -encode or -base64 argument, the command performs encoding, but the output file’s organization is sequential instead of stream. The output
The format is the same as the encoded stream file: a series of records, each of which contains 61 (with -encode) or 72 (with -base64) printable characters. The resulting file can be transmitted by remote_request, rsn_transfer, email, or any method that accepts an ASCII sequential file.

If you specify -output_sequential without specifying -encode or -base64, the command converts the file into sequential format without encoding the data. The input data stream is segmented into records that are laid into a sequential file without translation. The output format is a sequential file composed of 61-byte records containing binary data. The resulting file can be transmitted by remote_request, rsn_transfer (use the -binary argument), or any method that accepts a sequential file containing binary data. This processing increases the file size by approximately 8% percent. Compare this method of processing to the following:

- -encode (stream), which increases file size by 38 percent
- -encode (sequential), which increases file size by 46 percent
- -base64 (stream), which increases file size by 33 percent
- -base64 (sequential), which increases file size by 41 percent

**Explanation**

The encode_vos_file command encapsulates (if necessary) and optionally encodes a non-extent OpenVOS file with the sequential, stream (but not 64-bit stream), relative, or fixed file format so that it can be transported through non-OpenVOS systems without loss of the OpenVOS file-format information. If you specify an extent file, it becomes a non-extent file after it is decoded and therefore might not be able to grow large enough to hold the contents of the original extent file. In addition, any attributes associated with the original file (for example, open options, implicit locking, and so on) are not present on the decoded file.

**Related Information**

See the description of the decode_vos_file command.
Purpose
This command converts cleartext data into ciphertext data.

Display Form

```
input_file: encrypt  output_file:
-password:     -delete:  no
-s suppress_password: no
```

Command Line Form

```
encrypt input_file
          [output_file]
       -password string
          [-delete]
          [-suppress_password]
```

Arguments

- **input_file**  
  The name of the file to be encrypted. This file can have any file organization (sequential, relative, fixed, or stream), and it can have the pipe attribute. However, it cannot have any indexes.

- **output_file**  
  The name of the file to contain the encrypted data. This file has the same file organization as input_file. If input_file has the pipe attribute, output_file has it, too. If you specify neither output_file nor the -delete argument, the command asks if you want to replace input_file with an output_file of the same name. Type y or Y at the prompt if you want to replace it. If you type n or N at the prompt, input_file is not encrypted.

- **-password string**  
  A string that forms the encryption key. The string must have a length of one to eight characters and can contain any character.

  **Note:** Abbreviations are not expanded.
- **-delete**
  Suppresses the prompt that occurs if you specify the same file for both `input_file` and `output_file`, or if `output_file` already exists. By default (the value no), the command displays this prompt.

- **-suppress_password**
  Prevents the command from saving an encrypted form of the password into `output_file`. By default (the value no), the command saves this information into `output_file`.

**Explanation**

The `encrypt` command converts cleartext data into ciphertext data. The command reads the input file, encrypts the contents of the input file using the Data Encryption Algorithm (DEA) in cipher feedback 8 mode, and writes the encrypted data into the output file. By default, the output file replaces the input file.

By default, the `encrypt` command writes a value computed from the encryption key into the first record of the output file so that the `decrypt` command can check it. You can override this behavior by using the `-suppress_password` argument when the file is encrypted. In this case, the command does not write any information about the encryption key into the output file. If the password used to decrypt the file does not match the password used to encrypt the file, the decrypted file does not match the original cleartext input file.

You can prevent the `encrypt` command from writing the encrypted form of the password into the output file by specifying the `-suppress_password` argument. For a fixed or stream file, this argument suppresses writing the first record that contains the password. For a relative or sequential file, this argument writes a zero-length first record.

When you encrypt a fixed or stream file using the `-suppress_password` argument, you must also use this argument when decrypting the file.

When you encrypt a sequential or relative file using the `-suppress_password` argument, the initial zero-length record informs the `decrypt` command that no password is available. Therefore, you are not required to specify the `-suppress_password` argument when you are decrypting sequential or relative files.

If you use the `-suppress_password` argument to encrypt a file, and then you accidentally use the incorrect password to decrypt the file, and finally, you overwrite the input file with the (incorrectly) decrypted output, you must re-encrypt the file with the incorrect password, and then decrypt it with the correct password.

The encryption and decryption algorithm is entirely dependent on the user-specified password; no other information is used during the encryption or decryption process.

**Access Requirements**

You need read permission on the input file, and you need modify access and default write permission on the directory containing the output file.
Related Information

See the `decrypt` command. Also, for more information about DEA support, see the following files:

- `>system>doc>dea.doc`
- `>system>doc>tdea.doc`
enforce_region_locks

Purpose

This command turns mandatory region locking on or off for one or more stream files.

Display Form

```
----------------------------------- enforce_region_locks -----------------------------------
file_name: 
state: on
```

Command Line Form

```
enforce_region_locks file_name [state]
```

Arguments

- **file_name** Required
  The name or star name of a stream file or files for which you want to enforce region locks. The command sets a mandatory locking switch for all stream files with matching names.

- **state**
  The state of the region locking switch for the specified files, which can be either on or off. By default, `enforce_region_locks` sets a mandatory locking switch to on for the specified files.

Explanation

The `enforce_region_locks` command turns mandatory locking on or off for all stream files whose names match `file_name`.

When the mandatory locking switch for a stream file is set to on, the operating system enforces all region locks that are currently in effect for that file. You cannot read a region that has been write-locked by another process, and you cannot write a region that has been read-locked by another process.

You do not have to lock a region of a stream file that you want to read or write if a mandatory locking switch for the file is set to on. However, if you do lock the region, another process cannot access that region to perform an I/O operation that conflicts with the lock state of the region.
enforce_region_locks

When the mandatory locking switch for a stream file is set to off, advisory locking is in effect. Each process making a call to a region must check for appropriate lock states before performing an I/O operation.

You can invoke this command only for use on stream files.

Examples
The following command disables mandatory region locking for the stream file make_report.out.

    enforce_region_locks make_report.out off

Related Information
For a complete list of file attributes, see the description of the display_file_status command. For information on region locking, see the description of the s$lock_region subroutine in the OpenVOS Subroutines manuals.
Purpose

This command compiles an OpenVOS FORTRAN source module.

Display Form

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>source_file_name</td>
<td></td>
</tr>
<tr>
<td>-define</td>
<td></td>
</tr>
<tr>
<td>-processor</td>
<td>default</td>
</tr>
<tr>
<td>-mapping_rules</td>
<td>default</td>
</tr>
<tr>
<td>-list</td>
<td>no</td>
</tr>
<tr>
<td>-table</td>
<td>no</td>
</tr>
<tr>
<td>-optimize</td>
<td>yes</td>
</tr>
<tr>
<td>-mapcase</td>
<td>no</td>
</tr>
<tr>
<td>-cpu_profile</td>
<td>no</td>
</tr>
<tr>
<td>-fixedoverflow</td>
<td>no</td>
</tr>
<tr>
<td>-full</td>
<td>no</td>
</tr>
<tr>
<td>-short_logical</td>
<td>no</td>
</tr>
<tr>
<td>-recursive</td>
<td>no</td>
</tr>
<tr>
<td>-check_uninitialized</td>
<td>no</td>
</tr>
</tbody>
</table>
Command Line Form

```
fortran source_file_name
   [-define variable_name... ]
   [-processor processor_string]
   [-mapping_rules mapping_string]
   [-list]
   [-xref]
   [-table]
   [-production_table]
   [-no_optimize]
   [-check]
   [-mapcase]
   [-profile]
   [-cpu_profile]
   [-statistics]
   [-fixedoverflow]
   [-silent]
   [-full]
   [-fortran66]
   [-short_logical]
   [-short_integer]
   [-recursive]
   [-optimization_level number]
   [-check_uninitialized]
```

Arguments

- `source_file_name` Required
  The path name of an OpenVOS FORTRAN source module.

- `-define variable_name`
  Defines variables to be used by the preprocessor. These variables are used during the preprocessor phase of compilation. Preprocessor variables can contain letters, digits, or the underline character (_), in any position. (See the Explanation section of this command description or the description of the `preprocess_file` command for details.)

- `-processor processor_string` [CYCLE]
  Specifies the processor on which the program module (.pm) is to run. The display form for the `-processor` argument restricts the values that you can choose to values for the processor family of the current module.
If the current module uses a processor from the IA-32 family, or if you specify, on the command line, the `-processor` argument with the `pentium4` value, the allowed `processor_string` values are as follows:

- `default`
- `pentium4`

The `default` value indicates the system-wide default. Unless your system administrator has reset this value, `default` is `pentium4` for modules using IA-32 processors. To determine the `default` value, issue the `display_error m$default_processor` command. By default, the compiler produces code intended for the processor specified by `default`.

- `-mapping_rules mapping_string`  
  Specifies one of the following data alignment rules for a given compilation.

- `default`
- `default/check`
- `shortmap`
- `shortmap/check`
- `longmap`
- `longmap/check`

The `default` value indicates the system-wide default. The default alignment method is site-settable. To determine the `default` value, issue the `display_error m$default_mapping` command. By default, the compiler uses the data alignment rules specified by `default`. (See the Explanation section of this command description for details.)

- `-list`  
  Creates a compilation listing. A compilation listing shows all source statements from the source module and include files, as well as a summary of all data definitions and the path names of include files used. You need not specify `-list` if you specify `-full` or `-xref`, since those arguments create a compilation listing in addition to other listings. By default, the compiler does not generate a compilation listing.

- `-xref`  
  Creates a compilation listing and an alphabetized cross-reference listing of all data actually referenced in the program. By default, the compiler does not generate a cross-reference listing.

- `-table`  
  Creates a symbol table in the object module, for use by the debugger. The compiler also performs some related operations. (See the Explanation section of this command description for details.) In addition, `-table` suppresses interstatement code optimization, which results in code that is slower than normal. Specifying `-table` sets the maximum optimization level to 1, unless you explicitly set the level to 0. By default, the compiler does not create a symbol table, suppress interstatement code optimization, or perform any related operations.

  **Note:** A symbol table greatly increases the size of an object module.
If you specify both `-table` and `-production_table`, the compiler produces only a production table and sets the maximum optimization level to 3, unless you explicitly specify some other value.

* `-production_table`  
  Creates a symbol table in the object module, for use by the debugger in a production environment. Only variables actually referenced in the program are placed in the symbol table. The compiler also performs some related operations. (See the Explanation section of this command description for details.) Unlike `-table`, `-production_table` does not suppress interstatement code optimization. As a result, the `set` and `continue` requests of the `debug` command can lead to unpredictable results. Also, the contents of variables in registers cannot be accurately displayed with the `display` request of the `debug` command. In addition, if the optimization level is greater than 2, the contents of any variables may not be accurately displayed with the `display` request of the `debug` command. Specifying `-production_table` sets the maximum optimization level to 3, unless you explicitly specify some other value. By default, the compiler does not create a symbol table, suppress interstatement code optimization, or perform any related operations.

* **Note:** A symbol table greatly increases the size of an object module.

If you specify both `-production_table` and `-table`, the compiler produces only a production table and sets the maximum optimization level to 3, unless you explicitly specify some other value.

* `-no_optimize`  
  Generates the object code without optimizing it. Optimization produces more compact object code by removing unnecessary or redundant computations. Specifying `-no_optimize` sets the optimization level to 0. This overrides any other specification of the optimization level. By default, the compiler optimizes the object code.

* `-check`  
  Checks for out-of-bounds array subscripts and out-of-range substring references when the object module runs. The compiler checks while compiling and inserts code to check further when the program is run. By default, the compiler does not check or insert checking code.

* `-mapcase`  
  Interprets all uppercase letters, except those in character-string and Hollerith constants, as lowercase letters. If you specify `-mapcase`, and the source module contains an external variable name or entry name, you may not be able to bind the resulting object module. (See the Explanation section of this command description for details.) By default, the compiler distinguishes between uppercase and lowercase letters.

* `-profile`  
  Inserts code in the compiled program that counts the number of times each source statement is executed when the program runs. (See the description of the `profile` and `add_profile` commands.) By default, the compiler does not insert the counting code. You cannot specify both `-profile` and `-cpu_profile` in the same command.
-cpu_profile

Inserts code in the compiled program that counts the number of times each source statement executes, the amount of CPU time (in milliseconds) spent executing each statement, and the number of page faults taken executing each statement when the program runs. (See the description of the profile and add_profile commands.) By default, the compiler does not insert the counting code. You cannot specify both -cpu_profile and -profile in the same command.

**Note:** The code inserted by this argument uses much more CPU time, but provides more useful information, than the code inserted by -profile.

-statistics

Displays the following statistics about the compilation as it proceeds.

- version number of the compiler
- elapsed CPU time
- elapsed real time
- number of page faults taken
- amount of storage used

You can specify -statistics to see the progress of the compilation and to determine the phase in which an error occurs. If the compiler produces a listing, it puts the statistics in the listing. By default, the compiler does not display compilation statistics.

-fixedoverflow

Generates code to check for fixed-point arithmetic overflow when the program is run and to signal the fixedoverflow condition when it occurs. By default, the compiler does not detect most fixedoverflow exceptions. In this case, if a fixedoverflow condition occurs, the high-order bits that caused the overflow are lost, and the remaining bits appear as they normally would in the result. However, note that code generated for the exponentiation operator (**)) may check for fixedoverflow exceptions even if you do not specify -fixedoverflow.

-silent

Suppresses the warning messages of severity-1 or severity-0 errors on your terminal during compilation. The compiler, nevertheless, puts the messages in an error file and in any listing it produces. By default, the compiler writes all error messages on your terminal.

-full

Creates from the compiled object code an assembly listing, with added comments, in addition to a compilation listing (see the -list argument). The compiler uses a disassembler to create the listing. By default, the compiler does not create an assembly language listing.

-fortran66

Requires that each do-loop be executed at least once. By default, the compiler does not require that do-loops be executed.
-short_logical
  Determines the length of logical data items whose length is not explicitly declared. Length is 2 bytes with -short_logical and 4 bytes with -no_short_logical. By default, the length is 4 bytes.

-short_integer
  Determines the length of integer data items whose length is not explicitly declared. Length is 2 bytes with -short_integer and 4 bytes with -no_short_integer. By default, the length is 4 bytes.

-recursive
  Allocates dynamic storage for some variables, thus allowing recursion for subroutines and limited recursion for external functions. (An external function cannot call itself, but it can call another procedure that calls an external function.) By default, the operating system allocates all variables in static storage and disallows recursion.

-optimization_level number
  Specifies the degree of optimization. The possible values are 0, 1, 2, 3, and 4. (See the Explanation section of this command description for details.)

-check_uninitialized
  Issues diagnostics for all references to uninitialized variables if you also specify an optimization level of 3 or 4. If you specify this argument and an optimization level that is less than 3, the compiler issues an error. This argument is useful when verifying new code or checking for possible bugs, but it can return misleading diagnostics, as in the case of variables that are initialized within a conditional statement. The categories of uninitialized variables diagnosed by the compiler vary, depending on whether you choose both -check_uninitialized and an optimization level of at least 3, or choose only an optimization level of at least 3.

Explanation

The fortran command compiles an OpenVOS FORTRAN source module into an object module.

The name of the source module must have the suffix .fortran; you can either supply or omit the suffix when you give source_file_name. The compiler generates an object module, puts it in your current directory, and names it. The name of the object module is the name of the source file with the suffix changed from .fortran to .obj.

When you are compiling programs for an ftServer module at all optimization levels, the module on which you are compiling must have at least 30,000 pages of paging partition available to avoid running out of virtual memory. In addition, the module on which you are compiling should have 64MB of physical memory available to achieve optimal compiler performance.

Using the -define Argument

The -define argument defines variables to be used during the preprocessor phase of the compilation. For example, if you specify the following on the command line, the preprocessor
variables var_a and var_b will be initially defined during the preprocessing phase of the compilation:

```fortran
fortran prog1 -define var_a var_b
```

You use preprocessor variables with preprocessor statements to perform conditional compilation on a program. Conditional compilation enables you to switch on or off various statements in a program. This is useful, for example, if you want your program to compile different lines of source code on different processors. There are six preprocessor statements.

- `$define`
- `$undefine`
- `$if`
- `$else`
- `$elseif`
- `$endif`

Preprocessor statements must begin in the first column of the source program. Therefore, indentation of nested `$if` statements is not allowed.

A preprocessor statement must be contained on a single line. A line containing a preprocessor statement cannot contain comments or parts of the source language. (An exception is the `$endif` statement, which ignores any text following it on the line, thus allowing you to comment on the source code.)

For more information on the preprocessor, see the description of the `preprocess_file` command.

### Using the `-processor` Argument

The `-processor` argument allows you to specify the processor on which the program is to run. The `-processor` argument also allows you to perform cross-compilation on a source module if the FORTRAN cross compiler is available on your system. Cross-compilation occurs when a compiler running on one processor family translates a source module into object code for another processor family. The IA-32 cross compiler generates code to run on ftServer modules. Specify the value `pentium4` for the `-processor` argument to target an ftServer module.

Depending on the value specified in the `-processor` argument, the compiler automatically defines one preprocessor variable for the processor family and one or more preprocessor variables corresponding to the processor type(s), as shown in Table 2-21.

**Table 2-21. Predefined Preprocessor Variables**

<table>
<thead>
<tr>
<th>Processor Value</th>
<th>Preprocessor Variable</th>
</tr>
</thead>
<tbody>
<tr>
<td>default</td>
<td>Varies, depending on the default system processor</td>
</tr>
<tr>
<td>pentium4</td>
<td><strong>PENTIUM4</strong>, <strong>IA32</strong>, and __i386</td>
</tr>
</tbody>
</table>

If you specify `-processor pentium4` on the command line, the preprocessor variables `__PENTIUM4__`, `__IA32__`, and `__i386` are defined.
If the value specified in the `-processor` argument indicates the IA-32 processor, the maximum number of bytes available for each function’s initial stack frame is 2,147,483,584 bytes.

The amount of automatic storage you can actually declare is somewhat less than these limits because temporary variables generated by the compiler also count towards the limit. Note that although the OpenVOS FORTRAN compiler supports extremely large values (such as 2,147,483,646), the operating system does not support them.

**Using the `-mapping_rules` Argument**

The `-mapping_rules` argument allows you to specify the data alignment rules for a given compilation. The value `default` indicates the system-wide default. The default is site-settable. The value `shortmap` specifies that the shortmap alignment rules are to be used for the source module. The value `longmap` specifies that the longmap alignment rules are to be used for the source module. The values `default/check`, `shortmap/check`, and `longmap/check` are equivalent to `default`, `shortmap`, and `longmap`, respectively, except that they also diagnose alignment padding within structures. For example, if you specify `default/check`, the compiler displays a severity-0 message stating how many bytes of padding exist between members of a structure. The `%options mapping directives override `-mapping_rules` values, but alignment padding within structures is still diagnosed if you specify one of the checking values.

**Using the `-full`, `-list`, or `-xref` Argument**

If you specify the `-list`, `-full`, or `-xref` argument, the compiler creates a compilation listing file and puts it in your current directory. The name of the compilation listing is `source_file_name.list`. The `-full` argument creates an assembly language listing in addition to a program listing. The `-xref` argument creates a list of cross-references in addition to a program listing.

**Using the `-table` or `-production_table` Argument**

If you specify the `-table` argument, the compiler creates a symbol table, and allocates storage and generates addresses for all external references, including any that are not used. Symbol-table capacity is 2,147,483,647 nodes. The compiler generates internal subroutines that calculate size, offset, and bound expressions that determine the characteristics of adjustable data. This allows the OpenVOS Symbolic Debugger to display and modify variable-length data according to its current length. In addition, the compiler suppresses interstatement code optimization.

If you specify the `-production_table` argument, the compiler performs all of the same operations that it performs for `-table`, except that it does not suppress interstatement code optimization, and only variables actually referenced in the program are placed in the symbol table (most unreferenced variables are from include files). Code produced with `-table` executes more slowly than code produced with `-production_table`. Code produced with `-production_table` can yield unpredictable results if you invoke the OpenVOS Symbolic Debugger set and continue requests.

**Using the `-mapcase` Argument**

When you compile a source module using the `-mapcase` argument, and the module contains an external variable name or entry name with one or more uppercase letters, you may not be able to bind the resulting object module. If the binder encounters a reference to the original
name (for example, in a binder control file), it will not recognize the original name and its lowercase version as the same name.

**Using the `-recursive` Argument**

If you select the `-recursive` argument, the compiler allocates some variables as dynamic variables. Variables in dynamic storage lose their values when the containing procedure exits, and new storage is allocated at each invocation of the procedure. Variables in static storage, by contrast, are allocated storage only once, at the first invocation of the external procedure, and maintain their values until the program terminates. The `-recursive` argument allocates in dynamic rather than static storage all variables in the program module except the following:

- those that appear in common, data, or save statements, or are equivalenced to variables in those statements
- those that are initialized in type-statements, or are equivalenced to variables initialized in type-statements

If you omit the `-recursive` argument, the operating system allocates all variables in static storage and disallows recursion. If a save statement that specifies no data elements appears in an external procedure, it forces all variables in the procedure to be static. This overrides the recursive argument and disallows recursion for that procedure.

Recursion for external procedures is an OpenVOS FORTRAN extension.

**Optimizations for ftServer Modules**

The `-optimization_level` argument allows you to optimize programs at different levels. When you are compiling a source module to run on ftServer modules, the levels of optimizations are 1, 2, 3, and 4. Specifying optimization level 3 or 4 causes the compiler to perform level 3 optimizations.

If you specify optimization level 0, the compiler performs the following optimizations.

- local register allocation
- elimination of unreachable code

If you specify optimization level 1, the compiler performs all level 0 optimizations plus the following other local optimizations.

- local pattern replacement
- short-circuit evaluation of Boolean expressions
- recognition of algebraic identities
- constant folding
- local combination of common subexpressions within a statement
- peephole optimizations within a single statement
- result incorporation
If you specify optimization level 2, the compiler performs all level 1 optimizations plus the following global optimizations.

- branch retargeting
- global combination of common subexpressions
- removal of invariant expressions from loops
- subsumption
- peephole optimizations across statement boundaries
- global register allocation

If you specify optimization level 3, the compiler performs all level 2 optimizations plus the following global optimizations.

- constant propagation
- removal of invariant assignments from loops
- strength reduction
- linear test replacement
- elimination of dead assignments
- elimination of useless loops
- check for uninitialized variables
- elimination of dead code and dead stores
- inline expansion
- instruction scheduling

Using the -no_optimize, -table, or -optimization_level Argument

The level of optimization is determined by the arguments -no_optimize, -table, and -optimization_level. Specifying -no_optimize sets the optimization level to 0. Specifying -table sets the level to 1, unless you explicitly set the level to 0. The -optimization_level argument sets the level to any of the permitted levels: 0, 1, 2, or 3. The compiler sets the actual level to the lowest level set by any of the three arguments. By default, the level is 3.

Note: If you compile a program with either the -profile or -cpu_profile argument, you must specify an optimization level lower than 3. Otherwise, -profile or -cpu_profile might not return accurate information, since high optimization levels can cause code to be moved from one statement to another.

Using the -check_uninitialized Argument

The optimization level for a source module also affects the functionality of the -check_uninitialized argument.

- If you select the -check_uninitialized argument and an optimization level of at least 3, the compiler diagnoses all instances of uninitialized variables within the source module. In this case, the compiler diagnoses variables that are initialized as part of code executed conditionally.

- If you do not select the -check_uninitialized argument but do select an optimization level of at least 3, the compiler diagnoses instances of variables within the source module that it knows are uninitialized. In this case, the compiler does not diagnose variables that are initialized as part of code executed conditionally.
If you select an optimization level of less than 3, the compiler issues an error and does not diagnose uninitialized variables within the source module even if you select `-check_uninitialized`.

**Interpreting Compiler Diagnostics**

If the compiler discovers any errors in your source module, it displays an error message on your terminal. Severity-1 and severity-0 messages are not displayed on your terminal when you specify the `-silent` argument. The compiler also creates an error file named `source_file_name.error` in the current directory and writes the error messages to the file. The compiler also appends error messages to a compilation listing if it produces one. The system deletes any `.error` file if a subsequent compile to the same source file is successful (contains no errors).

The OpenVOS FORTRAN compiler diagnoses five types of errors.

- **SEVERITY 0**: Advice
- **SEVERITY 1**: Warning
- **SEVERITY 2**: Correctable error
- **SEVERITY 3**: Uncorrectable error: translation can continue
- **SEVERITY 4**: Uncorrectable error: translation cannot continue

The text of the error message explains the cause of the error.

A severity-0 error, although valid FORTRAN, indicates that improvement is possible, usually in the area of performance. The source module is syntactically correct, so the compiled object module can be bound and executed, but probably with less than optimum efficiency.

A severity-1 error, although valid FORTRAN, is probably a programming error. Since the source module is syntactically correct at the point of a severity-1 error, however, the compiler continues to compile the source. The compiled object module can be bound and executed, but the program probably will not perform as expected.

A severity-2 error is invalid FORTRAN, but the compiler can reinterpret the source in such a way that it can continue to compile the program. The compiler proceeds as if the faulty code were replaced with the most likely syntactically correct code. The compiled object module can be bound and executed, but it probably will not perform as expected.

A severity-3 error is invalid FORTRAN, and the compiler cannot reinterpret the source code in such a way that it can continue to compile the program into a usable object module. Nevertheless, the compiler continues to process the program to detect additional errors. However, the object module is not created.

A severity-4 error is invalid FORTRAN, and the compiler cannot reinterpret the source code in such a way that it can continue to process the program after the point of the severity-4 error. The object module is not created.

**Note:** If the compilation results in more than 100 errors, in any combination (excluding severity-0 errors), compilation terminates.

The compiler always overwrites an existing object module having the same name as the object module it produces.
Access Requirements
You need read access to the source module to compile it. You need modify access to the directory from which you are issuing the compile command, in which the .obj file will be created.

Related Information
See the VOS FORTRAN Language Manual (R013) for a complete description of the OpenVOS FORTRAN language.
**get_external_variable**

**Purpose**
This command returns the assigned value of an external static variable within a program module.

**Display Form**

```
----------------------------- get_external_variable -----------------------------
external_variable_name:    
-in:                        
-type:  char
```

**Command Line Form**

```
get_external_variable external_variable_name -in program_module_name
   [-type string]
```

**Arguments**

- **external_variable_name**  
  The name of the external variable whose value is to be returned. **Required**

- **-in program_module_name**  
  The path name of the program module containing the external variable. The program module can have either a fixed or stream file organization. **Required**

- **-type string**  
  Specifies the data type of the variable. The possible values are integer, char, char_varying, or bit1. The default is char. The length of the target is contained in the program module. Integers must be signed and either 16 or 32 bits. **CYCLE**

**Explanation**

The `get_external_variable` command returns the value of an external static variable within a program module. The variable must be defined as static external within the program. You must specify the variable name as it is defined in the program. You must also specify the program. The type must be the same as the type of the assigned value.

For unshared variables in tasking programs, the value from task one is returned.
get_external_variable

Examples

The following command displays the value of the integer variable record_size, from the program monthly_sales.

```
get_external_variable record_size -in monthly_sales -type integer
```

Related Information

To set the current value of an external static variable within a program module, use the set_external_variable command.
give_access

Purpose
This command gives users access to files, directories, and devices by adding entries in the access control lists of files, directories, and devices.

Display Form

```
<table>
<thead>
<tr>
<th>access:</th>
<th>read</th>
</tr>
</thead>
<tbody>
<tr>
<td>path_names:</td>
<td>current_user</td>
</tr>
<tr>
<td>-user:</td>
<td>current_user</td>
</tr>
</tbody>
</table>
```

Command Line Form

```
give_access access path_names ...
[-user user_names ...]
```

Arguments

- **access**
  - The access you give. The `give_access` command assigns this access type to users. The possible types of access for files are execute, read, write, and null. The possible types of access for directories are status, modify, and null. The possible types of access for devices are read, write, and null.

- **path_names**
  - One or more names or star names of files, directories, or device access lists. The path name for device access lists is `>system>acl>access_list_name`. The `give_access` command adds entries in the access control list of each matching file, directory, or device.

- **-user user_names**
  - Specifies one or more user names or star names. The `give_access` command adds entries in the access control lists for these user names. By default, the command uses your user name.

Explanation
The `give_access` command gives users access to files, directories, and devices by adding entries in the access control lists of files, directories, and devices. An access control list is a table of user-name access-type pairs. The command derives the pairs it adds from the `-user` and `access` arguments.
The path_names argument for device access lists contains the file name access_list_name. This file name matches the value for the access_list_name field for each device in a set in the devices.tin file. The access_list_name file contains the access control list (ACL) for a device or set of devices. The operating system creates the ACL from the devices.tin file. A user can then add user names to the ACL by issuing the give_access command. The system then automatically updates the ACL. If the access_list_name field for a device does not have a value, no ACL is created, and all users can access that device.

A user name is a name with two components, a person name and a group name, which are separated by a period. If either or both components are asterisks, the user name is called a user star name.

A user name can appear only once in an access control list. However, more than one user star name can represent an individual user, and each can have a different kind of access paired with it. The following rules define a user’s access.

- Each time the give_access command adds an entry to an access control list, it sorts the list. It puts the most specific entries ahead of more general entries. Specific user names appear before user star names. User star names with asterisks only in the second component are next. User star names with asterisks only in the first component follow those. The user star name *.* is last.

- When a user tries to use the file, directory, or device with this access control list, the operating system searches the list in order. The access associated with the first entry that matches the user’s user name is the user’s access.

There are four types of access to a file.

- Null access denies all access by a user to a file.

- Execute access allows a user to execute a program module or a command macro but not to read or write it.

- Read access allows a user to read or execute a file, if it is executable, but not to write it.

- Write access gives a user full access to the file.

There are three types of access to a directory.

- Null access denies all access by a user to a directory.

- Status access allows a user to display information about the directory, using commands such as list and display_file_status, but not to modify the directory by creating and deleting objects.

- Modify access gives a user full access to the contents of a directory, including the ability to create, delete, and rename objects.
There are three types of access to a device.

- **Null access** denies all access by a user to a device.
- **Read access** allows a user to use a device for reading, but not writing, operations.
- **Write access** allows a user to use a device for reading or writing operations.

**Note:** A device requires a user to have write access to log in to that particular device. If a user tries to log in to a device but does not have access to it, an error message is displayed.

**Access Requirements**

To add an entry in the access control list of a file, you need modify access to the directory containing the file.

To add an entry in the access control list of a directory, you need modify access to the directory containing the directory.

To add an entry in the access control list of a device, you need modify access to the directory containing the device access lists (\texttt{system\textless}acl\textgreater).

**Examples**

**Example 1.**

The following is a user name for a particular user.

\begin{verbatim}
Smith.Sales
\end{verbatim}

This name specifies the unique user \texttt{Smith.Sales}.

The following are user star names.

\begin{verbatim}
*.Sales
Smith. *
*
\end{verbatim}

The first name matches all user names of members of the group \texttt{Sales}; the second matches all user names of the user with person name \texttt{Smith}; the last matches all user names.

The \texttt{give\_access} command sorts the access control list entries of the four user names into the following order.

\begin{verbatim}
Smith.Sales
*.Sales
Smith. *
*
\end{verbatim}

The user \texttt{Smith.Sales} has the access in the access control list entry containing the user name \texttt{Smith.Sales}.
The user Smith.Accounting has the access in the list entry containing the user star name Smith.*. The user name does not match any particular user name in the access control list, but it does match the user star name Smith.*.

The user Jones.Sales has the access in the list entry containing the user star name *.Sales.

The user Jones.Accounting has the access of *.*.

**Example 2.**

Suppose your current directory contains the files `week.90-02-04`, `week.90-02-11`, `week.90-02-18`, `week.90-02-25`, and `week.90-03-03`, and you issue the following command.

```
give_access read week.90-02-* -user Smith.Sales Jones.*
```

The command adds the following entries to the access control list of the files `week.90-02-04`, `week.90-02-11`, `week.90-02-18`, and `week.90-02-25`.

Smith.Sales    read
Jones.*       read

**Example 3.**

Suppose the directory `weekly_old` in the current directory has the following access control list.

```
Smith.Sales    modify
Jones.*       modify
*.Accounting   status
```

The user Jones.Sales has modify access to `weekly_old`, because the access control list entry Jones.* is the first match for that user name. Now suppose that you issue the following command.

```
give_access null weekly_old -user Jones.Sales
```

The access control list now looks like this.

```
Jones.Sales    null
Smith.Sales    modify
Jones.*       modify
*.Accounting   status
```

The user Jones.Sales now has null access to `weekly_old`, since the access control list entry Jones.Sales is the first entry to match the user name.
Related Information

For more information about access, see the command descriptions of `display_access`, `display_access_list`, `display_default_access_list`, `give_default_access`, `propagate_access`, `remove_access`, and `remove_default_access`. For a detailed discussion of access, see *OpenVOS Commands User’s Guide* (R089) and *OpenVOS System Administration: Registration and Security* (R283).
**give_default_access**

**Purpose**

This command gives users default access to files in directories by adding entries to the default access control lists of the directories.

**Display Form**

```
----------------------------- give_default_access -----------------------------
access:  read
directory_names:  ...
-user:  current_user
```

**Command Line Form**

```
give_default_access access
directory_names ...
[-user user_names ...]
```

**Arguments**

- **access**
  
  The default access you give. The possible values are null, execute, read, and write.

- **directory_names**
  
  One or more names or star names of directories. The give_default_access command adds entries to the default access control list of each matching directory.

- **-user user_names**
  
  Specifies one or more user names or user star names. The give_default_access command adds entries in the default access control list for these user names. By default, the command gives default access to the current user.

**Explanation**

The give_default_access command gives users default access to files in directories by adding entries to the default access control lists of the directories.

The default access control list of a directory has the same form as the access control list of a directory or file. It is of user-name access-type pairs. However, unlike access control lists, which are associated with both directories and files, default access control lists are associated only with directories. Furthermore, the access types in a default access control list are access types to files.
A default access control list specifies the access of users to files in the directory when the users are not covered by the files’ access control lists. When checking a user’s access to a file, the operating system first searches the file’s access control list for a matching entry. If the user name does not match any entry in the access control list, the operating system searches the default access control list of the containing directory. If the user name does not match any entry in either list, the user has undefined access to the file. Undefined access is equivalent to null access.

When you create a directory, the operating system gives it the default access control list of its containing directory.

See the explanation of the `give_access` command for an explanation of user names.

**Access Requirements**

To add an entry in the default access control list of a directory, you need modify access to the directory.

**Examples**

The following command gives the user `Smith.Sales` null default access to the files in the directory `weekly_old`.

```
give_default_access null weekly_old -user Tom_Smith.Sales
```

The user `Smith.Sales` now has null access to each file in the current directory unless the user was explicitly given access to the file through the `give_access` command.

**Related Information**

See also the command descriptions of `display_access`, `display_default_access_list`, `give_default_access`, `propagate_access`, `remove_access`, and `remove_default_access`. For a complete description of access, see *OpenVOS Commands User’s Guide* (R089).
handle_sig_dfl

Purpose

The handle_sig_dfl command changes the default action of certain POSIX.1 signals.

Display Form

```
----------------------------- handle_sig_dfl -----------------------------
  -like: unix
  -sigint_like: unix
```

Command-Line Form

```
handle_sig_dfl

    [-like string]
    [-sigint_like string]
```

Arguments

- **-like string**
  
  Two values are permitted for `string`: `unix` and `vos`. The normal default signal-handling action is to process signals as UNIX does, by exiting the program or creating a debuggable `keep` file. When you specify the value `vos`, the default action for the signal that creates a `keep` file is changed to enter OpenVOS break level.

- **-sigint_like string**
  
  Two values are permitted for `string`: `unix` and `vos`. The normal default signal-handling action for the `SIGINT` signal is to process signals as UNIX does, by exiting the program. When you specify the value `vos`, the default action of `SIGINT` is changed to enter OpenVOS break level.

Explanation

The default behavior of some OpenVOS POSIX.1 signals is to exit the program or create a `keep` module. This command changes the default behavior for the signal whose default behavior is to create a `keep` file to suspend the program and enter OpenVOS break level. At OpenVOS break level, you can do one of the following:

- resume the program
- enter the debugger
- create a `keep` module
- signal the reenter condition
- exit the program.
If either switch is set to `vos`, the SIGINT signal enters OpenVOS break level. If the `-like` argument is set to `vos`, all signals that can be modified enter OpenVOS break level.

Some POSIX.1 signals have a default action of `ignore`, or `continue` or `stop`; this command does not affect the behavior of these signals. You cannot change the default action of unsupported signals or of the SIGSTOP and SIGKILL signals.

The new value persists for the life of the process or until it is changed again with this command.
Purpose

This command collects process information and program counter (PC) samples from an f1Server module for a specified length of time. The command output is stored in a file which you can analyze with the analyze_pc_samples command to evaluate the performance of one or more program modules.

Display Form

```
-------------------------------harvest_pc_samples----------------------------
```

duration: HH:MM:SS
-output_path: raw_pc_samples.current_date_and_time
-sampling_frequency: 16
-timing_jitter: 0
-program_validation_period:

Command Line Form

```
harvest_pc_samples [duration]

[-output_path path_name]
[-sampling_frequency seconds]
[-timing_jitter jitter]
[-program_validation_period period]
```

Arguments

- **duration**  
  The length of time during which the harvest_pc_samples command collects PC samples. Specify a value using the format HH:MM:SS, where HH is the number of hours, MM is the number of minutes, and SS is the number of seconds. For example, if you want to indicate a value of four and a half hours, specify the value as 04:30:00.

  The minimum value you can specify is 1 second, and the maximum value is 8,760 hours (one year).

- **-output_path path_name**  
  Specifies the path of the PC sample data file in which the command will place the collected data. The default path name is raw_pc_samples.current_date_and_time. The path name cannot be an extended name.
- **-sampling_frequency seconds**
  Specifies the frequency, in samples per second, at which PC samples are taken from each CPU. On ftServer modules, the minimum value you can specify is 3 samples per second, and the maximum is 256 samples per second. The default sampling rate is 16 samples per second.

  **Note:** Scheduler contention may not allow the harvest_pc_samples command to sample at as fast a rate as you have specified. For example, if you specify a value of 256, the sampling frequency may actually be only 250 samples per second. To determine the actual sampling frequency, check the value of the Actual Sampling Freq. field in the statistics summary and CPU usage subsection of the report generated by the analyze_pc_samples command. For more information, see the description of the analyze_pc_samples command.

- **-timing_jitter jitter**
  Specifies an integer value that allows you to slightly randomize the sampling frequency on the specified modules. You must specify an integer value of zero or greater. The default value is zero, which keeps the sampling frequency constant. A value of 1 causes the harvest_pc_samples command to collect data at intervals that cycle from 4 milliseconds (ms) less to 4 ms more than the specified sampling frequency. The timing jitter is always a multiple of 4 ms. A value of 2 causes the harvest_pc_samples command to collect data at intervals that cycle from 8 ms less to 8 ms more than the specified sampling frequency in 4 ms increments. For example, if the -sampling_frequency value is 32 times per second (once every 31 milliseconds), and the -timing_jitter value is 2, and sampling occurs at the following times:

  - 31 milliseconds - 8 ms
  - 62 milliseconds - 4 ms
  - 93 milliseconds + 0 ms
  - 124 milliseconds + 4 ms
  - 165 milliseconds + 8 ms
  - 196 milliseconds - 8 ms
  ...

  **Note:** The value of this argument does not change the value of the -sampling_frequency argument.

- **-program_validation_period period**
  Specifies the interval at which the harvest_pc_samples command performs program validation. Program validation occurs when the harvest_pc_samples command checks that the same programs are running under the same process IDs as they were when the command last checked. Specify a value using the format **HH:MM:SS**, where **HH** is the number of hours, **MM** is the number of minutes, and **SS** is the number of seconds. For example, if you want to indicate a value of 30 minutes, specify the value as **00:30:00**.

  If a program running under a process terminates and another program is invoked under the same process, the changed program will affect the results of the sample. The output of the analyze_pc_samples command lists processes that have changed programs during the validation period. If you specify a value that is greater than the value of duration, the command does not perform program validation. If you do not specify
a value for this argument, the command performs program validation at the end of the collection period specified by duration.

Explanation

The `harvest_pc_samples` command invokes the OpenVOS PC Sampler at a specified frequency, collects information about the processes running on the system, reads the PC data from an OpenVOS buffer, and writes it to the output file. You can use the `analyze_pc_samples` command to analyze this file and generate a report on the performance of one or more specified target program modules. A target program module is a program module about which you want to gather statistical performance data.

Notes:

1. To see function names and source code line numbers in the output report of the `analyze_pc_samples` command, bind the target program module with the `-retain_all` argument (OpenVOS is automatically bound with `-retain_all`) before running the `harvest_pc_samples` command.

2. Make sure the target program module runs during the same period of time that the `harvest_pc_samples` command runs. Note that you can specify the target program module with the `analyze_pc_samples` command, but not with the `harvest_pc_samples` command. The `harvest_pc_samples` command collects data for all modules currently running on the system.

3. You can run only one `harvest_pc_samples` process on a module at a time. If you try to run the `harvest_pc_samples` command and another process is already running it, the command displays the message: The PC Sampler has already been started. Another user may be executing the sampler. In this case, issue the `list_users` command to see who started the `harvest_pc_samples` process.

When to Use the `analyze_pc_samples` and `harvest_pc_samples` Commands

OpenVOS provides seven commands for measuring performance. The following table suggests when you might use each of these commands.

Table 2-22. Commands That Measure Performance

<table>
<thead>
<tr>
<th>Commands</th>
<th>Degree of Specificity</th>
<th>When to Use</th>
</tr>
</thead>
<tbody>
<tr>
<td>• harvest_meters</td>
<td>Most general. Collects, organizes, and displays performance information about CPU, memory, disk, file, server queue, communications device, and terminal use.</td>
<td>If you suspect system or application performance problems, use these commands to isolate bottlenecks.</td>
</tr>
</tbody>
</table>
Data Flow with the **analyze_pc_samples** and **harvest_pc_samples** Commands

Figure 2-5 shows the route that the program counter data takes as it is collected and analyzed. When invoked, the **harvest_pc_samples** command initializes the OpenVOS program counter buffer, where OpenVOS temporarily stores program counter information about program modules and routines. You specify the frequency at which OpenVOS stores PC samples in this buffer. The **harvest_pc_samples** command places sampled data from the program counter buffer into a raw data file whose name you specify. When you execute the **analyze_pc_samples** command with a specified program module (.pm), the command matches the contents of the raw data file to routines in a specified program module. The command then generates a report that shows the number of times each routine was executed in the specified program module.
Figure 2-5. The Program Counter Sampling System

Accuracy of the Data
If the `harvest_pc_samples` command collected information each time a program module executed in a module’s CPU(s), the data would be completely accurate. However, collecting data in this manner would greatly decrease system performance and quickly exhaust the available disk space. Therefore, the `harvest_pc_samples` command statistically samples a portion of the CPU activity. To make sure the sampled data accurately represents the actual activity in the CPU, you should take the following steps.

- Specify that the `harvest_pc_samples` command take samples as often as possible for as long as reasonable.

  **Note:** You can run the `harvest_pc_samples` command more than once with the same parameters if you think you have not collected enough data. The `analyze_pc_samples` command can concatenate several raw data files generated by the `harvest_pc_samples` command (effectively doubling or tripling the sampling duration) and create a more accurate report.

- Use timing jitter to increase the randomness of the sampled data.

- Validate the program names if you think the target program module will not run continuously under the same process.
The following paragraphs discuss how long you need to collect data, the frequency at which you should sample the data, when to use timing jitter, and when to validate program names.

**Length of Time to Collect Data**

Consider the following factors when specifying a value for the `duration` argument.

- Target program module execution time. The target program module must run during the entire period that the `harvest_pc_samples` command runs.

- The desired accuracy of the sample. The longer the duration, the greater the confidence you can place on the accuracy of the statistical results.

- The CPU usage of the target program module. If the target program module uses large quantities of CPU time, many data points will be generated in a short period, and you may specify a `duration` value of 10 or 15 minutes. If the target program module does not require much CPU time, specify a much longer period for the `duration` argument. If possible, try to maximize the CPU usage of the target program module while the `harvest_pc_samples` command is running. The higher the percentage of CPU usage by the target program module, the higher the percentage of useful samples. For example, if the target program module CPU usage is 10 percent, then 10 percent of the samples will be useful.

**Frequency at Which the Command Samples Data**

In general, when specifying a duration of less than one hour, use a high sampling frequency, such as 256 samples per second. To generate statistics in which you can have high confidence, you need to generate a large number of data points. Nevertheless, you should consider the available disk space before specifying a duration of greater than one hour with a high sampling frequency, since the raw data file can quickly grow to millions of bytes. For example, when you run the `harvest_pc_samples` command for one hour with a sampling frequency of 256 samples per second, each logical CPU will generate about 7.5 million bytes of data (or 1831 blocks) for the raw data file.

If the system on which `harvest_pc_samples` is running is very busy, the command may display the following error message: `sampling rate exceeds harvesting capacity`. This error means that the OpenVOS program counter buffer is filling before the command has read the buffer. If this occurs, use the `set_priority` command and rerun the `harvest_pc_samples` command at a higher priority.

**Timing Jitter to Use**

The `harvest_pc_samples` command uses the scheduler to generate interrupts at the specified frequency. This may skew the collection of scheduler and other program module samples. If you suspect that too many samples are concentrated in or absent from a set of program modules, use the `-timing_jitter` argument. Note that the maximum `-timing_jitter` value cannot exceed the maximum sampling frequency of once every 4 ms or the minimum sampling frequency of once per second. The following table illustrates the relationship between selected `-sampling_frequency` values and the maximum `-timing_jitter` value.
Validation of Program Names

If you have control over when the target program module starts, stops, and who can invoke it, then you do not need to specify a value for the \texttt{-program\_validation\_period} argument. If you cannot control the execution of the target program module, specify a value for the \texttt{-program\_validation\_period} argument in the range of 30 to 60 seconds.

**Examples**

The following examples show several different ways of specifying the arguments to the \texttt{harvest\_pc\_samples} command.

**Example 1.**

In this example, the \texttt{harvest\_pc\_samples} command runs for 10 minutes. The command samples data 256 times per second and performs program validation once just before it stops executing.

\begin{verbatim}
harvest_pc_samples 10:00
 -output_path pc_samples
 -sampling_frequency 256
\end{verbatim}

**Example 2.**

In this example, the \texttt{harvest\_pc\_samples} command also runs for 10 minutes and samples data 256 times per second. However, the command does not perform program validation.

\begin{verbatim}
harvest_pc_samples 10:00
 -output_path pc_samples
 -sampling_frequency 256
 -program\_validation\_period 10:01
\end{verbatim}

\begin{table}[h]
\centering
\begin{tabular}{|l|l|}
\hline
\textbf{Selected \texttt{-sampling\_frequency} Values} & \textbf{Maximum \texttt{-timing\_jitter} Values} \\
\hline
256 samples/sec (4ms) & 0 (0 milliseconds (ms) or no jitter) \\
\hline
125 samples/sec (8 ms) & 1 (+/- 4 ms) \\
\hline
84 samples/sec (12 ms) & 2 (+/- 8 ms) \\
\hline
50 samples/sec (20 ms) & 4 (+/- 16 ms) \\
\hline
16 samples/sec (63 ms) & 15 (+/- 90 ms) \\
\hline
4 samples/sec (125 ms) & 63 (+/- 256 ms) \\
\hline
2 samples/sec (500 ms) & 127 (+/- 496 ms) \\
\hline
1 sample/sec (1000 ms) & 0 (0 ms or no jitter) \\
\hline
\end{tabular}
\caption{Selected \texttt{-sampling\_frequency} Values and Maximum \texttt{-timing\_jitter} Values}
\end{table}
Example 3.
In this example, the \texttt{harvest\_pc\_samples} command runs for 1 hour. The command samples data 128 times per second and performs program validation every 30 seconds.

\begin{verbatim}
harvest\_pc\_samples 1:00:00
-\textbf{output\_path} pc\_samples
-\textbf{sampling\_frequency} 128
-\textbf{program\_validation\_period} 00:00:30
\end{verbatim}

Example 4.
In this example, the \texttt{harvest\_pc\_samples} command runs for 12 hours. The command samples data 32 times per second with a timing jitter of 7. The command also performs program validation every 30 seconds.

\begin{verbatim}
harvest\_pc\_samples 12:00:00
-\textbf{output\_path} pc\_samples
-\textbf{sampling\_frequency} 32
-\textbf{timing\_jitter} 7
-\textbf{program\_validation\_period} 00:00:30
\end{verbatim}

Related Information
For information on how to analyze the output created by the \texttt{harvest\_pc\_samples} command, see the description of the \texttt{analyze\_pc\_samples} command in this manual. For information on related performance measuring tools, see the descriptions of the \texttt{profile} and \texttt{add\_profile} commands in this manual. For more information on setting priorities, see the description of the \texttt{set\_priority} command in this manual.
Purpose

This command helps you find the names of commands and command functions.

Note: Support for updating or adding new topic information to the online help facility ended as of VOS Release 13. However, you can still use the help command for finding commands and command functions via the -match and -type arguments. For information about commands and command functions, use the StrataDOC online-documentation service instead of the help command. See the Preface for more information about StrataDOC.

Display Form

```
------------------------------------- help -------------------------------------
 -type: subsystem
 -match:                      
```

Command Line Form

```
help [-type help_type] [-match string]
```

Arguments

- `-type help_type`
  Specifies one of the following five types of item to display.

  - all
  - internal
  - external
  - function
  - subsystem

  If you specify any type but subsystem, help displays the names of all these items, or only the names of internal commands, external commands, or command functions, respectively. By default, help uses the value subsystem, and displays the top-level menu of the online help facility.

- `-match string`
  Displays only the names that contain the character string string. If you also use the -type argument with a value of internal, external, or function, help displays only names of the specified type that contain the string; with a value of all or
subsystem, the command displays all names that contain the string. By default, `help` displays all the names of items specified by `-type help_type`; if `help_type` is subsystem, the command displays the menu interface to the online help facility. Note that `-match` is case sensitive.

**Explanation**

The `help` command lists the names of commands and command functions.

To determine whether a command with a similar name exists, you can use the `-type` `command` argument and the `-match` argument with a common, yet descriptive, portion of the familiar name as the `string`.

Use `-type` when you know the item’s type but not its name, to list all the items of the same type.

If you want to see a display of the names of items related to a particular topic, use both the `-type` and `-match` arguments. Specify an appropriate value for the `-type` argument and your topic as the value of the `-match` argument. The `help` command displays all items of the specified type whose names contain the string `string`.

**Examples**

**Example 1.**
Enter the following command to display all of the command functions whose names contain the character string `current`.

```
help -type function -match current
```

This command displays the following output.

```
current_dir
current_module
```

**Example 2.**
You can use the `help` command when you have issued a command that does not exist, for example, `set_locking`. The following command helps you find the correct name.

```
help -type command -match lock
```
help
kill

Purpose
The kill command sends a signal to the process(es) specified by each pid operand.

Display Form

Command-Line Form

Arguments

- `-s signal_name pid ...

- `pid` argument is a string of numbers in one of three forms:
  - hexadecimal 0X1a3f3
  - hexadecimal 1A3F3x
  - decimal 1075

The `pid` argument is the process identifier of the process that you want to send a signal. Both POSIX PIDs and OpenVOS process identifiers are supported. More than one process can be affected by specifying the respective process identifiers as arguments to the command. When the first process identifier is negative, precede it by two hyphens (- -) to prevent the process identifier from being confused with an argument. Subsequent negative process identifiers can only have a single hyphen (-) in front of them.
kill

► -l [signal_number]
   Without the signal_number, -l displays all the signal names supported by kill. If you specify a signal_number (an unsigned decimal) value, -l displays only the signal name corresponding to the value of signal_number.

► -h --help
   The -h and --help arguments display a brief help message illustrating the formats of the command line.

Explanation
   The kill command sends signals to one or more processes. It only accepts signal names but lists all of the supported signal names. When a signal number is given, it gives a corresponding signal_name value. The kill command is a wrapper around the kill function. Thus, the kill command behaves similarly to the kill function from the POSIX.1 standard. The value of the pid operand is used as the pid argument. The sig argument is the value specified by the -s argument. Invalid arguments exit with a nonzero exit status.

Related Information
   For information on stopping a process, see the description of the stop_process command.
ldd

Purpose
This command lists all of the shared libraries on which the specified program module or shared library depends.

Display Form
None.

Command Line Form
```
ldd [ --version ] [ --help ] path_name [ path_name... ]
```

Arguments
- `--version`
  Displays the version number of the ldd command.
- `--help`
  Displays usage information for the ldd command.
- `path_name [ path_name... ]`
  One or more program module or shared library names. This value must be a POSIX-style path name.

Explanation
The ldd command lists all of the shared libraries on which `path_name` depends. The command displays POSIX-style path names in its output.

Examples
An example of ldd follows:
```
ldd salestest.pm
libvosposix.1.so => /lib/libvosposix.1.so
```

In the output, `libvosposix.1.so` is the shared library on which `salestest.pm` depends, and `/lib/libvosposix.1.so` is the location of the shared library.

Related Information
See the description of the list_dynamic_dependencies command.
line_edit

Purpose
This command calls the line editor.

Display Form

```
---------------------------------- line_edit -----------------------------------
| text_file_name: | -verbose: yes |
| -mapcase: no    |
| -numbers: no    |
| -keystrokes: yes|
| -backup: yes    |
| -backup_name:   |
| -keystroke_in:  |
```

Command Line Form

```
line_edit [text_file_name]
[-no_verbose]
[-mapcase]
[-numbers]
[-no_keystrokes]
[-no_backup]
[-backup_name backup_file_name]
[-keystroke_in keystroke_file_name]
```

Arguments

* text_file_name
  The path name of a text file to be edited. If the file exists, the editor reads it into the editor buffer for editing. The editor saves the path name so that you can write the edited text to the file text_file_name without retyping the path name. If the file does not exist, the editor buffer is initially empty. In this case, the editor asks you whether to create the file. Unless you tell the editor to create the file, the editor returns your process to command level. By default, the editor buffer is empty when you begin, and the editor does not save a path name.
- **-no_verbose**
  Suppresses verbose mode. In verbose mode, the editor prints every line that it modifies or selects with any of the following requests.

  - append
  - change
  - find
  - goto
  - locate
  - overlay

  Specifying -no_verbose has the same effect as making a no_verbose request while editing. By default, the editor begins in verbose mode.

- **-mapcase**
  Begins editing in mapcase mode. In mapcase mode, the editor disregards the case of alphabetical characters when comparing text strings with the change, find, locate, and overlay requests. This has the same effect as making a mapcase request. By default, the editor distinguishes uppercase and lowercase letters.

- **-numbers**
  Prints the number of a text line in the editor buffer whenever the editor prints the line on your terminal. The line number is printed in the leading four columns. This has the same effect as making a numbers request. By default, the editor does not print the line number.

- **-no_keystrokes**
  Edits a file without creating a keystrokes file. By default, the editor creates or overwrites a file in your home directory named `_line_edit.terminal_name`, where `terminal_name` is the device name of the terminal that you are using. The editor saves the sequence of editor requests you make and the characters you insert in the edited text. With a keystrokes file, you can usually recover from editing mistakes. If you are not editing from a terminal, the editor does not create a keystrokes file. For example, a keystrokes file is not created when the editor requests come from a command macro.

- **-no_backup**
  Edits without creating a backup copy of the file `text_file_name`. By default, the editor creates a backup copy of `text_file_name` the first time you write out the contents of the editor buffer to the file `text_file_name`. The path name of the backup copy is the same as the original file with the suffix `.backup`.

- **-backup_name backup_file_name**
  Gives the name `backup_file_name` to the backup copy of the edited file. By default, the path name of the backup file is the same as the file read in with the suffix `.backup`.

- **-keystroke_in keystroke_file_name**
  Edits a file using the keystrokes file `keystroke_file_name` created during a previous editing session. This allows you to recover from editing errors made during a long editing session in which most of the processing you did was acceptable. You must
have either a copy of the original file which has not been overwritten, or a backup copy of the original file.

**Explanation**

The `line_edit` command calls and sets up the line editor. After you issue the `line_edit` command, your process is at the line editor request level. At request level, you can make any of several editor requests to enter and modify text in a temporary work space and to write out the edited text from the work space to a file in the directory hierarchy. One request, the `quit` request, returns your process to command level. You can also return to command level by pressing `CTRL-BREAK`, which puts you at break level. At break level, typing `stop` returns you to command level, typing `continue` lets the editor resume what it was doing, and typing `re-enter` returns you to `line_edit` request level.

To use the keystrokes file, perform the following steps.

1. Copy or rename the keystrokes file. Do this before you invoke the line editor, since the keystrokes file will be overwritten by the editor.

2. Edit the copy of the keystrokes file to remove the section containing errors, and remove any `CTRL-BREAK` processing.

3. Invoke the `line_edit` command, supplying either the name of the file to be edited (if it has not been overwritten) or the backup file for `text_file_name` and the path name of the edited keystrokes file for `keystrokes_file_name`. The editor reads in the backup file to its buffer and performs the requests you have prepared in the backup file.

4. When the editor reaches the end of the backup file, it begins to accept requests from your terminal. You can now edit the buffer further interactively.

If you are unsure of the recovery procedures, use copies of the file or backup file and the keystrokes file to protect the file originals.

**Related Information**

For additional information about the `line_edit` command, see the *OpenVOS Commands User’s Guide* (R089).
link

Purpose
This command creates a link to a specified file, directory, or link.

Display Form

<table>
<thead>
<tr>
<th>Argument</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>target_name:</td>
<td></td>
</tr>
<tr>
<td>link_name:</td>
<td></td>
</tr>
<tr>
<td>-delete:</td>
<td>no</td>
</tr>
<tr>
<td>-brief:</td>
<td>no</td>
</tr>
</tbody>
</table>

Command Line Form

```
link target_name
    [link_name]
    [-delete]
    [-brief]
```

Arguments

- **target_name**  
  The path name or star name of target files, directories, or links. If you specify a target name that is not a star name, the target does not need to exist.

- **link_name**  
  Specifies the link path name or star name. If you specify a link name that is a star name, the `link` command creates and names links according to the rules for star-name pairs. By default, the command puts the link in the current directory and gives it the object name portion of `target_name`.

- **-delete**  
  Deletes a file or directory or unlinks a link if it has the same path name as the new link. By default, if the path name of the new link conflicts with an existing object, the operating system prompts you before deleting the object. If there is a conflict and you do not delete the existing object, the operating system does not create the link.

- **-brief**  
  Suppresses the display of each object and link that matches a star name, before the link is created. By default, the `link` command displays the names of the links and their targets.
Explanation

The `link` command creates links to one or more objects that you specify by `target_name`.

A link is an object contained in a directory that directs all references to itself onward to another object, that is, to a file, a directory, or another link. Like many other objects, a link has a path name that identifies it as a unique entity in the system directory hierarchy. The object to which the link refers is called its target. When you refer to a link, for example in a command, the operating system links you to the target by replacing the path name of the link with the path name of its target.

A link referenced by another link is called a nested link. The maximum number of nested links is 16. If you create a link that contains more than 16 nested links, the operating system displays the following warning message.

The link created is too long or circular.

Specify the name and location of the target with the argument `target_name`. If you specify a `link_name`, you can give a path name to the link; otherwise, the link appears in the current directory and has the same name as the object `target_name`.

If you specify `-delete`, you can delete a file or directory, or unlink a link in the current directory if its name conflicts with the name of the new link.

Because of the way in which links function, a link can be created to a device, instead of a file, directory, or link. The link name serves as a local abbreviation for the device. There is, however, little reason to create a link to a device; instead, put an abbreviation for the full path name of the device in your abbreviations file. This abbreviation serves as a global abbreviation for the device.

It is possible to create a circular link; a link that is its own eventual target. If you create a circular link, the operating system creates the link and sets the command status to 0, but returns the following warning message.

The link created is too long or circular.

Access Requirements

You must have modify access to the directory that contains the new link. Your access to the target of the link is unchanged when you set up a link to it.

Examples

Suppose your current directory is `%s1#d02>Sales>east>Smith`, and that Jones is a subdirectory of `>east`. The following command creates the link `jones_customers` in your current directory.

```
link <Jones>customers jones_customers
```

The target of the link `jones_customers` is the file, directory, or link with the following path name.

```
%s1#d02>Sales>east>Jones>customers
```
Related Information

For information about how to remove a link, see the description of the `unlink` command. See also the *Introduction to VOS* (R001) for discussions of abbreviations files and star-name pairs.
link_dirs

Purpose
This command provides each directory in a specified set with links to all objects in every other
directory in the set.

Display Form

```
-------------------------- link_dirs --------------------------
directories: ...
-brief: no
-files: yes
-dirs: yes
-links: yes
-depth: 1
-warnings: yes
```

Command Line Form

```
link_dirs directories...
    [-brief]  [-no_files]
    [-no_dirs]  [-no_links]
    [-depth number]  [-no_warnings]
```

Arguments

- **directories**  Required
  Two or more directories in which links are to be placed.

- **-brief**  [CYCLE]
  Suppresses messages about links created. By default, `link_dirs` displays messages
  about links as they are created.

- **-no_files**  [CYCLE]
  Ignores any files in the specified directories. By default, `link_dirs` creates links to
  files contained in the specified directories.

- **-no_dirs**  [CYCLE]
  Ignores any directories in the specified directories. By default, `link_dirs` creates
  links to subdirectories of the directories specified in the `directories` argument. This
  has the same effect as specifying `-depth 1`. 
link_dirs

- **-no_links**
  Ignores any links in the specified directories. By default, link_dirs creates links to links in the specified directories.

- **-depth number**
  Specifies the number of directory levels down the directory hierarchy in which links are to be created. By default, link_dirs creates links only in the specified directories; the depth is 1. Specifying -no_dirs also sets the depth to 1.

- **-no_warnings**
  Suppresses warnings in the event of conflicts between object names that occur in two or more directories. By default, link_dirs displays warnings about any conflicts.

**Explanation**

The link_dirs command places links in each of several directories to objects contained in only one or more of the other directories. As a result, the directories are made equivalent, since an object in one can be located (via a link) from any of the others. The path names specified in the directories argument can be on any level of the hierarchy. The number of directories you can specify is restricted only in that the path names must fit into a line no more than 300 characters long.

The principal use of the link_dirs command is to create links in top-level directories on multiple-logical-disk systems.

If the same object name occurs in two or more of the directories specified in the directories argument, the link_dirs command displays a warning that a conflict exists. Specifying -no_warnings suppresses messages reporting conflicts. In either case, the command places a link to the first object of a conflicting pair in any of the directories where the object name does not appear. Specifying -no_warnings also enables the link_dirs command to run much faster over large directories where most objects have already been linked, because the command does not need to verify that all existing links point to the same object.

Specifying -depth limits the number of levels in the directory hierarchy in which links are to be created. The value of -depth increases by 1 for each subdirectory.

**Examples**

Suppose that the directory %s1#d02>Sales>east contains the directories Smith and Jones, and the directory %s1#d02>Sales>south contains the directories Clark and Rogers.

Suppose also that the current directory is %s1#d02>Sales. The command link_dirs east south produces this output.

```
%s1#d02>sales>east>Clark  ->  %s1#d02>sales>south>Clark
%s1#d02>sales>south>Jones  ->  %s1#d02>sales>east>Jones
%s1#d02>sales>east>Rogers  ->  %s1#d02>sales>south>Rogers
%s1#d02>sales>south>Smith  ->  %s1#d02>sales>east>Smith
```
The directory `east` now contains these links.

- Clark -> %s1#d02>Sales>south>Clark
- Rogers -> %s1#d02>Sales>south>Rogers

The directory `south` contains these links.

- Jones -> %s1#d02>Sales>east>Jones
- Smith -> %s1#d02>Sales>east>Smith
list

Purpose
This command lists the contents of a directory.

Display Form

```plaintext
------------------------------------- list -------------------------------------
path_name: [ ]
-files: yes
-dirs: no
-links: no
-sort: name
-full: no
-names_only: no
-totals: no
-header: no
-exclude: 
-quotes: yes
```

Command Line Form

```plaintext
list [path_name]
    [-no_files]
    [-dirs]
    [-links]
    [-all]
    [-sort sort_code]
        [-full]
        [-names_only]
        [-totals]
    [-header]
    [-exclude exclude_name]
    [-no_quotes]
```

Arguments

- **path_name**
  Specifies the name or star name of the object or objects to be listed. The `list` command lists all objects that have matching names and are of the types specified by `-files`, `-dirs`, `-links`, and `-all`. If you specify a `path_name` and omit `-files`,
-dirs, -links, and -all, the list command lists only files. By default, the command lists all objects of the specified types in your current directory.

- **-no_files**
  Suppresses a listing of all files whose names match `path_name` or, if you omit the `path_name` argument, of all files in the current directory. Use this argument in combination with -dirs or -links if you do not want to list files and either directories or links (but not both). If you do not use this argument and specify -dirs and/or -links, the command lists only the directories and/or links. By default, the command lists files only.

- **-dirs**
  Lists all directories whose names match `path_name` or, if you omit `path_name`, all directories in the current directory. By default, the command lists no directories.

- **-links**
  Lists all links whose names match `path_name` or, if you omit `path_name`, all links in the current directory. By default, the command lists no links.

- **-all**
  Lists all files, directories, and links whose names match `path_name` or, if you omit `path_name`, all objects in your current directory. By default, the command lists files only.

- **-sort sort_code**
  Sorts the objects according to the code `sort_code` before listing them. There are six possible values for `sort_code`.

  - name
  - size
  - date_created
  - date_modified
  - date_used
  - date_saved

  If you specify any sort code except name, the list command sorts the objects numerically on the attribute specified. If you specify name, the command sorts the object names according to a variation of the ASCII collation sequence. By default, the command sorts the objects by name.

- **-full**
  Displays more information about the objects listed. You cannot specify -full with -names_only or -totals.

- **-names_only**
  Displays only the names of the objects listed. You cannot specify -names_only with -full or -totals.

- **-totals**
  Displays only the number of specified objects and the number of disk blocks used. By default, the command displays the names of each object, the number of disk blocks it
uses, and your access level to the object. You cannot specify -totals with -full or
-names_only.

* -header [CYCLE]
  Displays, at the top of the list, the path name of the directory that contains the objects
  in the list.

* -exclude exclude_name
  Specifies one or more object names or star names to be excluded from the list.

* -no_quotes [CYCLE]
  Suppresses the default display in which names containing special command-line
  characters are enclosed by apostrophes. Names beginning with a hyphen are displayed
  with the prefix . > (that is, the period and greater-than characters). See Using OpenVOS
  Extended Names (R631) for more information about special command-line characters.

Explanation

The list command displays information about some or all of the contents of a directory.

Use path_name to specify the objects you want to list and, indirectly, their containing
directory. The -files, -dirs, -links, and -all arguments determine the types of objects
you want to list. If you omit all four of these and do not specify the name of a specific
directory or link for path_name, the list command displays only files.

The list command is particularly useful for checking the contents of your current directory.
When you invoke the list command without any arguments, your terminal displays a list of
all files in your current directory. If you invoke the command and specify only path_name,
for which you give a star name, you see a list of all files with matching names. When you
invoke list with only path_name, but give a path name that is not a star name, you see a
list containing just that file, if it exists. Thus you can use the list command to check whether
a specific file is in the current directory without listing its entire contents. Even if you do not
know the precise name of the file, you can check for it by giving a star name as an
approximation. Note that you must usually specify -dirs, -links, or -all to list objects
other than files. The exception is that if you specify only path_name and give a path name
that is the exact name of an existing directory or link, the list contains just that object. Note
also that when you give the path name of a directory, the command displays only the directory
name and not its contents.

To list the contents of a directory other than the current directory, you must enter a command
that contains the full or relative path name of the directory, for example:

list >Sales>weekly>reports>*

The list command groups objects by type when it lists them. Within a group, it sorts objects
by name, unless sort specifies a different sort code. In a name sort, the objects are sorted
alphabetically, with an uppercase letter preceding its corresponding lowercase letter. For
example, AB precedes Ab, which precedes aB, which precedes ab.

The list command sorts all other types numerically on the attribute specified.
The following information on files is displayed:

- the number of files listed
- the total number of disk blocks used by all the files
- your access to each file: read, write, execute, null, or undefined. (Access is considered undefined when none of the standard access types is appropriate. Undefined access defaults to null.)
- the number of disk blocks used by each file
- the name of each file
- if you specify `-full`, the date the file was last modified or the date specified in `sort_code`; the (maximum) record size of each file, if it is a relative or fixed file; and file organization

The following information on directories is displayed:

- the number of directories listed
- your access to each directory--status, modify, null, or undefined. (Access is considered undefined when none of the standard access types is appropriate. Undefined access defaults to null.)
- the number of disk blocks used by each directory
- the name of each directory

If you specify `-full`, the operating system also displays the date the directory was last modified or the date specified in `sort_code`.

The following information on links is displayed:

- the number of links listed
- the immediate target of each link
- the name of each link

If you specify `-full`, the operating system also displays the date the link was last modified or the date specified in `sort_code`.

**Access Requirements**

You need status access to the directory containing the listed objects to invoke this command.

**Examples**

**Example 1.**
The following command lists the files in the current directory whose names start with the character string *s*.

```
list s* -names_only
```
The files are sorted by name, and only the names appear.

```
sales_class
sales_closed
sales_leads
sales_pending
september_goals
status
```

**Example 2.**

Now consider the following command.

```
list weekly_old\* week* -all -sort date_created
```

This command might display the following list.

```
Files: 5, Blocks: 27.

w 08 90-03-13 10:51:09 week90-03-03
w 06 90-03-05 10:30:22 week90-02-25
w 04 90-02-26 10:43:59 week90-02-18
w 05 90-02-21 11:28:10 week90-02-11
w 04 90-02-13 10:35:26 week90-02-04

Directories: 0
Links: 0
```

**Example 3.**

If you list an object whose name uses special command-line characters, by default, the name is surrounded by apostrophes in the output. Also, by default, if the object’s name begins with a hyphen, the output is shown with the prefix `\>`. For example:

```
m102: list *test -names_only
\>hyphentest
'a btest'
'a name with "quotes" test'
```

If you intend to parse the output of the `list` command and your program is not prepared to deal with an apostrophe escape convention or `\>`, you may want to specify `-no_quotes`, as follows:

```
m102: list *test -names_only -no_quotes
\>hyphentest
a btest
a name with "quotes" test
```

However, problems may result from using names that contain spaces, actual apostrophes, or hyphens (for example, if `list` assumes these characters as delimiters). OpenVOS provides command-macro statements (`&dcl_name`, `&set_name_string`, and `&set_name`) for
handling names containing spaces and other special characters. See the *OpenVOS Commands User’s Guide* (R089) for more information.
list_batch_requests

Purpose

This command displays information about a set of batch requests you specify.

Display Form

```
---------------------------------------- list_batch_requests ----------------------------------------
-queue: normal
-module: 
-all: no
-long: no
-user: 
-process: 
```

Command Line Form

```
list_batch_requests [-queue queue_name]
               [-module module_name]
               [-all]
               [-long]
               [-user user_name]
               [-process process_name]
```

Arguments

- `-queue queue_name`
  Specifies the batch queue `queue_name` containing the batch requests you want to list. By default, the command displays the requests in the default batch queue, either on the module specified by `module_name` or on the current module.

- `-module module_name`
  Specifies the module serving the queue in which the batch requests are waiting. By default, the command uses your current module.

- `-all`  
  Displays information about all batch requests in the specified batch queue. You must have read access to the pertinent queue to list all batch requests. If you have only execute access to the queue, the command lists only your batch requests. By default, the command lists only your batch requests.
list_batch_requests

- **-long**
  Displays detailed information about the batch requests in the specified batch queue.

- **-user user_name**
  Specifies the name or star name of the user whose batch requests you want to list. By default, the command lists batch requests of all users. You cannot specify -user and -all in the same command.

- **-process process_name**
  Specifies the name or star name of the process whose batch requests you want to list. By default, the command lists batch requests of all processes. You cannot specify -process and -all in the same command.

**Explanation**

The `list_batch_requests` command lists information about a specified set of batch processes.

When you omit -long, the command displays:

- the name of each batch request
- the queue sequence number of each batch request
- the time each batch request was submitted
- the state of each batch request
- the command line to be executed for each batch process selected, if you omit -all
- the name of the user who submitted the request, if you specify -all

The `list_batch_requests` command indicates that a batch process is running by displaying an asterisk in the line that describes the request or by printing a message stating that the batch process is being executed.

If you specify -all and you have read access to the pertinent batch queue, the `list_batch_requests` command lists all batch requests for that queue. However, if you specify -all but you have only execute access, the command lists only your batch requests. In either case, the command displays the name of the user who issued a request instead of the command line from that request.

If you specify -long, the `list_batch_requests` command also displays the following information.

- the time to which the request is deferred, if any
- the path name of the output file or device
- the queue priority
- the batch process priority
- the error codes file to be used (a file other than the default error codes file)

**Access Requirements**

You must have read access to the batch queue file to list all batch requests. If you have only execute access to the queue file, the `list_batch_requests` command lists your batch requests.
Examples

Example 1.

The command list_batch_requests -all might display the following output.

<table>
<thead>
<tr>
<th>Process Name</th>
<th>#</th>
<th>Time</th>
<th>Options</th>
</tr>
</thead>
<tbody>
<tr>
<td>weekly_sales_report</td>
<td>9670</td>
<td>00:02</td>
<td>Jones.SysAdmin (Deferred)</td>
</tr>
<tr>
<td>daily_functions</td>
<td>9756</td>
<td>00:02</td>
<td>Overseer.System (Deferred)</td>
</tr>
<tr>
<td>monthly_expenses</td>
<td>9759</td>
<td>12:33</td>
<td>*Smith.Sales</td>
</tr>
</tbody>
</table>

Example 2.

If you issued the command list_batch_requests -all -long, the following output might be displayed.

<table>
<thead>
<tr>
<th>Request: 9670</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>User: Jones.SysAdmin</td>
<td></td>
</tr>
<tr>
<td>Time queued:</td>
<td>17-01-26 09:16:49 est</td>
</tr>
<tr>
<td>Deferred until:</td>
<td>17-01-27 00:01:00 est</td>
</tr>
<tr>
<td>Attributes:</td>
<td>deferred, not privileged, restart</td>
</tr>
<tr>
<td>Process priority:</td>
<td>5</td>
</tr>
<tr>
<td>Queue priority:</td>
<td>4</td>
</tr>
<tr>
<td>Process name:</td>
<td>weekly_sales_report</td>
</tr>
<tr>
<td>Command:</td>
<td>weekly_sales_report.cm</td>
</tr>
<tr>
<td>Output path:</td>
<td>%s1#d01&gt;Sales&gt;reports&gt;weekly_sales_report.out</td>
</tr>
<tr>
<td>Current dir:</td>
<td>%s1#d01&gt;Sales&gt;Reports</td>
</tr>
<tr>
<td>Home dir:</td>
<td>%s1#d01&gt;Sales&gt;Jones</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Request: 9756</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>User: Overseer.System</td>
<td></td>
</tr>
<tr>
<td>Time queued:</td>
<td>17-01-26 10:02:13 EDT</td>
</tr>
<tr>
<td>Deferred until:</td>
<td>17-01-28 00:01:00 EDT</td>
</tr>
<tr>
<td>Attributes:</td>
<td>waiting, privileged, restart</td>
</tr>
<tr>
<td>Process priority:</td>
<td>5</td>
</tr>
<tr>
<td>Queue priority:</td>
<td>4</td>
</tr>
<tr>
<td>Process name:</td>
<td>daily_functions</td>
</tr>
<tr>
<td>Command:</td>
<td>daily_functions</td>
</tr>
<tr>
<td>Output path:</td>
<td>%s1#d01&gt;Overseer&gt;daily_functions.out</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Request: 9759</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>User: Smith.Sales</td>
<td></td>
</tr>
<tr>
<td>Time queued:</td>
<td>17-01-26 11:10:13 EDT</td>
</tr>
<tr>
<td>Attributes:</td>
<td>executing (on %s1#d01), not privileged, restart</td>
</tr>
<tr>
<td>Process priority:</td>
<td>1</td>
</tr>
<tr>
<td>Queue priority:</td>
<td>4</td>
</tr>
<tr>
<td>Process name:</td>
<td>monthly_expenses</td>
</tr>
<tr>
<td>Command:</td>
<td>&gt;Sales&gt;tools&gt;monthly_expenses.cm</td>
</tr>
<tr>
<td>Output path:</td>
<td>%s1#d01&gt;Sales&gt;reports&gt;monthly_expenses</td>
</tr>
<tr>
<td>Home dir:</td>
<td>%s1#d01&gt;Sales&gt;Smith</td>
</tr>
<tr>
<td>Notify users:</td>
<td>Smith.Sales on %s1#*</td>
</tr>
</tbody>
</table>
Example 3.
If you issued the command `list_batch_requests -process *.*` the following output might be displayed.

<table>
<thead>
<tr>
<th>Process Name</th>
<th>#</th>
<th>Time</th>
<th>Options</th>
</tr>
</thead>
<tbody>
<tr>
<td>report.analysis</td>
<td>03</td>
<td>09:17</td>
<td>Jones.SysAdmin (Deferred)</td>
</tr>
</tbody>
</table>

Related Information
See also the command descriptions of `batch`, `cancel_batch_requests`, `cancel_device_reservation`, `display_batch_status`, `move_device_reservation`, `update_batch_requests`, and `reserve_device`. 
list_devices

Purpose
This command displays a list of the path names of a specified set of devices in a specified set of modules.

Display Form

```
--------------------------------- list_devices ---------------------------------
   -module: [ ]
   -type: window_term
   -long: no
```

Command Line Form

```
list_devices [-module module_name]
   [-type device_type]
   [-long]
```

Arguments

- **-module module_name**
  One or more module names or module star names. The command lists the devices connected to the specified modules. By default, the command uses your current module.

- **-type device_type**
  Lists the path names of devices of the type device_type. Possible values for device_type are as follows:

  - vterm
  - window_term
  - server
  - tape
  - streams
  - streams_pci

If your module is part of a multiple-module system that supports various releases of VOS and OpenVOS, device_type supports additional values. See the manual OpenVOS System Administration: Configuring a System (R287) for information about types of devices.
list_devices

By default, the command lists the path names of window_term devices.

- **-long**
  Displays more information about the selected set of devices.

**Explanation**

The `list_devices` command lists the path names of all devices of a given type connected to a specified set of modules.

If you specify `-long`, `list_devices` displays the following information for each device, in addition to the path name:

- device type
- baud rate
- chassis slot in which the controller for the device is located
- channel number on the communications chassis to which the device is connected
- module to which the device is connected
- primary user, if one is logged in

**Examples**

The following example displays information about the terminal `t2.5` connected to slot 5 on `%s1#m2`.

```
%s1#t2.5
  Type:  terminal
  Baud:  9600
  Slot:  5
  Channel:  5
  Module:  %s1#m2
  Primary user:  Clarke
```

The primary user is the person name of the current user of the device, if any; if no one is currently using the device, no primary user is listed.
list_dynamic_dependencies

Purpose
This command lists all of the shared libraries on which the specified program module or shared library depends.

Display Form

```
------------------------- list_dynamic_dependencies -------------------------
pathnames:
```

Command Line Form

```
list_dynamic_dependencies pathname [pathname...]
```

Arguments

- `pathname [pathname...]`
  One or more program module or shared library names. This value must be an OpenVOS-style path name.

Explanation

The `list_dynamic_dependencies` command lists all of the shared libraries on which `pathname` depends. The command displays OpenVOS-style path names in its output.

Examples

An example of `list_dynamic_dependencies` follows:

```
list_dynamic_dependencies salestest.pm
  libvosposix.1.so => %s1#m2>lib>libvosposix.1.so
```

In the output, `libvosposix.1.so` is the shared library on which `salestest.pm` depends, and `%s1#m2>lib>libvosposix.1.so` is the location of the shared library.

Related Information

See the description of the `ldd` command.
**list_gateways**

**Purpose**
This command lists network gateways in the system.

**Display Form**

```
------------------------------- list_gateways -----------------------------
No arguments required. Press ENTER to continue.  
```

**Command Line Form**

`list_gateways`

**Explanation**
The `list_gateways` command lists network gateways in the system.

A gateway provides a communication link from the current system to one or more other systems. This link can be directly connected to another system or it can be connected to a common network shared by multiple systems.

The gateways listed are defined in the `nodes.tin` file for the current system. If there are no gateways in the system, nothing is displayed.
list_library_paths

Purpose
This command displays the list of directories that define libraries for the current process.

Display Form

```
---------------------------------------- list_library_paths ----------------------------------------
library_name: all
```

Command Line Form

```
list_library_paths [library_name]
```

Arguments

- `library_name` [CYCLE]
  Specifies the library whose list of directories is to be displayed. There are five possible values for `library_name`.
  - include
  - object
  - command
  - message
  - all

  By default, the command lists all directories that define libraries for the current process.

Explanation

The `list_library_paths` command displays a list of directories that define the libraries for the current process.

A library is one or more directories that the operating system searches for objects of a particular type. Each module has the following libraries.

- include library
- object library
- command library
- message library
The compilers search the include library for include files; the binder searches the object library for object modules; and the command processor searches the command library for commands and the message library for messages associated with individual commands.

A library library_name is defined by an ordered sequence of path names. The order of the list reflects the order in which the operating system searches the libraries for objects of a designated type.

Examples

The following example illustrates a typical list of libraries. Each directory of a library is specified by its full path name. The order in which the directories are listed reflects the order of the search.

include library directories:
  (current_dir)
  %s1#d02>system>include_library

object library directories:
  (current_dir)
  %s1#d02>system>object_library

command library directories:
  (current_dir)
  %s1#d02>system>command_library
  %s1#d02>system>tools_library
  %s1#d02>system>applications_library

message library directories:
  (referencing_dir)
  %s1#d02>system>message_library>(language_name)

Related Information

For more information about libraries, see the descriptions of the list_default_library_paths and the set_default_library_paths commands in OpenVOS System Administration: Administering and Customizing a System (R281). See also the command descriptions of add_library_path, delete_library_path, and set_library_paths. See OpenVOS Commands User's Guide (R089) for information about the conventions for searching libraries.
list_modules

Purpose

This command lists one or all of the modules in a specified system.

Display Form

```
--------------------------------- list_modules ---------------------------------
 module_name:           | -all:     no |
                      -brief:  no |
                      -long:   no |
```

Command Line Form

```
list_modules [module_name]

[ -all ]
[ -brief ]
[ -long ]
```

Arguments

- **module_name**
  Specifies a module name or star name. The command lists all the module names in the specified system, with information about which modules are online and which are offline. By default, `list_modules` lists this information for the modules in the current system.

- **-all**
  Lists the names of all modules in all of the systems accessible to the current system. By default, the command lists the names of the modules in the current system.

- **-brief**
  Lists only the names of the modules in the specified system.

- **-long**
  Displays, for any offline module in the specified system, information about why the module is offline.
list_modules

Explanation

The list_modules command lists the names of the modules in a specified system. You can specify a module name or select -all. If you specify neither, the list_modules command lists the modules in the current system.

The list_modules command indicates your current module and shows which modules are not communicating with the current module over an OpenVOS communications network by displaying offline next to their names.

Examples

Example 1.
The list_modules command with no arguments specified might display the following information.

%s1#m1 . . . . . . . . . . online  
#m2 . . . . . . . . . . online (current module)  
#m3 . . . . . . . . . . offline

Example 2.
The list_modules -brief command might display the following list of module names.

%s1#m1  
#m2  
#m3

Example 3.
The list_modules -long command might display the following information.

%s1#m1 . . . . . . online  
#m2 . . . . . . online (current module)  
#m3 . . . . . . The target module is offline.  
#m4 . . . . . . You are not a registered user of the target system.
list_port_attachments

Purpose
This command lists information about a set of ports you specify in your process.

Display Form

<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>port_names:</td>
<td>list_port_attachments</td>
</tr>
<tr>
<td>-number:</td>
<td></td>
</tr>
<tr>
<td>-last_port_number:</td>
<td></td>
</tr>
<tr>
<td>-pathname:</td>
<td></td>
</tr>
</tbody>
</table>

Command Line Form

list_port_attachments [port_names]...

[-number number]
[-last_port_number last_port_number]
[-pathname path_name]

Arguments

- **port_names**
  Specifies one or more names of ports about which you want information displayed. By default, the command displays information about all ports in your process.

- **-number number**
  Specifies a port ID number. The number must be between 1 and 4096. If you also specify -last_port_number, -number specifies the start of a range of port IDs, and -last_port_number specifies the end of a range. Both port IDs must refer to the same range, either low or high. If you specify -number but do not specify -last_port_number, -number specifies a single port ID. The command displays information about the specified port ID(s). You cannot specify the -number argument together with the port_names argument.

- **-last_port_number last_port_number**
  Specifies the highest port ID number of the ports about which you want information. The highest permissible value is 4096. If you also specify -number, it specifies the start of a range of port IDs; otherwise, the range starts at 1. The command displays information about ports in the specified range. If the value you specify is too high, the command lowers it to that of the highest numbered port in the range specified by
list_port_attachments

-number. You cannot use both the -last_port_number and -port_names arguments.

If the value you specify for -number is greater than that for -last_port_number, then -last_port_number is set equal to -number.

-pathname path_name

Specifies the full or relative path name of a file or device to which the current process has attached one or more ports. The command displays information about these ports. You can use this argument together with any of the preceding arguments to specify only those ports attached to a particular file or device.

Explanation

The list_port_attachments command displays the following information.

- the name of the port
- the path name of the port’s attachment
- the type of the attachment
- the I/O type of the port
- the access mode of the port

If you do not specify port_name or any other arguments, the command displays information about all ports.

If the attachment is a sequential file and the port is open, the command also displays the next available byte.

If the attachment is an extended sequential file and the port is open, the command displays the number of record offset units in brackets after the file organization (for example, Type: sequential file [4] indicates that there are four record offset units). This bracketed value appears in the output only for extended sequential files.

If the attachment is a relative file or a fixed file and the port is open, then the following additional information is included.

- the (maximum) record size of the file
- the record number of the current record
- the record number of the last record written

For files of any type, when the port is open, the command displays the number of disk blocks used.
Common port attachment types include the following:

- unknown
- fixed file
- relative file
- sequential file
- stream file
- stream file (64-bit)
- sequential file \([N]\)
- printer
- tape drive
- terminal
- window terminal (64-bit stream file)

Common port I/O types include the following:

- closed
- input
- output
- append
- update

Common port access modes include the following:

- closed
- sequential
- random
- indexed

Output from the `list_port_attachments` command displays attributes of the port, as illustrated in the examples later in this command description. A description of these attributes follows.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>wait mode</td>
<td>The system waits for I/O on this port; if tasking is in use, no task switches will occur.</td>
</tr>
<tr>
<td>no wait mode</td>
<td>The user is responsible for waiting for I/O on the port.</td>
</tr>
<tr>
<td>tasking mode</td>
<td>The system waits for I/O on this port; if tasking is in use, task switches may occur.</td>
</tr>
<tr>
<td>hold attachment</td>
<td>The port was attached with <code>hold attachment</code> specified (see <code>s$attach_port</code> in the OpenVOS Subroutines manuals) and will not be detached at program termination.</td>
</tr>
<tr>
<td>hold open</td>
<td>The port was opened with <code>hold open</code> specified (see <code>s$attach_port</code> in the OpenVOS Subroutines manuals) and will not be closed at program termination.</td>
</tr>
<tr>
<td>remote</td>
<td>The port is attached to a device or file on another module.</td>
</tr>
</tbody>
</table>
Examples

Example 1.
Suppose you have a port named `a_port` in your process that is attached to the file `%s1#d02>sales>smith>file_27`. The command `list_port_attachments a_port` displays the following information about the specified port.

```plaintext
a_port
Pathname:  %s1#d02>sales>smith>file_27
Type:      sequential file
I/O type:  update
Access mode: sequential
Attributes: wait mode, hold attached and open
  Cur byte offset:  0
  Next available byte:  1210
  Disk blocks:  1
```

Example 2.
The command `list_port_attachments -number 10` displays the following information about the port whose ID is 10.

```plaintext
_aaaweXreo9SaWo7P.log (10)
Pathname:  %s1#d02>sales>smith>file_30
Type:      sequential file
I/O type:  output
Access mode: sequential
Attributes: wait mode, hold attached and open
  Cur byte offset:  0
  Next available byte:  2652
  Disk blocks:  1
```

Example 3.
The command `list_port_attachments -number 10 -last_port_number 20` displays the following information about ports located in the range of IDs 10 and 20.

```plaintext
_aaaweXreo9SaWo7P.log (10)
Pathname:  %s1#d02>sales>smith>file_30
Type:      sequential file
I/O type:  output
Access mode: sequential
Attributes: wait mode, hold attached and open
  Cur byte offset:  0
  Next available byte:  3106
  Disk blocks:  1
```

(Continued on next page)
Example 4.
The command `list_port_attachments -pathname a_file -last_port_number 7` displays the following information about the file whose path name matches the one specified and whose port ID is less than or equal to 7.

```plaintext
Pathname:   %s1#d02>Sales>Smith>a_file
Type:       sequential file
I/O type:   input
Access mode: indexed
Attributes: wait mode, hold attached and open
Cur byte offset:       67470
Next available byte:   287594
Disk blocks:           314
```
list_print_requests

Purpose
This command displays information about a specified set of print requests.

Display Form

```
-queue: Standard
-module:
-all: no
-long: no
```

Command Line Form

```
list_print_requests [-queue queue_name]
[-module module_name]
[-all]
[-long]
```

Arguments

- **-queue queue_name**
  Specifies the print queue whose requests you want to list. By default, the command displays the requests in the default print queue, either on the module specified by `module_name` or on the current module.

- **-module module_name**
  Specifies the module containing the specified queue. By default, the command uses your current module.

- **-all**
  Displays information about all the print requests in the designated print queue. By default, the command lists your print requests only.

- **-long**
  Displays additional information about the print requests. By default, the command displays the information listed in the Explanation section of this command description.
Explanation

The `list_print_requests` command displays information about a specified set of print requests.

When you omit `-long`, the `list_print_requests` command displays the following.

- the name of each file to be printed
- the queue sequence number of each print request
- the time you made each print request
- the state of each print request
- the number of copies requested
- the size, in blocks, of each file to be printed

If you specify `-long`, the additional information is the full path name of the file to be printed and the values of all the `print` command arguments selected for each print request. See the `print` command for a list of the arguments. If you do not select a value for the `-destination` argument of `print`, `list_print_requests` displays `Destination` and the value `None`.

If you specify `-all`, the `list_print_requests` command also displays the names of the users who made the print requests. Use the `display_print_status` command to determine the current state of the various printers on your system; then use the `list_print_requests` command with `-all` to find out where the load is heaviest.

Examples

The following example illustrates the information displayed for two print requests when you specify `-all` and `-long` with the `list_print_requests` command.

```
list_print_requests -all -long -module Sales

Request:        01
User:           Smith.Sales
Destination:    Smith
Time:           90-03-16 10:20:45 EDT
Path Name:      %s1#d02> Sales> Smith> sales_by_rep
Queue Priority: 4

Attributes: printing on %s1#p27
Title:          none.
Header:         none.
Footer:         none.

Copies:         01 File size:       74 Sort index:   none.
Line length:    00 Page size:       00 Indent:       00
Line numbers:   no Page Breaks:     yes Delete:      no
Edited:         no Wrap:           yes Controls:   canonical
```

(Continued on next page)
list_print_requests

(Continued)

Request: 02
User: Smith.Sales
Destination: None
Time: 90-03-16 10:20:59 EDT
Path Name: %s1#d02>Sales>Smith>sales_by_region
Queue Priority: 4

Attributes: waiting
Title: none.
Header: none.
Footer: none.

Copies: 01 File size: 115 Sort index: none.
Line length: 00 Page size: 00 Indent: 00
Line numbers: no Page Breaks: yes Delete: no
Edited: no Wrap: yes Controls: canonical

Note that only the first request is running; the second request is waiting in the queue.
list_process_cmd_limits

Purpose
This command lists the initial and maximum limits for commands executed by an existing process.

Display Form

```
------------------------ list_process_cmd_limits------------------------
process_name: [process_name]
-user: current_user
-module: current_module
```

Command Line Form

```
list_process_cmd_limits [process_name]

[-user user_name]
[-module module_name]
```

Arguments

- **process_name**
  The name or star name of existing processes for which you want to list the process initial and maximum limits by each new command that the process runs. By default, if the value of -user is the current_user and the value of -module is current_module, then the command lists the process limits of the current process. Otherwise, the default process name is an asterisk (*), indicating all processes of the specified user on the specified module.

- **-user user_name**
  Specifies a name or user star name indicating one or more user names for which you want to list the initial and maximum process limits. By default, the command uses your user name.

- **-module module_name**
  Specifies the module on which the processes are running. By default, the command uses the current module.

Explanation
The list_process_cmd_limits command lists the initial and maximum command limits for an existing process.
Command limits control the amount of a resource (specifically, stack space, heap space, total address space, CPU time, stream file size, keep module size, and the number of attached ports) that an executing program can use. A system administrator can adjust the limits to prevent runaway programs from using excessive resources or simply to enforce a degree of fairness on the use of system resources.

The module’s default command limits are the limits that apply to all processes that are created on the module. These limits, once inherited by each newly-created process, become the process command limits for each process. A system administrator can change these limits with the following commands:

- The update_default_cmd_limits command changes the module’s default command limits. Any changes apply only to subsequently-created processes.

- The update_process_cmd_limits command changes the process command limits.

Each time that OpenVOS executes a program, it copies the process command limits into the program command limits. The program command limits apply only to the currently-executing program.

Each default or process command limit has an initial value and a maximum value, and each program command limit has a current value and a maximum value. When each program command limit is established, the current program value is taken from the initial process value, and the maximum program value is taken from the maximum process value.

The current program value specifies the limit on the amount of a resource that can be obtained by the executing program. A program can dynamically raise its current limit up to its maximum limit by using the s$set_current_cmd_limit subroutine. A program can also lower its maximum limit down to its current limit. Only a privileged user, or the root user, can raise one of its maximum limits.

The current program value of a limit can never be set below its current usage, nor can it be raised above its maximum value.

The maximum size of the user address space of a process is 2048 megabytes (MB). OpenVOS reserves 128 MB for the executing program, and it reserves approximately 2 MB for system use. By default, approximately 996 MB are reserved for dynamically-allocated shared virtual memory, and approximately 1022 MB is reserved for the combined size of the heap and stack. Arguments of the bind command enable you to adjust these default values.

For a description of the process command limits, see the description of the update_process_cmd_limits command.

Issue the list_users command to obtain a list of existing process names and user names.
Sample Output

The following shows sample output of the `list_process_cmd_limits` command.

```
Process command limits of list_process_cmd_limits.
Initial total limit:   infinity.
  Initial heap limit:   infinity.
  Initial stack limit:  8388608.
  Initial cpu limit:    infinity.
  Initial file limit:   infinity.
  Initial keep limit:   infinity.
  Initial port limit:   4096.
Maximum total limit:  infinity.
Maximum heap limit:   infinity.
Maximum stack limit:  132513792.
Maximum cpu limit:    infinity.
Maximum file limit:   infinity.
Maximum keep limit:   infinity.
Maximum port limit:   4096.
```

Related Information

To specify the initial and maximum resource limits for any program module executed within an existing process, use the `update_process_cmd_limits` command. For information about setting the heap (max_heap_size), stack (max_stack_size), and total (max_program_size) limits in a bound program module, see the `bind` command description. For information about setting and listing the default resource limits for all new processes on a module, see the descriptions of the `update_default_cmd_limits` and `list_default_cmd_limits` commands in *OpenVOS System Administration: Administering and Customizing a System* (R281) For information about the $get_current_cmd_limit and $set_current_cmd_limit subroutines, see the OpenVOS Subroutines manuals.
list_save_tape

Purpose

This command lists the contents of a save tape or a disk file.

Display Form

Command Line Form

```
list_save_tape tape_device_or_port_name
    [-volume_id volume_name]
    [-unattended]
```

Arguments

- `tape_device_or_port_name`  
  The name of a tape device, or the name of a port attached to a tape drive, holding the tape to be listed, or the name of a port attached to a disk file.

- `volume_id volume_name`  
  Specifies the volume ID of the first tape volume on which file system objects are saved.

- `unattended`  
  Causes tape drives with automatic loaders to switch from one tape to the next, without user intervention. This argument has no effect on tape drives for ftServer modules.

Explanation

The list_save_tape command lists the contents of a save tape or of a disk file containing one or more saved objects. When you save an object with the save or save_object command, the operating system saves the full path name of each object saved on the tape or disk file. You can use the list_save_tape command to find the path names of the objects you want to restore to the file system. You can then use the restore or restore_object command to restore the objects to the file system.

Before restoring an object, particularly when you do not need to restore the entire save tape, issue the list_save_tape command to list on your terminal the path names of all the saved objects. From that list, you can determine the correct path names to specify in the restore or restore_object command.
If you are listing the contents of a save tape and have not yet attached a port with the `attach_port` command, the `list_save_tape` command implicitly attaches a port. If you have not yet mounted a save tape with the `mount_tape` command, the `list_save_tape` command automatically mounts the save tape before executing. Unlike the other commands that automatically mount tapes, the only prompt that `list_save_tape` issues is that for the volume ID. When execution is completed, if `list_save_tape` implicitly attached a port, it implicitly detaches the port, forcing the tape to be unloaded. For more information, see Explanation in the `mount_tape` command description.

If you are listing the contents of a disk file, you must first attach a port with `attach_port`. Then specify the port name for the `tape_device_or_port_name` argument of `list_save_tape`. After `list_save_tape` is executed, you can either restore the object with `restore_object`, or detach the port with `detach_port`.

**Related Information**

The `save` and `restore` commands are described in *OpenVOS System Administration: Backing Up and Restoring Data* (R285). The `save_object` and `restore_object` commands are described later in this manual. See the description of the `restore_object` command for an explanation of how to specify the names of the objects you wish to restore. Unless you are backing up all or a substantial part of the system, `save_object` and `restore_object` are sufficient.

See also the command descriptions of `copy_tape`, `create_tape_volumes`, `dismount_tape`, `display_tape_params`, `dump_tape`, `mount_tape`, `position_tape`, `read_tape`, `restore_object`, `save_object`, `set_second_tape`, `set_tape_drive_params`, `set_tape_mount_params`, `set_tape_file_params`, `verify_save`, and `write_tape`. 
list_systems

Purpose
This command lists the systems accessible from your module.

Display Form

```
list_systems
-brief: no
-long: no
```

Command Line Form

```
list_systems [-brief]
[-long]
```

Arguments

- `-brief`
  Lists only the names of the systems accessible from the current module.

- `-long`
  Displays, for any system that is offline, additional information about why the system is unavailable.

Explanation
The `list_systems` command lists the names of the systems accessible from the current module, with additional information about which systems are online and which systems are offline.

If you specify `-brief`, only the names of the systems are listed. By default, the systems are listed with a comment telling whether each is online or offline. If you specify `-long`, you are given the reason that a system is offline.

**Note:** If you specify both `-brief` and `-long`, `-brief` overrides `-long`.

2-502
Examples

Example 1.
If you give the command `list_systems -brief`, the following output is displayed.

```
s1     s2     s3     s4     s5
s6
```

Example 2.
If you give the command `list_systems`, the following output is displayed.

```
s1 . . . . . . . . . online
s2 . . . . . . . . . offline
s3 . . . . . . . . . online (current system)
s4 . . . . . . . . . online (local)
s5 . . . . . . . . . offline
s6 . . . . . . . . . offline
```

Example 3.
The following example illustrates the information displayed when you give the `list_systems` command specifying `-long`.

```
s1 . . . . . . . . . online
s2 . . . . . . . . . The target server is not in operation.
s3 . . . . . . . . . online (current system)
s4 . . . . . . . . . online (local)
s5 . . . . . . . . . No one is listening to the specified extension.
s6 . . . . . . . . . offline
```
list_tape

Purpose
This command lists the contents of a labeled tape volume.

Display Form

```
---------------------------------- list_tape ------------------------------------
tape_device_or_port_name:   
-full:   no
```

Command Line Form

```
list_tape  tape_device_or_port_name
           [-full]
```

Arguments

- **tape_device_or_port_name**  Required
  The name of the tape device, or the name of the port attached to the magnetic tape drive, holding the tape from which the files are to be listed.

- **-full**  [CYCLE]
  Lists the tape file type, the record length, and the block length in addition to the file ID and the file number of each tape file on the tape. By default, the command lists only the tape file ID and tape file number of the tape files.

Explanation
The list_tape command lists the files of a labeled tape volume mounted on the specified tape device or on the tape drive connected to the specified port.

The list_tape command displays the volume ID and owner ID of the tape volume, lists the information about the tape files in order on the tape, and rewinds the tape after listing its contents.

If you have not yet used the attach_port command to attach a port, the list_tape command implicitly attaches a port. If you have not yet used the mount_tape command to mount a tape, the list_tape command implicitly mounts the tape before executing. When execution is completed, if list_tape implicitly mounted a tape, it implicitly dismounts the tape. If it implicitly attached a port, it implicitly detaches the port. For more information, see the Explanation section in the mount_tape command description.
Related Information

See also the command descriptions of `copy_tape`, `create_tape_volumes`,
`dismount_tape`, `display_tape_params`, `dump_tape`, `list_save_tape`,
`mount_tape`, `position_tape`, `read_tape`, `restore_object`, `save_object`,
`set_second_tape`, `set_tape_drive_params`, `set_tape_mount_params`,
`set_tape_file_params`, and `verify_save`. 
list_terminal_types

Purpose
This command lists the terminal types for which default parameter values are defined.

Display Form

```
----------------------------- list_terminal_types -----------------------------
-match:  
-module: current_module
```

Command Line Form

```
list_terminal_types [-match string]
 [ -module module_name ]
```

Arguments

- **-match string**
  Lists only the types of terminals with names that contain the character string `string`. By default, the command lists all the terminal types for which default parameters are defined.

- **-module module_name**
  Specifies a module. The command lists the types of terminals for which default parameters are defined on the specified module. By default, the command uses your current module.

Explanation

The `list_terminal_types` command lists the types of terminals for which default parameter values are defined.

By default, the command lists all terminal types. If you specify `-match`, `list_terminal_types` selects only the terminal types that contain `string` in the terminal type name. An asterisk in the `-match string` is treated as a literal character to match and does not form a star name.

If you specify `-module`, the `list_terminal_types` command lists the types of terminals for which default parameters are defined on the module you specify. The `module_name`
value **cannot** be a module star name. The following are examples of the accepted forms of `module_name`.

```
%s1#m2
#m2
m2
```

By default, the command lists the terminal types for the current module.
list_users

Purpose
This command displays information about selected processes.

Display Form

```
user_name: .*
-process: *
-module: current_module
-full: no
-interval: no
-process_id: no
-admin: no
-prelogin: no
-system: no
-more_system_user_ids: root.root nobody.nobody mysql.mysql
```

Command Line Form

```
list_users [user_name]

[-process process_name]
[-module module_name]
[-full]
[-interval [interval]]
[-process_id process_id]
[-admin [data_set]]
[-kernel]
[-prelogin]
[-system]
[-more_system_user_ids person_name[.group_name]]
```

Arguments

- **user_name**
  Specifies one or more user names or star names. The command displays information only about processes belonging to the specified user or users. By default, the command displays information about all processes satisfying the other argument criteria.
-process process_name
   Specifies one or more process names or star names. If you specify login for
   process_name, list_users displays information only about interactive processes.
   By default, the command displays information about all processes satisfying the other
   argument criteria.

-module module_name
   Specifies one or more module names or module star names. By default, the command
   uses your current module.

-full
   Displays additional information about the specified processes. This is ignored if you
   specify -admin. See the Explanation section of this command description for details.

-interval [interval]
   Displays information about the specified processes on the terminal screen and updates
   it every interval seconds. If you specify -interval, your current terminal must be a
display terminal. If you specify -interval on the command line but omit the value
   for interval, the command uses a value of 10.

-process_id
   Adds a PID column to the output to the left of the USER field, if you specify the
   -process_id argument without specifying -interval time. (The PID is the
   process ID, an 8-digit hexadecimal number, followed by an x.) See the Explanation for
   information about how the -process_id argument interacts with the -interval
   argument.

-admin [data_set]
   Displays additional information about selected processes. There are four possible
   values for data_set.

   - usage
   - page_faults
   - disk_io
   - deltas

   By default, the command uses usage for data_set even though the display form
   value for this argument is blank.

-kernel
   Includes kernel processes that match user_name and process_name. Kernel
   processes are processes created automatically by the operating system
   (Diagnostic_Utility, Kernel_Utility, and Maintenance_Utility). By
default, the command displays no information about kernel processes.

-prelogin
   Includes prelogin processes that match user_name and process_name. If you do not
   specify -prelogin, and specify either PreLogin.System or PreLogin.* or
   PreLogin for user_name or pre-login for process_name, the command includes
prelogin processes. By default, the command displays no information about prelogin processes.

- **-system**
  Includes system processes that match `user_name` and `process_name`. System processes are processes running for users whose group name is `.System`, except kernel and pre-login processes. If you specify either `user_name` or `-process`, the command displays information about system processes. By default, the command displays no information about system processes.

- **-more_system_user_ids person_name[.group_name]**
  Includes user names of the form `person_name.group_name` (group_name is optional). The default list of user names is `root.root`, `nobody.nobody`, and `mysql.mysql`. The `list_users` command displays noninteractive processes with any of these names only if you also specify the `-system` argument.

**Explanation**

The `list_users` command displays information about a selected set of processes.

The `user_name`, `-process`, `-module`, `-kernel`, `-prelogin`, `-system`, and `-more_system_user_ids` arguments determine which processes the command selects.

The `user_name` argument allows you to specify a user or set of users. The `list_users` command displays information about only the processes of these users. The following are examples of the accepted forms of the `user_name` argument.

- `Smith.Sales`
- `Smith.*`
- `.*.Sales`
- `*.*`
- `*`
- `Smith`

The default value of `user_name` is `*.*`.

The `-process` argument allows you to specify a process or set of processes. The `list_users` command displays information about only the processes whose names match `process_name`.

The `list_users` command can display the following items of data about the specified processes.
The **-full**, **-admin data_set**, **-process_id**, and **-interval** arguments determine what information **list_users** displays. Table 2-23 lists the data items that are displayed when you select various combinations of these arguments.
If you specify `-admin` and `-full`, `-full` has no effect.

If you specify `-admin` and not `-interval`, the choice of `data_set` is irrelevant. If you specify both `-admin` and `-interval`, you can specify a different `data_set` while the data is being displayed.
If you specify `-interval`, you can use the following keys to control the display.

- **BREAK**
  - Stop
- **CANCEL**
  - Stop
- **CANCEL_FORM**
  - Stop
- **CYCLE**
  - Change `-admin data_set`
- **→**
  - Change `-admin data_set`
- **CYCLE_BACK**
  - Change `-admin data_set`
- `<`
  - Change `-admin data_set`
- **DISPLAY_FORM**
  - Refresh display
- **↓**
  - Move window up a line
- **GOTO END**
  - Move window to last process displayed
- **GOTO END**
  - Move window to last process displayed. If you specify `-interval` on an EPC keyboard, the `END` key repositions the display to show information for the last process.
- **GOTO ↑**
  - Move window to first process displayed
- **GOTO BEGINNING**
  - Move window to first process displayed. If you specify `-interval` on an EPC keyboard, the `HOME` key repositions the display to show information for the first process.
- **NEXTSCREEN**
  - Move window down a page
- **PREVSCREEN**
  - Move window up a page
- **QUIT**
  - Stop
- **REDISPLAY**
  - Refresh display
- **SCROLL ↓**
  - Move window down a line
- **SCROLL ↑**
  - Move window up a line
- ↑
  - Move window down a line
**list_users**

**Examples**

**Example 1.**
The following example shows the output of the `list_users` command with the `-full` argument specified. The asterisk (*) indicates that Harris is the current user. The x indicates a stopped process.

```
Example 1.
The following example shows the output of the list_users command with the -full argument specified. The asterisk (*) indicates that Harris is the current user. The x indicates a stopped process.

s122: list_users -full

<table>
<thead>
<tr>
<th>LOGIN TIME</th>
<th>CPU</th>
<th>PF</th>
<th>USER</th>
</tr>
</thead>
<tbody>
<tr>
<td>04-04-30 11:26:31</td>
<td>0.84</td>
<td>2939</td>
<td>Fung.Sales</td>
</tr>
<tr>
<td>04-05-03 10:45:12</td>
<td>0.02</td>
<td>235</td>
<td>Harris.SysAdmin</td>
</tr>
<tr>
<td>04-04-30 11:28:18</td>
<td>2\23:17</td>
<td>1487</td>
<td>Smith.SysAdmin (UPS_MD_Daemon)</td>
</tr>
<tr>
<td>04-05-03 10:50:36</td>
<td>0.04</td>
<td>160</td>
<td>George.Sales</td>
</tr>
</tbody>
</table>
```

**Example 2.**
The following example shows the output of the `list_users` command with the `-admin` argument specified.

```
Example 2.
The following example shows the output of the list_users command with the -admin argument specified.

s122: list_users -admin

<table>
<thead>
<tr>
<th>CPU</th>
<th>PF</th>
<th>PFTIME</th>
<th>DISKR</th>
<th>DISKW</th>
<th>PS</th>
<th>F P</th>
<th>USER</th>
</tr>
</thead>
<tbody>
<tr>
<td>16.14</td>
<td>2186</td>
<td>0.07</td>
<td>961</td>
<td>4787</td>
<td>+5</td>
<td>Jones.Sales</td>
<td></td>
</tr>
<tr>
<td>0.84</td>
<td>2939</td>
<td>0.19</td>
<td>1444</td>
<td>948</td>
<td>+5</td>
<td>Fung.Sales</td>
<td></td>
</tr>
<tr>
<td>0.03</td>
<td>265</td>
<td>0.00</td>
<td>40</td>
<td>14</td>
<td>+5</td>
<td>Harris.SysAdmin</td>
<td></td>
</tr>
<tr>
<td>2\23:22</td>
<td>1487</td>
<td>0.04</td>
<td>148</td>
<td>23</td>
<td>+7</td>
<td>Smith.SysAdmin (UPS_MD_Daemon)</td>
<td></td>
</tr>
<tr>
<td>0.04</td>
<td>160</td>
<td>0.00</td>
<td>137</td>
<td>26</td>
<td>+5</td>
<td>George.Sales</td>
<td></td>
</tr>
</tbody>
</table>
```

**Example 3.**
The following example shows the output of the `list_users` command with the `-interval`, `-admin deltas`, and the `-system` arguments specified.

```
Example 3.
The following example shows the output of the list_users command with the -interval, -admin deltas, and the -system arguments specified.

s122: list_users -interval -admin deltas -system

<table>
<thead>
<tr>
<th>USERS at 11:53:46</th>
<th>DCFU</th>
<th>DPF</th>
<th>DPFTM</th>
<th>DDKR</th>
<th>DDKW</th>
<th>PS</th>
<th>F P</th>
<th>PROGRAM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jones</td>
<td>WS</td>
<td>5</td>
<td>emacs.pm</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Harris</td>
<td>RD</td>
<td>5</td>
<td>list_users.pm</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Smith (UPS_MD_Daemon)</td>
<td>9.912</td>
<td>RD</td>
<td>diag_ups_monitor.p</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>George</td>
<td>WS</td>
<td>5</td>
<td>batch_overseer.pm</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Overseer (BatchOverseer)</td>
<td>WS</td>
<td>7</td>
<td>ftpd.pm</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Overseer (osl_daemon)</td>
<td>WS</td>
<td>7</td>
<td>osl_daemon.pm</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Overseer (osl_server)</td>
<td>WS</td>
<td>7</td>
<td>osl_server.pm</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Overseer (osl_server)</td>
<td>WS</td>
<td>7</td>
<td>osl_server.pm</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

2-514
When the screen is updated, the changes that have occurred within the specified interval are displayed.

Example 4.
When you specify the -process_id argument with -interval time, the display line output becomes longer than a single line per process. To avoid this problem, you can change the view of the fields of the interval display with the use of a PID display toggle. If you press the P or p key, the display changes between the PID display and the normal display. The following example shows the normal output.

```
list_users -interval 5 -full
```

When you press the P or p key, the output changes to the PID display:

```
list_users -interval 5 -full -process_id
```

If the -admin argument was specified, the PID column toggles between the PS, F, and P columns. The following example shows the normal output.

```
list_users -interval 5 -admin
```

When you press the P or p key, the output changes to the PID display:

```
list_users -interval 5 -admin -process_id
```
list_users

Related Information
See also the description of the who_locked command.
locate_expandable_dirs

Purpose
This command identifies expandable directories.

Display Form

Umprove locate expandable_dirs Umprove locate expandable_dirs

Arguments

* path_name
The name of a directory to examine. If you specify path_name, the command examines all directories that are directly or indirectly subordinate to path_name, but it does not examine path_name itself. If you omit path_name, the command examines all disks on the module.

Explanation
The locate_expandable_dirs command identifies expandable directories. If you are a privileged user, the command forces status access on the directory. If you are not a privileged user and do not have sufficient access to the directory, the command displays a warning.

Examples
In the following example, the command is specified without path_name, so it examines all disks on the module.

locate_expandable_dirs

Checking directories in disk %s#raid4...

Checking directories in disk %s#m111_mas...
%s#m111_mas> Sales> Smith> d3>d4
%s#m111_mas> Sales> Smith> d3>d4>d5
%s#m111_mas> Sales> Smith> d3>d4>d6
%s#m111_mas> Sales> Smith> d3>d4>d6>d7
locate_expandable_dirs

In the following example, the command is specified with a path name, so it examines only #m111_mas and its subdirectories.

```
locate_expandable_dirs #m111_mas

%s#m111_mas>Sales>Smith>d1
%s#m111_mas>Sales>Smith>d3>d4
%s#m111_mas>Sales>Smith>d3>d4>d5
%s#m111_mas>Sales>Smith>d3>d4>d6
%s#m111_mas>Sales>Smith>d3>d4>d6>d7
```

Related Information

See also the description of the `locate_large_dirs` command.
locate_files

Purpose

This command lists all files in a directory subhierarchy whose names match one or more file names you specify.

Display Form

```
locate_files
```

**Arguments**

```
file_names: ...
-root_dir:  current_dir
```

Command Line Form

```
locate_files file_names... 
[root_dir directory_name]
```

**Arguments**

- **file_names**
  
  One or more names or star names of files you want to locate. The command lists the full path name of a directory that contains one or more files with matching names, followed by the names of all files in that directory with matching names.

- **-root_dir directory_name**
  
  Specifies the top directory of a subhierarchy of directories that the locate_files command is to search for files. By default, the command looks in the current directory.

Explanation

The locate_files command lists the files in a directory subhierarchy whose names match one of the names in a list of names you specify. No message is displayed if the files are not found.

Examples

Consider the following command.

```
locate_files *.cobol
```

This command lists all the files whose names end with *.cobol contained in the directory subhierarchy whose top directory is your current directory.
locate_files

Related Information

See also the descriptions of the list and walk_dir commands.
locate_indexed_files

Purpose

This command enables you to search a module for indexed files with certain characteristics.

Display Form

```
----------------------------- locate_indexed_files -----------------------------
directory_names: [directory_names]
-depth: [depth]
-min_key_size: [min_key_size]
-min_level: [min_level]
-min_keys: [min_keys]
-min_blocks: [min_blocks]
-extent_type: [extent_type]
-tp_type: [tp_type]
-verify: [verify]
-pace: [pace]
-count_keys: [count_keys]
-verbose: [verbose]
-full_path: [full_path]
-details: [details]
-brief: [brief]
```

Command-Line Form

```
locate_indexed_files [directory_names]
[ -depth number_of_subdirs]
  [ -min_key_size key_size]
  [ -min_level number_of_levels]
  [ -min_keys number_of_keys]
  [ -min_blocks number_of_blocks]
  [ -extent_type extent_type]
  [ -tp_type tp_type]
  [ -no_verify]
  [ -pace]
  [ -no_count_keys]
  [ -no_verbose]
  [ -no_full_path]
  [ -no_details]
  [ -no_brief]
```
**locate Indexed Files**

**Arguments**

- **directory_names**
  One or more directories to be searched. If you do not specify a directory name, the command searches the root directory of all disks on the current module. The search goes to unlimited depth unless you specify the `-depth` argument.

- **-depth number_of_subdirs**
  Limits the depth of subdirectories that the command searches. You can specify 1 to 25 for `number_of_subdirs`. If you specify 1, the command searches only the directory specified in `directory_names`. If you do not specify this argument, the command searches all subdirectories.

- **-min_key_size key_size**
  Displays only those files with indexes having the specified `key_size` or greater. If you do not specify this argument or if you specify a `key_size` of 0, the command displays all applicable indexes, including those with no intrinsic key size (such as item indexes).

- **-min_level number_of_levels**
  Displays only those files with indexes having a current level of `number_of_levels` or greater. You can specify 1 through 64 for `number_of_levels`. If you do not specify this argument, the command displays files with empty indexes.

- **-min_keys number_of_keys**
  Displays only those files with an index containing `number_of_keys` or greater. The value of `number_of_keys` must be at least 1. If you do not specify this argument, the command displays files with empty indexes.

- **-min_blocks number_of_blocks**
  Displays only those files with an index containing `number_of_blocks` or greater. The value of `number_of_blocks` must be at least 1. If you do not specify this argument, the command displays files with empty indexes.

- **-extent_type**
  Displays indexed files based on extent type. Values are as follows:
  - static—Displays indexed files with static extents.
  - dynamic—Displays indexed files with dynamic extents.
  - either—Displays indexed files with either static or dynamic extents.
  - none—Displays indexed files with no extents.

  If you do not specify this argument, the command displays indexed files with any extent type, including no extents.
-tp_type
Displays indexed files based on whether they are transaction-protected. Values are as follows:

- **tp**—Displays transaction-protected indexed files.
- **non-tp**—Displays non-transaction-protected indexed files.

If you do not specify this argument, the command displays both types of indexed files.

-no_verify
Performs basic verification on all applicable indexes. By default (no), the command does not verify indexes.

Files may be opened and may be transaction protected. The command does not perform verification on temporary files (that is, those whose names start with _), reserved indexes (for example, record_indexes), or item indexes. The command reports empty indexes, but verification is not applicable. Only files containing indexes are located, not queues that may use indexes internally.

If you also specify the -brief argument, the command identifies only those indexed files that fail verification. Otherwise, the command identifies all indexes being verified or not verified.

-pace pace_value
Changes the speed at which the command transverses records during verification.

The -pace argument is meaningful only if you specify it with the -verify argument. Indexes are verified by using them to traverse the file in both directions. The file's records are not accessed. This ensures that the index is self-consistent but does not catch every possible index error. (Use the verify_index command, which is described in OpenVOS System Administration: Starting Up and Shutting Down a Module or System (R282), to perform a more thorough analysis.) This basic validation is designed to be minimally intrusive and be performed when the files in question are active.

By default, the command traverses 1024 records at a time, followed by a short sleep so that the file is available for other users. You can set this value from 0 through 32767. Lower values require much more time to traverse the index. Setting -pace to 0 causes traversal of 32,767 records at a time with no intervening sleeps and can be used when no other critical file activity is occurring.

-no_count_keys
Counts the keys in each applicable index. By default (no), the command does not count keys.

Unlike the count_keys option of the s$get_index_status subroutine, this argument is minimally invasive. However, -count_keys does not guarantee an accurate count. (See the OpenVOS Subroutines manuals for more information about s$get_index_status.)
locate_indexed_files

If you specify the -min_keys argument, the command counts keys only up to the minimum required for determining applicability, whereas -count_keys counts all keys and implies use of the -detail argument.

▸ -no_verbose
   Displays all directories that the command searches, even those with no applicable files. By default (no), the command displays only directories containing applicable files.

The -full_path and -verbose arguments are mutually exclusive.

▸ -no_full_path
   Displays the full path name on each line and eliminates the directory identification and summary, unless you also specify -brief. By default (no), the command does not display the full path name.

This argument may be useful if you use the output to produce command macros.

If you specify both -full_path and -verify, the command always displays a full path name, even if you also specify -brief.

The -full_path and -verbose arguments are mutually exclusive.

▸ -no_details
   Displays information about each index involving type, number of blocks, extent type, transaction protection, current level, and if -count_keys is used, the current number of keys. By default, the command does not display these details about each index.

▸ -no_brief
   Displays only a count of applicable indexed files. If you specify -brief with -verify, the command displays only those files with indexes that failed verification. By default, the command displays all applicable indexed files.

Explanation

The locate_indexed_files command allows you to search for indexed files with various characteristics.

Examples

Example 1.
The following command performs a verification on indexed files and then displays the results.

locate_indexed_files -verify

locate_indexed_files . -verify
%s#Raid4>AcmeCompany>Smith:
   Verifying as_meter_file[meter_index]
   Not verifying reserved index as_meter_file[_deleted_record_index]
   Verifying emacs_keystroke_macros[macro_name]
   Not verifying reserved index

(Continued on next page)
locate_indexed_files

emacs_keystroke_macros[_deleted_record_index]
  Verifying error_codes.text[number]
  Verifying error_codes.text[name]
  Verifying error_codes.text[name_to_number] Total of 3 indexed files examined.
  5 indexes were verified.

%#Raid4>AcmeCompany>Smith>cm-429:
  Verifying rel_file[ix]
  Verifying seq_file[ix]
  Verifying seq_file1[ix]
Total of 3 indexed files examined.
  3 indexes were verified.

Example 2.
The following example shows examination of a large index. The ... line indicates progress. The < > shows the halfway point where the traversal is reversed.

locate_indexed_files #raid8>pf_repro4 -verify -pace 2000 -details

%swls#raid8>pf_repro4>nmatst:
  Not verifying reserved index network_network_id[_deleted_record_index]
  Verifying network_network_id[network_key]
    separate key index Blocks: 590262 High: 590196 TP SAE-65536 (524288)
    Current level: 5
  .......................< >.......................
Total of 1 indexed file examined.
  1 index was verified.

%swls#raid8>pf_repro4>sbt>common:
  Verifying sbt_file[sbt_idx]
    embedded key index Blocks: 4262 High: 4256 TP No extents
    Current level: 3
locate_indexed_files: Internal format error in file. sbt_file[sbt_idx]
  Not verifying reserved index sbt_file[_deleted_record_index] Total of 1 indexed file examined.
  1 index failed verification.

Related Information
See the description of the verify_index command in OpenVOS System Administration: Starting Up and Shutting Down a Module or System (R282).
locate_large_dirs

**Purpose**

This command reports the names of directories that are at or over a specified percentage of their maximum capacity.

**Display Form**

```plaintext
------------------------- locate_large_dirs -------------------------
path_name:               
-warn_at:    75
-check_block_usage: no
```

**Command Line Form**

```
locate_large_dirs [path_name]

[-warn_at number]
[-check_block_usage]
```

**Arguments**

- **path_name**
  The name of a directory to examine. If you specify `path_name`, the command examines all directories that are directly or indirectly subordinate to `path_name`, but it does not examine `path_name` itself. If you omit `path_name`, the command examines all disks on the module.

- **-warn_at number**
  Indicates the percentage of entries used (or blocks used, if you specify `-check_block_usage`) of the maximum allowed. Specify a value between 0 and 100; the default value is 75.

- **-check_block_usage**
  Identifies directories that contain more blocks than the maximum number allowed, rather than entries. If you do not specify this argument, this command identifies only expandable directories, because current entry usage is not tracked for standard directories.
Explanation

The `locate_large_dirs` command reports the names of directories that are at or over a given percentage of their maximum capacity. It takes into account the settable maximum growth limit of expandable directories.

The command enables you to detect the growth potential of a directory by the number of entries it currently contains versus the maximum number of entries allowed. Because only expandable directories track the current number of entries and can have entry limits set, the `locate_large_dirs` command identifies only expandable directories unless you specify `-check_block_usage`.

If you are a privileged user, the command forces status access on the directory. If you are not a privileged user and do not have sufficient access to the directory, the command displays a warning.

Examples

In the following example, the command finds a disk that contains large directories.

```
locate_large_dirs <
*** Scanning for large directories in %s1#d02>Sales
%s1#d02>Sales>Jones>d1b: 31888 (97%) of 32700 entries allowed
*** WARNING: Located large directories in %s1#d02>Sales
```

In the following example, the command identifies the number of blocks used in all of the large directories on the module’s disks.

```
locate_large_dirs -check_block_usage

*** Scanning for large directories in %s1#d02-1

*** Scanning for large directories in %s1#d02-2

*** Scanning for large directories in %s1#d02-03
%s1#d02-03>Jones>d4: 527 (100%) of 527 blocks allowed
%s1#d02-03>Jones>d5: 527 (100%) of 527 blocks allowed
*** WARNING: Located large directories in %s1#d02-03

*** Scanning for large directories in %s1#d04
%s1#d04>Sales>Jones>d1: 527 (527%) of 100 blocks allowed
%s1#d04>Sales>Jones>d3a: 530 (100%) of 530 blocks allowed
%s1#d04>Sales>Jones>d4: 527 (100%) of 527 blocks allowed
*** WARNING: Located large directories in %s1#d04
```

Note: As shown in the following example, the current blocks or entries used in a directory may exceed the maximum value. You can set the maximum value lower than the current value to avoid further growth until the size of the directory is sufficiently reduced. This accounts for usage percentages greater than 100 percent; however, such directories are always diagnosed.

Related Information

See also the description of the `locate_expandable_dirs` command.
locate_large_files

**Purpose**
This command allows you to search for large files.

**Display Form**

```
-------------- locate_large_files --------------
path_name:  
-warn_at:  
-syserr_log: no  
-max_blocks:  
```

**Command-Line Form**

```
locate_large_files [path_name]
  [-warn_at percentage]
  [-syserr_log]
  [-max_blocks blocks]
```

**Arguments**

- **path_name**
The relative path name of the directory to be searched. If you do not specify a path name, the command searches all disks on the current module.

- **-warn_at percentage**
  Specifies the percentage of the maximum size to be used as a warning threshold. The percentage can be between 50 and 100; the default is 75. If you specify 75, for example, the command displays any files or indexes whose last record is greater than or equal to 75% of the maximum size allowed for that type of file.

- **-syserr_log**
  Writes messages about large files to the system error log. By default, these messages are not written to the system error log.

- **-max_blocks blocks**
  A value between 1 and $2^{31}$. This value designates block usage; all files occupying at least this many blocks are identified. This argument and -warn_at are mutually exclusive.
Explanation

In OpenVOS, the four standard file types used for storing and indexing data are sequential, stream, relative, and fixed. The maximum file size is determined by whichever is smaller: the maximum byte offset (for sequential or stream files only) or the maximum number of disk addresses available in the file map (for sequential, stream, relative, or fixed files). The file map is a list of blocks that contain the disk addresses of the actual data or index information for the file.

By default, every disk address in a file map represents one disk block of 4096 bytes. You can use statically-allocated extents (SAE files, or extent files) or dynamically-allocated extents (DAE files) so that the file map can track data blocks for very large files. SAE files allocate and initialize the blocks when you create the file. For DAE files, each file-map address has the address of one extent_size group of blocks. The blocks are allocated and initialized when that extent first gets referenced.

For fixed files, relative files, or 64-bit stream files, data does not have to be added in ascending order from the beginning of the file. Instead, it may be written based on the record number or byte offset, and many lower-numbered blocks may not need to be allocated to disk. Such files are called sparse files. Because the calculation made by the locate_large_files command compares the block number of the highest block in use to the maximum block that can be written to the file, the command, when used with the -warn_at argument, may report sparse files as large files, even if they do not have many disk blocks assigned to them. Use the -max_blocks argument to identify files based on actual block usage, as opposed to the block number containing the last record in the file.

Examples

An example of the locate_large_files command follows.

```
locate_large_files -warn_at 55

*** Scanning for large files in %disk_test#m1
%disk_test#m1>relative.4094.16.1.none.all 100%.
%disk_test#m1>relative.none.4094.16.1 100%.
*** WARNING: Located large file or index in %disk_test#m1

*** Scanning for large files in %disk_test#m1-1

*** Scanning for large files in %disk_test#m1-2

*** Scanning for large files in %disk_test#m1-3

*** Scanning for large files in %disk_test#m1-4

*** Scanning for large files in %disk_test#m1-5
%disk_test#m1-5>prod_cust_data  58% [idx_1_50 63%] [idx2_1_56 77%].
*** WARNING: Located large file or index in %disk_test#m1-5
```
An explanation of the preceding example follows:

- The **WARNING** message indicates that the specified disk or directory contains one or more files that exceeded the percentage specified in the `-warn_at` argument.

- The line `%disk_test#m1-5>prod_cust_data 58% [idx_1_50 63%] [idx2_1_56 77%]` indicates that the data is at 58% of the maximum size for that file type. Two indexes (`idx_1_50` and `idx2_1_56`) are also more than 55% full.

  If the data exceeds *percentage*, that percentage will show up immediately after the file’s path name, before any indexes. The command checks each index. If the index exceeds the limit, it appears after the file’s path name, enclosed in brackets.

  A period at the end of the line indicates that the data and all indexes for that file have been checked and reported on, as necessary.

- The line `%disk_test#m1>relative.4094.16.1.none.all 100%.` indicates that the file cannot grow beyond the last record that was written. If it is a sparse file, the system can write new data in the unwritten lower records. If it is not a sparse file, the system can only rewrite records.
locate_stream_files

Purpose
This command enables you to search a module for stream files with certain characteristics.

Display Form

```
----------------------------- locate_stream_files -----------------------------
directory_names: [ ]
-depth: -type: 64-bit
-brief: no
-long: no
----------------------------- locate_stream_files -----------------------------
```

Command-Line Form

```
locate_stream_files [directory_names]
[ -depth number_of_subdirs ]
[ -type file_type ]
[ -brief ]
[ -long ]
```

Arguments

- `directory_names`
  One or more directories to be searched. If you do not specify a directory name, the command searches the root directory of all disks on the current module.

- `-depth number_of_subdirs`
  Limits the depth of subdirectories that the command searches. If you do not specify this argument, the command searches all subdirectories.

- `-type` [CYCLE]
  Determines which type(s) of stream file to display in the output. Values are as follows:
  - 64-bit (the default) displays only 64-bit stream files.
  - flex displays only 64-bit stream files with flexible extents.
  - large displays only 64-bit stream files larger than 2 GB.
locate_stream_files

- **sparse** displays only sparsely allocated 64-bit stream files.
- **all** displays all stream files.

- **-brief**
  Displays only a count of the number of files in each visited directory that contains the specified type of stream file. You cannot specify this argument with **-long**. If you specify neither argument, the command displays only those directories containing applicable files, with the file names and a non-zero count.

- **-long**
  Displays all directories visited, the names of applicable files in each directory, and a count (possibly zero) of the applicable files in each directory. You cannot specify this argument with **-brief**. If you specify neither argument, the command displays only those directories containing applicable files, with the file names and a non-zero count.

**Explanation**

The `locate_stream_files` command allows you to search for stream files with various characteristics. Such searches can identify certain types of stream files on a module that is running an OpenVOS release that may be incompatible with previous releases. For example, 64-bit stream files are supported on OpenVOS Release 17.2.x but not on earlier releases.

Before you move a disk to a module running an older release, make sure that you first identify any stream files with characteristics that are incompatible with that release, and then convert or remove them.

**Examples**

In the following example, the command searches for sparse files in the root directory of all disks on the current module and then displays the results.

```
locate_stream_files -type sparse
```

Checking directories on disk %s#Raid3...
Checking directories on disk %s#raid0-1...
Checking directories on disk %s#raid0-2...
Checking directories on disk %s#Raid4...
%s#Raid4>Smith:
  bigflex2 (DAE-256/large)
  bignempty (DAE-32/large)
  e1
  e3 (DAE-128/large)
  e4 (DAE-128)
  s4 (DAE-8)
  s5 (DAE-8)
  z1a (DAE-16/large)
Total of 8 sparse stream files.

%s#Raid4>Smith>xstream:
  new1 (DAE-8)
Total of 1 sparse stream file.
```
Checking directories on disk $s#m111_mas...
$s#m111_mas>Sales>Jones:
    big-file (DAE-8/large)
    stm64
    Total of 2 sparse stream files.

$s#m111_mas>Sales>Jones>test:
    sparse_file (DAE-8)
    Total of 1 sparse stream file.

$s#m111_mas>Sales>Smith:
    p4be_vos_18.0.dev.eq.pm (DAE-8)
    xxx.pm (DAE-8)
    xxx2.pm
    zzz1.pm (DAE-8)
    Total of 4 sparse stream files.

Related Information
For information about the types of stream files, see the description of the create_file command.
login

Purpose
This command creates a process for you and gives you access to system resources.

Display Form — Initial Process

Command Line Form — Initial Process

login user_name
  [-privileged]
  [-password password]
  [-change_password]
  [-priority priority]
  [-home_dir directory_name]
  [-module module_name]
  [-subsystem subsystem_name]
Command Line Form — Subprocess or Subsequent Process

```
login  [group_name]

[-privileged]
[-priority priority]
[-password password]
[-module module_name]
```

Arguments

► **user_name**  
Required
A person name or full user name as registered in the system’s registration file. This argument is case-insensitive. You can use an alias instead of your person name. If the value of the `-require_full_person` argument of the `login_admin` command is no, you can use a word of your person name if the word is unique on your system. For example, if you are registered as Tom_Clark and no one else named Clark is registered on your system, you can use Clark for your user name. However, if the value of the `-require_full_person` argument of the `login_admin` command is yes, you must use your person name, full user name, or an alias. In this case, you must use Tom_Clark as your person name.

If you omit the group name, the operating system adds your default group name to form a user name for your process. Any subprocess you create will have this user name.

► **-privileged**  
Makes your process privileged. You must be registered as a privileged user to log in as privileged. By default, your initial process has the default privilege defined for your user name in the system registration file. Also, by default when you log in to a subprocess or subsequent process, the subprocess or subsequent process has the same privilege as the process from which you create it, the parent process. You cannot create a privileged subprocess from a nonprivileged process.

► **-password password**  
Specifies your password. By default, the operating system prompts you for it. In this case, the characters do not appear on your terminal as you type them. If you supply your password as part of the command line form, the characters are displayed. If you supply your password in the display form of the command, the characters are not displayed. When you create a subprocess on the same system, you do not supply a password. When you create a subprocess on a different system, you supply a password.

**Note:** Abbreviations are not expanded.

► **-change_password**  
Changes your password. You can only change your password while logging in. When you first log in, you must supply your old password; the operating system then prompts for your new password. After you give your new password, the operating system prompts you for it a second time for verification. If the two match, the new password replaces the old password on all modules on the system. The new password is not displayed as you type it in. If your two entries of the password do not match, the operating system again prompts you for your new password.
**login**

**Note:** If the new password contains certain punctuation marks that the operating system recognizes as delimiters (!, (, ), ', ;, or &), you may not be able to log back in and give the password using the command line form. The password will be accepted if you wait for the prompt.

- **-priority priority**
  Sets the priority of your login process. You cannot specify a priority greater than your registered maximum priority. By default, the command gives your process your registered default priority.

- **-home_dir directory_name**
  Sets your home directory to `directory_name`. By default, the command sets the home directory of your process to your default home directory. Any subprocess you create acquires the home directory set for your initial login.

- **-module module_name**
  Creates your process on the specified module. The valid forms of `module_name` are `%system#module`, or `#module` to indicate the current system. By default, the command creates a process for you on the module to which your terminal is connected, or, in the case of a subprocess, on the module that is executing the current process.

- **-subsystem subsystem_name**
  Specifies the name of a subsystem supported under the operating system. The command executes the `start_up.cm` corresponding to the specified subsystem. The startup command macro for the specified subsystem must be of the form `subsystem_name_start_up.cm`. By default, the command executes the `start_up.cm` in your home directory.

- **group_name**
  Logs in to a subprocess as a member of a different group than the group for your current process. The possible values are the group names for which you are registered in the system registration file. You can specify `group_name` only when you log in to a subprocess, and not when you log in initially. By default, the command logs you in as a member of your current group.

**Explanation**

The `login` command authenticates and creates an interactive process for you. It allows you to use system resources.

You can issue a `login` command either before you log in or after you are logged in. In the second case, by default, the `login` command creates a subprocess or subsequent process that has exactly the same attributes as the process from which you create it. However, as the second display form shows, you can select only a subset of the arguments when you log in to a subprocess or subsequent process.

At an initial `login`, the default value for the `-privileged` argument is `as_registered`, which means that your process has the privilege defined for your user name in the system registration file. If you try to initially log in as privileged when your user name is registered as non-privileged, the operating system will not log you in. On a subsequent `login`, the default value for the `-privileged` argument is `yes` if you are currently logged in as privileged or `no` if you are not currently logged in as privileged. If you try to subsequently log
in as privileged when your current process is non-privileged, the operating system will not log you in.

Depending on how your system administrator has registered you, an additional message may appear when you perform an initial login. If this occurs, you must type a response before you can log in successfully.

You must be registered in the system registration file of the system to which you are logging in. The operating system validates your use of the system. For example, if you must supply a password, the operating system asks you for it before allowing you to issue commands.

The operating system attaches the command_input, default_input, terminal_output, and default_output ports of your process to your terminal.

Your current directory at initial login is set to your default home directory or to the directory you specify in the -home_dir argument. If your home directory contains a command macro named start_up.cm, the operating system executes it before accepting commands, both when logging in a new process and when creating a subprocess or a subsequent process.

If the user that is logging in is registered to use external authentication, and if they have a non-null external person name, that name is used instead of their OpenVOS user name.

**Examples**

**Example 1.**
The following command logs in a user with the person name Smith.

    login Smith

The group for the user Smith is the default group registered for the user name Smith.

**Example 2.**
To log the user Smith.east in to the system %s1, use this command:

    login Smith.east -module %s1#m2

The module that executes the user’s interactive process is named m2.

**Related Information**

For additional information about changing passwords, see the description of the change_password command as well as the OpenVOS Commands User’s Guide (R089). For information about how to start a noninteractive process, see the description of the start_process command in this manual. For information about how to display the state of your login process, see the description of the list_users command in this manual. See the Introduction to VOS (R001) for information about processes and subprocesses.
logout

Purpose

This command terminates a login process or login subprocess.

Display Form

Command Line Form

logout [-hold]

Arguments

- **-hold**
  Logs out without breaking a telephone connection, when logged in over a telephone line. By default, `logout` breaks the connection. If you are not logged in over a telephone line, the command disregards the argument.

Explanation

The `logout` command terminates your login process.

The operating system closes all files you have opened, unlocks all locks you have locked, detaches all ports you have attached, and detaches all events you have attached.

Specifying `-hold` when you are logged in over a telephone line gives you the following advantages.

- When you log in again, you do not have to wait to be connected to a dial-up line.
- You can log in to your current module again.

Do not specify `-hold` if you are logged in over a telephone line and do not expect to log in again soon. This frees the dial-up line for another user.

When you are using a vterm device and issue the `logout -hold` command, terminal parameters (such as `prompt_message` and `terminal_type`) are retained for the next login. Asynchronous or window terminal devices clear these parameters on logout.
Examples

The following command terminates your login process or subprocess.

    logout

When you log out from a subprocess, the operating system reactivates the interactive process from which you created the subprocess.
Purpose

This command mounts the tape volume on the specified tape device or on the tape drive connected to the specified port.

Display Form

```
---------------------------------- mount_tape ----------------------------------
tape_device_or_port_name:          [ -tape_format:
- volume_id:
- owner_id:
- message:
- access_rights:
- cartridge_no:
- create_volume: no
- reel_retention:
- compression:
- unattended: no
```

Command Line Form

```
mount_tape tape_device_or_port_name
    [ -tape_format tape_format ]
    [ -volume_id volume_id ]
    [ -owner_id owner_id ]
    [ -message message ]
    [ -access_rights access_rights ]
    [ -cartridge_no cartridge_no ]
    [ -create_volume ]
    [ -reel_retention reel_retention ]
    [ -compression ]
    [ -unattended ]
```

Arguments

- **tape_device_or_port_name** Required
  The name of the tape device or the name of the port attached to the tape drive, holding the tape to be mounted.
-tape_format tape_format

Specifies the type of labels of the tape and the tape files. The operating system can process the following types of tapes:

- ANSI-labeled tapes
- IBM OS/VS and MVS labeled tapes
- UNIX-labeled tapes
- unlabeled tapes

The possible values for tape_format are ansi for ANSI-labeled tapes, ibm for IBM OS/VS-labeled tapes, ibm_mvs for tapes used on MVS/RACF systems, unlabeled for unlabeled tapes, and unix for tapes that have UNIX tar, cpio, or cpio format.

The operating system also sets the default translation mode according to the tape format you specify. When you choose the format ansi, or unix, the default translation mode is ascii; when you choose the format ibm or ibm_mvs, the default translation mode is ebcidic; and when you choose the format unlabeled, the default translation mode is binary. You can explicitly set the default translation mode by giving the -translation argument of the set_tape_file_params command. In this case, be careful to choose a translation mode that is consistent with the specified tape format. The -translation argument overrides any implicit default.

-volume_id volume_id

Specifies the volume ID of the tape to be mounted. The tape facility disregards volume_id when the tape is unlabeled, although it is printed as part of the mount request message. By default, the tape facility disregards a volume ID on the tape volume it is mounting.

-owner_id owner_id

Specifies the owner ID of the tape to be mounted. The tape facility disregards owner_id when the tape is unlabeled, although it is printed as part of the mount request message. By default, the tape facility disregards an owner ID on the tape volume it is mounting.

-message message

Specifies a message to send to the operator that gives information about the tape to be mounted. By default, the tape facility uses the default message value previously defined by the set_tape_drive_params command.

-access_rights access_rights

Specifies the access to set on the tape to be mounted. The possible values are read_write and readonly. If you specify read_write, the tape facility checks to make sure that you have put the write ring in the tape reel or, for cartridge tape, that you have set the write-protect mechanism to off. If you specify readonly, the tape facility accepts a tape with or without a write ring, but does not allow you to write to the tape. By default, the tape is mounted with read_write access.

-cartridge_no cartridge_no

Determines which tape is mounted next. This argument has no effect on tape devices for ftServer modules; specify 0 or blank.
mount_tape

- **-create_volume**
  Specifies that the tape does not contain a tape volume; therefore, the tape facility does not check the tape’s volume ID and owner ID. If the tape is to be a labeled tape, the tape facility writes a new volume ID and a new owner ID on the tape. If the tape is to be an unlabeled tape, the tape facility writes two tape marks at the beginning of the tape. By default, if the tape is labeled, the tape facility checks the volume ID and owner ID of the tape volume.

  **Note:** In order to use this argument, you must have write access to the applicable device.

- **-reel_retention reel_retention**
  Specifies whether the tape should remain loaded when the port is detached. The possible values are `dismount` and `retain`. The default value, `dismount`, causes the tape to be unloaded, unless you give the `set_tape_drive_params` command to enable reel retention. Regardless of the value of `-reel_retention`, the `dismount_tape` command dismounts the tape and unloads it, unless you use the `-no_unload` argument of `dismount_tape`.

- **-compression**
  Enables you to select data compression if you have a tape drive that supports data compression. The default value for the `-compression` argument is `no`. If you select the alternative value, `yes`, it remains for the duration of the port attachment (that is, it is not reset with the `dismount_tape` command). This argument has no effect on tape drives for ftServer modules.

  **Note:** The compressibility of data may vary widely.

- **-unattended**
  Causes tape drives with automatic loaders to switch from one tape to the next, without user intervention. If you specify the value `yes` for the `-unattended` argument, the command does not check the `owner_id` and `volume_id` values against the values you specified for the `-volume_id` and `-owner_id` arguments of the `mount_tape` command. This argument has no effect on tape drives for ftServer modules.

  The `-unattended` argument allows you to use the `read_tape` and `write_tape` commands to consecutively process multiple tapes. However, you cannot use `mount_tape` in unattended mode if you also want to create tape volumes using the `-create_volume` argument.

**Explanation**

The `mount_tape` command mounts the tape volume on the specified tape drive or on the tape drive to which the specified port is attached.

For more information about tape format, see the command description for `set_tape_mount_params`.

The tape facility checks that the tape is mounted with the write ring installed according to the access you specify. If the tape volume is labeled and you specify `-volume_id`, then the tape facility checks the volume ID in the volume label. Similarly, if you specify `-owner_id` and the volume is labeled, then the tape facility checks the owner ID.
Before you perform a cross-module mount, make sure that the devices.tin files on the host and destination modules have identical file entries for the tape drive. For example, if you are specifying the mount_tape command on an ftServer module in order to mount a tape volume on a Continuum-series module’s tape drive, the devices.tin file entry on both modules might look like the following:

```
/ =name tape.16
  =module_name m16
  =terminal_type tape_drive_type_n
  =device_type tape
  =slot 4
  =bio_port 1
  =bio_device 6
```

File entries for devices on ftServer modules have a different format from the one shown in the preceding example. See OpenVOS System Administration: Configuring a System (R287) for more information.

You can explicitly attach a port and mount a tape before a tape command is executed, and explicitly dismount the tape and detach the port after the command is executed. Or, you can let the tape commands perform the attachment and mounting.

You should explicitly attach a port with attach_port and detach the port with detach_port when you perform these tasks:

- reserve the tape drive for your use over a period of time
- set the tape parameters with the set_tape_drive_params or set_tape_mount_params command before mounting the tape with mount_tape

You should explicitly mount a tape with mount_tape and dismount a tape with dismount_tape when you perform these tasks:

- create a tape volume
- specify the volume ID of the tape to be mounted
- perform multiple tape operations to a given tape, such as a sequence of write_tape operations or a sequence of different commands

You can let the tape commands attach a port and mount the tape when you perform these tasks:

- issue a single command, such as write_tape, save_object, or dump_disk
- process a combination of read_tape and list_tape commands
- read a tape using read_tape when you do not know the tape’s format

### Explicit Port Attachment and Tape Mounting

If you explicitly attach a port with attach_port, you must explicitly detach the port with detach_port.

If you explicitly mount a tape with mount_tape, you must explicitly dismount it with dismount_tape. The dismount_tape command by default unloads the tape. If you want the command to dismount the tape but not unload it, specify -no_unload.
The `mount_tape` command can implicitly attach a port. If you use `mount_tape` to attach a port implicitly, you must use `dismount_tape` to detach the port implicitly. Implicit port attachment is explained in the next section.

**Implicit Port Attachment and Tape Mounting**

If you have not yet used the `mount_tape` command to mount the tape, some commands implicitly mount the tape; that is, they mount the tape without any action on your part. These commands are `list_tape`, `read_tape`, and `write_tape`.

The commands that implicitly mount a tape can also implicitly attach a port.

If a command implicitly mounts a tape, it implicitly dismounts the tape after execution. The command dismounts the tape by "releasing" it; that is, by rewinding the tape to the beginning (BOT).

**Note:** To avoid overwriting data on your tape by consecutive `write_tape` operations, you should mount the tape explicitly.

To specify that the command implicitly mount the tape, give a device name or a port name for the first argument, `tape_device_or_port_name`. If you give a device name and a port is not already attached, the command implicitly attaches one.

If, however, you previously attached a port with `attach_port`, the tape command uses that port. If you give a port name for the first argument, you must first attach the port with `attach_port`. When you issue the tape command, it implicitly mounts the tape.

When execution is completed, if the command implicitly mounted the tape, it implicitly dismounts the tape. If the command implicitly attached the port, it implicitly detaches the port.

Implicitly mounting a tape changes the actual tape mount parameters to reflect the values used on the tape label, the density of the tape, and the access rights determined by the write-ring or SAFE switch. The tape command does not prompt you as `mount_tape` does. If the tape command attempts implicitly to mount an uncreated tape, the mount fails.

A tape command can implicitly mount only the first tape of a multivolume set of tapes. The operating system prompts you to mount the second and remaining tapes in the set.

**Automatic Tape Mounting**

If you have not yet used the `mount_tape` command to mount the tape, the following user commands automatically mount the tape: `save_object`, `list_save_tape`, and `restore_object`. (The system administration commands `save`, `restore`, and `dump_disk` also use automatic mounting.) These commands mount the tape for you after displaying the `mount_tape` prompts, to which you must respond. The prompts can prevent you from inadvertently overwriting the contents of a save tape by mounting the wrong tape, or can prevent you from using the wrong tape to restore from.

These commands can also implicitly attach and detach a port. Implicitly detaching the port forces the tape to be unloaded.
To have the tapes mounted automatically for one of these three commands, specify a device name or a port name for the first argument, `tape_device_or_port_name`. If you give a device name and a port is not already attached, the command implicitly attaches one. If, however, you previously attached a port with `attach_port`, the command uses that port. If you give a port name for the first argument, you must first attach a port with `attach_port`. After you issue the tape command, the command automatically mounts the tape for you, displaying the `mount_tape` prompts.

When execution is completed, if the command implicitly attached a port, the command implicitly detaches the port, forcing the tape to be unloaded. If, however, you attached a port explicitly, with `attach_port`, the tape stays mounted after execution, at which time you can optionally issue other tape commands. You must then either detach the port by issuing `detach_port`, or dismount the tape by issuing `dismount_tape`. Both commands cause the tape to be unloaded. If you dismount the tape, you must then detach the port with `detach_port`.

**Access Rights**

The items listed below address what happens when the values you specify for the access rights, tape format, volume ID, owner ID, or the default values do not match the values on the tape.

**Mismatched Access Rights**

The tape access rights are read/write or read only. The default is read/write. When reading a tape, you can set the access rights to `readonly` to prevent the accidental overwriting of the tape. When writing a tape, set the access rights to `read_write`. The access rights you specify with the `set_tape_mount_params` or `mount_tape` command should match the setting of the write-protect mechanism on the tape, as shown in Table 2-24. Set the tape write-protect mechanism before you load the tape. On reel tapes the write-protect mechanism is a write-ring. On 1/4-inch cartridge tapes, the write-protect mechanism is a SAFE switch, and on 1/2-inch cartridge tapes, the write-protect mechanism is a FILE PROTECT switch.

<table>
<thead>
<tr>
<th>Write-Protect Mechanism Setting</th>
<th>Tape Type</th>
<th>1/4-Inch Cartridge</th>
<th>1/2-Inch Cartridge</th>
<th>Access Rights</th>
</tr>
</thead>
<tbody>
<tr>
<td>On</td>
<td>Reel</td>
<td>Write-ring out</td>
<td>SAFE switch on</td>
<td>Read only</td>
</tr>
<tr>
<td></td>
<td>1/4-Inch Cartridge</td>
<td>SAFE switch on</td>
<td>FILE PROTECT switch on</td>
<td>Read only</td>
</tr>
<tr>
<td>Off</td>
<td>Reel</td>
<td>Write-ring in</td>
<td>SAFE switch off</td>
<td>Read/write</td>
</tr>
<tr>
<td></td>
<td>1/4-Inch Cartridge</td>
<td>FILE PROTECT switch off</td>
<td>Read/write</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1/2-Inch Cartridge</td>
<td>FILE PROTECT switch off</td>
<td>Read/write</td>
<td></td>
</tr>
</tbody>
</table>

If the access rights are `read_write` (the default), the tape’s write-protect mechanism must be off. If the write-protect mechanism is on and the access rights are `read_write`, the `mount_tape` command displays the message Tape must be mounted with the write ring in. The message is followed by the prompt Okay to mount tape volume?. The
The mount_tape command displays this prompt until you unload the tape and set the write-protect mechanism to off, or until you set the access rights to readonly. You can alter the access rights with alter_parameters, as described below.

Mismatched Tape Formats
If you specify a tape format that does not match the tape format value on the tape label, the mount_tape command displays a message such as the following.

The tape is an ibm labeled tape, not an ansi labeled tape.
Mount tape on %s1#tape.1.0 as specified in the following.
   Access Rights: read_write (ring_in).
   Tape Format: ansi.
   Volume ID: Vol1.
   Owner ID: Smith.
Okay to mount (yes, alter_parameters, create_volume, abort)?

To correct this mismatch, you can alter the tape format by typing alter_parameters, pressing RETURN, and typing the new value.

Mismatched Volume IDs or Owner IDs
If you specify a value for the volume ID or owner ID with the set_mount_tape_params or mount_tape command that does not match the values on the tape label, the mount_tape command displays a message such as the following.

Tape Volume ID (Vol1) is not the volume requested
Mount tape on %s1#tape.1.0 as specified in the following.
   Access Rights: read_write (ring_in).
   Tape Format: ansi.
   Volume ID: Vol1.
   Owner ID: Smith.
Okay to mount (yes, alter_parameters, create_volume, abort)?

You can correct this mismatch by altering the volume ID or owner ID with alter_parameters as shown above, or, if you have loaded the wrong tape, you can now load the right tape, and type yes in response to the prompt.

If you do not specify a volume ID and owner ID, the command searches for the volume ID and owner ID of the tape. The command displays the tape values for the volume ID and owner ID, and asks you whether to use the tape. Respond with yes or no. If yes, the command uses the mounted tape. If no, the command aborts. You must then reissue the mount_tape command if you want to continue. See the example that follows.

Example
In this example, the user attaches the port a_port to the drive #tape.2.0 and issues the mount_tape command. The command uses the default values for the volume ID and owner ID.

   attach_port a_port #tape.2.0
   ready  14:01:47
   mount_tape a_port
The command displays the default values and asks whether it is all right to mount the tape volume. The user responds with yes.

```
Mount tape on %se#tape.2.0 as specified in the following.
Access Rights:  read_write (ring_in).
Tape Format:    ansi.
Owner ID:       Smith.
Okay to mount tape volume? (yes, alter_parameters, create_volume, abort) yes
```

The command then indicates that the volume ID and owner ID on the tape label are different from the default values, and asks whether to use the loaded tape. The user responds with yes.

```
The tape loaded on %se#tape.2.0 is designated by the following.
Access Rights:  read_write (ring_in).
Tape Format:    ansi.
Volume ID:      VOLONE.
Owner ID:       JONES.
Proceed to use the loaded tape? (yes, no) yes
```

At command level, the user displays both the default values and the actual values on the tape label by issuing the `display_tape_params` command for data that has not been compressed. The output follows.

```
*** DEFAULT TAPE PARAMETERS ***

DRIVE:   Compression:    no
Disposition:  rewind
Reel Retention:  dismount on detach
Multivolume Default:  yes
Issue Op Messages:  yes
Message:
```

(Continued on next page)
(Continued)

MOUNT: Volume ID: 
Owner ID: Smith 
Tape Format: ansi 
Tape Density: 1600 
Cartridge No: 0 
Access Rights: read/write 

FILE: File ID: 
File Number: 1 
Expiration Date: 00000 
Translation: ascii 
Open Type: Create Create Create 
File Format: f vb vb u f 
Block Length: 80 4096 4096 32767 80 
Record Length: 80 4092 4092 32767 80 
Blocking Factor: 

*** USER TAPE PARAMETERS ***

DRIVE: Compression: NOT SET 
Disposition: NOT SET 
Reel Retention: NOT SET 
Multivolume Default: NOT SET 
Issue Op Messages: NOT SET 
Message: NOT SET 

MOUNT: Volume ID: NOT SET 
Owner ID: NOT SET 
Tape Format: NOT SET 
Tape Density: NOT SET 
Cartridge No: NOT SET 
Access Rights: NOT SET 

FILE: File ID: NOT SET 
File Number: NOT SET 
Expiration Date: NOT SET 
Translation: NOT SET 
Open Type: Create Create Create 
File Format: N/S N/S N/S N/S N/S 
Block Length: NOT SET NOT SET NOT SET NOT SET NOT SET 
Record Length: NOT SET NOT SET NOT SET NOT SET NOT SET 
Blocking Factor: NOT SET NOT SET NOT SET NOT SET NOT SET 

(Continued on next page)
(Continued)

*** ACTUAL TAPE PARAMETERS ***

DRIVE: Compression: no
Disposition: rewind
Reel Retention: dismount on detach
Multivolume Default: yes
Issue Op Messages: yes
Message:

MOUNT: Volume ID: VOLONE
Owner ID: JONES
Tape Format: ansi
Tape Density: 1600
Cartridge No: 0
Access Rights: read/write

Tape is not open. FILE Parameters are not valid for this state.

Related Information
See also the command descriptions of copy_tape, create_tape_volumes, dismount_tape, display_tape_params, dump_tape, list_save_tape, position_tape, read_tape, restore_object, save_object, set_second_tape, set_tape_drive_params, set_tape_mount_params, set_tape_file_params, verify_save, and write_tape.
move_device_reservation

Purpose
This command moves a device reservation from the current process to one other process.

Display Form

move_device_reservation

device_path_name: [process_name]
-process_name: *
-user: current_user_name
-module: 

Command Line Form

move_device_reservation device_path_name

[process_name]
-user current_user_name
-module module_name

Arguments

► device_path_name Required
The path name of a device, such as a tape drive or a disk, reserved for the current process.

► process_name
The name or star name of a process for which the device is to be reserved. If you use a
star name, move_device_reservation prompts you to select the desired process.
By default, the -user and -module arguments determine the process to which the
device reservation is moved.

► -user current_user_name
Specifies the name or star name of the user for whom the device is to be reserved. By
default, move_device_reservation uses your user name.

► -module module_name
Specifies the module executing the process for which the device is to be reserved. By
default, move_device_reservation uses your current module.

Explanation
The move_device_reservation command moves a device reservation from one process
to another.
move_device_reservation

If you give `process_name`, the `move_device_reservation` command moves the device reservation from the current process to a process matching the `process_name` term. If you specify a `user_name`, the `move_device_reservation` command moves the device reservation from the current process to a process matching the `user_name` term. If you specify both a `process_name` and a `user_name`, the `move_device_reservation` command moves the device reservation from the current process to a process matching the `process_name` and `user_name` terms. Thus, if `process_name` matches more than one process, you can further define it by specifying `user_name` as well. If the process is not running on the current module, specify the `module_name`; `move_device_reservation` moves the device reservation from the current process to a process running on the specified module.

Access Requirements

By default, you have write access to all devices. If your system administrator restricts access to a tape device, you need read access to read from tapes, or write access to read from and write to tapes. To use other devices on which access is restricted, you need only write access.

Related Information

See also the command descriptions of `batch`, `cancel_batch_requests`, `display_batch_status`, `list_batch_requests`, `reserve_device`, and `cancel_device_reservation`. 
move_dir

Purpose

This command moves a directory and its contents from one place to another.

Display Form

```
----------------------------------- move_dir -----------------------------------
source_directory: ...
destination: ...
-pack: no
-delete: no
-keep_dates: no
-brief: no
-pacing: disk_type
```

Command Line Form

```
move_dir [destination]
[-pack]
[-delete]
[-keep_dates]
[-brief]
[-pacing pacing_value]
```

Arguments

- `source_directory` **Required**
  The name or star name of directories to be moved.

- `destination`
  The path name of the moved directory. By default, `move_dir` makes `source_directory` a subdirectory of your current directory.

- `-pack` **[CYCLE]**
  Packs any files being moved, discarding deleted records. You cannot specify `-pack` if any file to be moved has separate-key or item indexes. An error message is returned for each file that cannot be packed, and those files are not moved. By default, `move_dir` does not pack the file.

- `-delete` **[CYCLE]**
  Deletes an existing directory if its path name conflicts with the path name of the moved directory. By default, `move_dir` asks you if you want to delete the directory with the conflicting path name.
move_dir

- **-keep_dates**
  Assigns to the new directory and its contents the creation date, modification date, and last-used date of the directory being moved. The initial last-saved date of all objects is never. By default, the current time of the move is used for the creation date, modification date, and last-used date.

  **Note:** If you have set an expiration date for files in `source_directory`, this command does not preserve the expiration date of the moved files in `destination`, even if you specify `yes` as the value of `-keep_dates`. For more information, see the description of the `set_expiration_date` command.

- **-brief**
  Suppresses the display of each directory name that matches a star name before the directory is moved. By default, `move_dir` displays the name(s).

- **-pacing pacing_value**
  Determines the pacing behavior of the move operation. Possible values are `disk_type` (the default value), `yes`, and `no`. Pacing occurs during the move operation if either of the following is true:

  - If you specify `disk_type` and the source or target disk is optimized for fast response time
  - If you specify `yes` and the source or target disk is not optimized for fast response time

  If you specify `no`, pacing does not occur, regardless of the type of the source or target disk. Only privileged users can specify the `no` value. See the Explanation section for more information about pacing.

**Explanation**

The `move_dir` command moves a directory, including any contained files, subdirectories, links, and access control lists. In effect, this command gives a directory a new path name that designates a new location in the hierarchy. Normally, the object designated by the new name does not exist before the move. If the source directory contains a pipe file, the command moves the file as an empty pipe file.

Specify the name and location of the directory to be moved with the `source_directory` argument. Use the `destination` argument to specify the path name the directory is to have after it is moved. The `destination` argument should not conflict with the name of an existing directory unless you intend to delete the existing destination directory, actually replacing it with the moved directory. If you specify as the `destination` argument the path name of an existing directory, the `move_dir` command asks if you really want to delete the existing directory.

All objects in a directory and in all its subdirectories must be able to be deleted, or the directory cannot be deleted. For example, you cannot delete files that have the safety switch set, nor can you delete hidden files. (See *Using OpenVOS Extended Names* (R631) for information about hidden files.) If it is not possible to delete the existing destination directory, `move_dir` returns an error message and does not move any objects into that directory. However, `move_dir` will delete all objects in the `destination` directory that can be deleted.
move_dir

This command copies all objects in source_directory to the destination directory and after doing so, deletes source_directory. If this directory contains objects that cannot be deleted, move_dir does not delete it. However, move_dir does delete all objects in the directory that can be deleted. Normally, the command returns an error message for each file and subdirectory that it cannot delete.

The move_dir command moves the contents of directories in the following manner unless the current execution environment does not allow the name of the directory being moved or the name of the target directory.

If the directory being moved contains objects with extended names, those objects are copied only if both the module on which move_dir is invoked and the module containing the destination directory supports them.

Depending on the module you are logged in to, different results can occur if objects with extended names are not copied:

- If you are logged in to a module that supports only legacy names, no error is returned.
- If you are logged in to a module that supports version 1 or version 2 extended names, hidden objects are ignored and are not diagnosed individually. Otherwise, an error is reported for all other objects that cannot be copied. If no error has been reported, the operating system returns the error e$all_objects_not_copied(7771).

If you omit the destination argument, source_directory and all of its contents are built as a subdirectory of the current directory. The command uses the directory name portion of the path name specified in the source_directory argument as the name of the moved directory. If a subdirectory with that name already exists in the current directory, for example as the result of a previous use of the move_dir command, move_dir asks if you really want to delete the existing directory.

If you omit destination, and source_directory is an existing subdirectory in your current directory, the operating system tells you that both source_directory and destination name the same object.

If you specify destination, and no directory with the destination name exists, the move_dir command creates a directory with that name.

When you move a directory, it keeps its type and limit attributes. However, if the destination module is running a release that does not support expandable directories, the result is a normal directory without limits.

If the directory being moved (or any of its subdirectories) is expandable or has nonstandard limits, the type and limit information is retained only if both the module on which move_dir is issued and the module containing the destination directory support expandable directories.

If you attempt to move a directory that is expandable or has nonstandard limits to a disk that is either set with restricted expand mode or is on a module running an OpenVOS (or VOS) release that does not support expandable directories, the result is a move into a standard directory (assuming standard capacity is sufficient), with the type and limit information removed, along with a warning message issued for the top-level directory (only) affected. For
move_dir

more information about restricted expand mode, see the description of the
set_dir_expand_mode in OpenVOS System Administration: Disk and Tape
Administration (R284).

If the directory contains more entries than will fit in a normal directory, an error occurs after
the command moves all objects that will fit. The source_directory is not deleted. Since
the maximum number of entries in a normal directory is based on blocks used, the number of
entries that will fit depends on various factors, including the order in which the objects are
moved. Before you move an expandable directory or a directory containing expandable
directories to a module that does not support them, first convert them to normal directories to
ensure that move_dir does not result in an error. To identify such directories, use the
locate_expandable_dirs command. Use the set_dir_type or consolidate_dir
(with the -revert argument) to convert them to normal if it is possible to do so.

The value of source_directory and destination can be a star name. See the
copy_file command for a description of how star names function.

If you specify a link for source_directory and specify a star name for destination,
be careful that the link does not point to a directory that is identified by the star name, or the
command may determine that source_directory and destination are identical. For
example, if you have a subdirectory called subd, a link called xxx to subd>yyy, and two
subdirectories in subd called xxx and yyy, the move_dir command behaves as follows
when you specify source_directory as a link and destination as a star name.

    move_dir xxx subd>*
    Moving xxx to %sys#m1>Sales>Joe_Smith>subd>xxx.
    Do you really wish to delete %sys#m1>Sales>Joe_Smith>subd>xxx?
        (yes, no) n

The command does not expand the link and uses the unexpanded link name to resolve the
star name to subd>xxx. The command behaves in a similar fashion if you specify
source_directory as a link and destination as a directory name.

    move_dir xxx subd
    move_dir: Either source or destination directory is a subdirectory
        of the other. %sys#m1>Sales>Joe_Smith>subd.

If you specify -pack, all indexes are re-created regardless of file organization. In some cases,
the resulting indexes are empty. You cannot delete a record from a fixed file with no record
index. If you ask the operating system to delete a record from such a file, it updates
embedded-key and deleted-record indexes appropriately, but does not actually delete the
record. Therefore, such records reappear if their file is packed.

If you specify -delete, the move_dir command deletes a directory whose path name
matches the path name of the moved directory. If destination is a link and you specify
-delete, the operating system replaces the target of the link with the moved directory.

Links to targets in the directory being moved become links to the relative path names of those
targets in the new directory. Links to targets outside the directory being moved are copied
exactly.
move_dir

Pacing prevents the move operation from dominating the disks, and it allows other processes to access other files on the disks involved (both source and target) without long delays. Pacing is relevant only to block-mode moves; the value of the -pacing argument is ignored for record-mode moves (that is, those for which the -truncate or -pack argument has been specified).

The move_dir command assigns ownership of any moved files to the user name of the person doing the copying.

Access Requirements
To move a directory, you need modify access to the directory source_directory, to the directory containing it, and to the directory that contains the moved directory.

Examples
Example 1.
Suppose that this is the current directory.

%s1#d02>Sales>Jones

The following command moves the directory %s1#d02>Sales>Clark>orders into the current directory.

move_dir >Sales>Clark>orders clarks_orders

The moved directory has the following path name.

%s1#d02>Sales>Jones>clarks_orders

Example 2.
Again, assume that this is the current directory.

%s1#d02>Sales>Jones

The following command moves all of the subdirectories of >Sales>Clark>orders into the current directory hierarchy.

move_dir >Sales>Clark>orders>*

If you specify a star name, the command displays the names of the objects being moved, as shown by the following example.

Moving %s1#d02>Sales>Clark>orders>* to *
    new_accounts
    new_orders
    closed_accounts

The directory %s1#d02>Sales>Jones now has three new subdirectories: new_accounts, new_orders, and closed_accounts.
Related Information

For more information about directory management, see the command descriptions of change_current_dir, compare_dirs, consolidate_dir, copy_dir, create_dir, delete_dir, display_current_dir, give_default_access, locate_expansible_dirs, remove_access, and set_dir_type.
move_file

Purpose

This command moves a file or set of files to another file or directory.

Display Form

```
---------------------------------- move_file -----------------------------------
source_file: ____________________________
destination: ____________________________
-pack: no
-truncate: no
-delete: no
-keep_dates: no
-keep_acl: no
-brief: no
-keep_audit: no
-pacing: disk_type
```

Command Line Form

```
move_file source_file
    [destination]
    [-pack]
    [-truncate]
    [-delete]
    [-keep_dates]
    [-keep_acl]
    [-brief]
    [-keep_audit]
    [-pacing pacing_value]
```

Arguments

- `source_file` Required
  The name or star name of the files to be moved.

- `destination`
  Either a file path name, or an existing directory into which `move_file` puts the moved files. A file name can be a star name. If you specify a file name, the command moves the file into the current directory and names it `destination`. If you specify an existing directory, the command moves the files into that directory. By default, `move_file` moves the files into the current directory without changing their names, as long as the current directory is not the source directory.
• **-pack**
  Packs a file being moved, discarding deleted records. You cannot specify `-pack` if the
  file to be moved has separate-key or item indexes. An error message is returned for each
  file that cannot be packed, and those files are not moved. By default, `move_file` does
  not pack the file.

• **-truncate**
  Truncates an existing destination file before moving an input file to it. If you
  specify `-truncate` and the destination file is an existing file, `move_file`
  preserves the file’s attributes and indexes but deletes its contents (data records). Any
  embedded key indexes previously defined on the destination are rebuilt from the new
  records. However, if the destination file does not exist, the command creates an
  output file with the same organization, maximum record length, and allocation size as
  the file to be moved but creates no indexes on the new file. If you specify `-truncate`,
  the file is packed regardless of the value of the `-pack` argument.

• **-delete**
  Deletes a file if it has the same path name as the destination path name of the moved
  file. By default, `move_file` asks if you want to delete a file that has a conflicting path
  name. By default, `-delete` determines what happens if there is a name conflict.

• **-keep_dates**
  Assigns to the new file the creation date, modification date, and last-used date of the
  file being moved. The initial last-saved date of all objects is `never`. By default, the
  current time of the move is used for the creation date, modification date, and last-used
  date.

  **Note:** If you have set an expiration date for `source_file`, this command
  preserves the expiration date of the moved files in the destination file, even
  if you specify `no` as the value of `-keep_dates`. For more information, see the
  description of the `set_expiration_date` command.

• **-keep_acl**
  Keeps the access control list, but not the default access control list, with the file. By
  default, the access control list is not moved with the file.

• **-brief**
  Suppresses the display of each file name that matches a star name before the file is
  moved. By default, `move_file` displays the name(s).

• **-keep_audit**
  Specifies that the new file retains the audit options of the source file. By default, the
  command does not retain the audit options of the source file.
- **-pacing pacing_value**
  Determines the pacing behavior of the move operation. Possible values are `disk_type` (the default value), `yes`, and `no`. Pacing occurs during the move operation if either of the following is true:
  
  - If you specify `disk_type` and the source or target disk is optimized for fast response time
  
  - If you specify `yes`

  If you specify `no`, pacing does not occur, regardless of the type of the source or target disk. Only privileged users can specify the `no` value. See the Explanation section for more information about pacing.

**Explanation**

The `move_file` command moves a file or files. If the source file is a pipe file not open by another process, the command moves the file as an empty pipe file.

Use the `source_file` argument to specify the file or files to be moved.

With the optional `destination` argument, you can specify either a directory into which the operating system is to put the moved file or files, or a file path name.

The value of `source_file` and `destination` can be a star name. See the `copy_file` command for a description of how star names function.

If you give a directory name as the `destination` argument, the `move_file` command moves `source_file` into that directory using its same file name. If you specify a star name for `source_file`, all files whose names match the `source_file` argument move into the destination directory retaining their file names.

If you give a file path name as the `destination` argument, and it is not a star name, then `source_file` must match only one file path name. In this case, the `move_file` command moves the file and names it as specified in `destination`. If the `destination` and `source_file` arguments are both star names, the command moves, and renames as appropriate, each star name pair.
If you omit the destination argument, the move_file command moves all of the files whose names match source_file into your current directory using their same file names. If the files to be moved already reside in your current directory, the operating system displays this message:

Both the source and destination name the same object.

If you specify a link for source_file and specify a star name for destination, be careful that the link does not point to a file that is identified by the star name, or the command may determine that source_file and destination are identical. For example, if you have a subdirectory called subd, a link called xxx to subd>yyy, and two files in subd called xxx and yyy, the move_file command behaves as follows when you specify source_file as a link and destination as a star name.

```
move_file xxx subd>*
Moving %sys#m1>Sales>Joe_Smith>xxx to
 %sys#m1>Sales>Joe_Smith>subd>xxx.
%sys#m1>Sales>Joe_Smith>subd>xxx already exists.
 Delete the old one? (yes, no) n
```

The command does not expand the link and uses the unexpanded link name to resolve the star name to subd>xxx. The command behaves in a similar fashion if you specify source_file as a link and destination as a directory name.

```
move_file xxx subd
%sys#m1>Sales>Joe_Smith>subd>xxx already exists.
 Delete the old one? (yes, no) n
```

When you specify -pack, all indexes are re-created, regardless of file organization. In some cases, the resulting indexes are empty. You cannot delete a record from a fixed file with no record index. If you ask the operating system to delete a record from such a file, it updates embedded-key and deleted-record indexes appropriately, but does not actually delete the record. Therefore, such records can reappear if their file is packed. To prevent this, move_file ignores records that consist entirely of hexadecimal FF when packing a fixed file without a record index.

The system displays warning messages if indexes are not moved or rebuilt. If the -truncate argument is specified and the target file exists, move_file checks the indexes and displays a message for each embedded index that is not on the target file as well as for each non-embedded index. If -truncate is specified and the target file does not exist, move_file displays a message for each index on the source file. If -no_truncate and -pack are specified, move_file displays a message for each embedded index that is not in the target file, and for each non-embedded index.

**Note:** The display_file_status command shows index names in order of the index address inside the file. This order may change if the file is specified as the subject of the copy_file, move_file, restore_object, or save_object command.

The -truncate and -pack arguments allow you to convert a regular sequential file to an extended sequential file. Similarly, if an extended sequential file is less than approximately...
2 GB in size, you can use `-truncate` and `-pack` to convert it to a regular sequential file. Extended sequential files can exist only on systems running VOS Release 15.1.0 or later.

- Assume the following scenario: your module does not support extended sequential files because it is running a release prior to VOS Release 15.1.0, and `source_file` is a regular sequential file. In this scenario, to move `source_file` to a destination file that is an extended sequential file on a module running VOS Release 15.1.0 or later, specify the `-truncate` argument. This operation truncates the existing destination file, then moves the data from `source_file` into it while maintaining the destination file’s extended sequential format.

- Assume the following scenario: your module is running VOS Release 15.1.0 or later, and `source_file` is an extended sequential file. In this scenario, to move `source_file` to a destination file that is a regular sequential file on a module running a release prior to VOS Release 15.1.0, specify the `-pack` argument. If `source_file`’s size is small enough to fit in the destination file’s regular sequential file format (that is, it is less than 2.14 gigabytes), this operation moves the data from `source_file` into the destination file while maintaining the destination file’s regular sequential format. If no destination file exists, `move_file` creates one, provided you specify `-pack`.

**Note:** In either of the preceding situations, you must issue the `move_file` command from a module running VOS Release 15.1.0 or later.

If a non-empty RAM file is moved, the newly created file does not have RAM file usage. Therefore, if you move a RAM file while it is activated in order to capture its contents, the contents appear in the new non-RAM file.

However, if this is the case, the source file cannot be deleted, since the only way a RAM file can be non-empty is for it to be currently open. When a RAM file is deactivated, its contents are discarded.

When a server queue has RAM usage, the new server queue retains RAM usage and is thus always empty. The contents of a server queue are never copied.

If a RAM file’s containing directory is being copied or moved, the RAM file in the newly created directory retains its RAM usage and is always empty. See the description of `set_ram_file` for more information about RAM files.

**Pacing** prevents the move operation from dominating the disks, and it allows other processes to access other files on the disks involved (both source and target) without long delays. Pacing is relevant only to block-mode moves. The value of the `-pacing` argument is ignored for record-mode moves (that is, those for which the `-truncate` or `-pack` argument has been specified) and also when a file is moved from one directory to another on the same disk.

**Access Requirements**

To move a file, you need write access to the destination file and modify access to both the old and new directories that contain the file.
Examples

To move all of the OpenVOS COBOL source modules in the current directory into the directory >east>Clark, use this command.

    move_file *.cobol >east>Clark

The object names of the copies are the same as the names of the original files.

Related Information

See also the command descriptions of `compare_files`, `copy_dir`, `copy_file`, `create_file`, `cvt_stream_to_fixed`, `delete_file`, `display_file_status`, `locate_files`, `move_dir`, `set_file_allocation`, `set_ram_file`, and `truncate_file`.

```
Purpose

This command calls the multi-process debugger to debug one or more processes in the current network.

Display Form

```
mp_debug
```

No arguments required. Press ENTER to continue.

Command Line Form

```
mp_debug
```

Explanation

The `mp_debug` command calls the multi-process debugger to debug one or more processes. The set of processes that are debugged in a multi-process debugging session is called the *debug process set*; it can include processes from anywhere in your network. This command is particularly useful when you want to debug a process that does not usually have a terminal associated with it, such as a server process.

**Note:** To include a process that does not have your user name, you must be registered as privileged.

You must call the multi-process debugger from a terminal; you cannot invoke the `mp_debug` command from a batch process. However, you can debug a batch process with the multi-process debugger.

When you invoke the `mp_debug` command, you intercept the debugging input and output requests for other processes. Initially, there are no processes in the debug process set. You add and remove processes from the debug process set with the `mp_debug` requests. The processes that you include are called *slave processes* because they are under the control of the multi-process debugger. Using the `mp_debug` requests, you can list these processes, suspend them, and restart them. You can also create a new process, call the debugger for it, and then start the process, all from within the multi-process debugger.

During a multi-process debugging session, a slave process can encounter a break point. When this happens, the process that encounters the break point sends a message to the multi-process debugger. In response, the multi-process debugger suspends all processes in the debug process set that are not already suspended. If more than one process encounters a break point
before all processes are suspended, each sends a message. The multi-process debugger selects
one of these processes to become active and respond to input from your terminal.

Usually, you issue the OpenVOS mp_debug command after a command line like this.

    a -args  b -args  c -args &

The & character causes the mp_debug program to run as a background process, which
effectively frees your terminal from these processes.

Multi-Process Debugger Requests

The mp_debug command is a request-loop command that has several internal requests. Once
you have invoked the command, the multi-process debugger prompts you for requests as
follows:

    mp_debug:

If you specify an additional prompt message for your terminal, the prompt consists of both
the standard prompt and your own prompt. For example, assume that the prompt message for
your terminal is the > character. In this case, the prompt message is as follows:

    mp_debug:  >

The multi-process debugger allows you to enter any request except quit in either the
command line form or display form. To display the form, press the DISPLAY_FORM key. The
quit request does not have a display form.

The mp_debug command replaces first abbreviation directives in your multi-process
debugger requests. Abbreviations for multi-process debugger requests and within the source
file path name are both expanded.

While your process is in the debugger, you can issue internal commands as if you were at
command level. Use the ..help command to display a list of these commands. To issue an
internal command from debugger request level, type the name of the command preceded by
two periods. For example, ..list invokes the list command. The set of internal
commands may change in subsequent releases of the operating system. Note that you can use
abbreviations for internal commands, for example, ..l. Lines starting with .. are executed
as OpenVOS internal commands in the mp_debug process.

Lines starting with >> are passed to the active slave process to be executed as OpenVOS
internal commands. For example, if you specify >>list_port_attachments, the
operating system displays the port attachments for the process currently being debugged.

If you enter a request that the multi-process debugger does not recognize as an mp_debug
request, the request is forwarded to the current active process as a debug request line.
mp_debug

The mp_debug requests follow.

- `include_process [process_name] [-user user_name] [-module module_name]
  Includes all processes specified by process_name in the debug process set. The
  process_name term can be a star name. Each process you include is assigned a debug
  process number by the multi-process debugger; use these numbers when issuing some
  of the requests that follow. If you specify an asterisk for process_name, the request
  includes all processes of the specified user on the specified module. If you do not use
  the -user or -module terms, the value of user_name is the current user, and the value
  of module_name is the current module.

  If you specify a process that is executing one or more program modules, the process is
  included once for each program module.

  To enter a user name that is different from yours, you must be registered as privileged.

- `exclude_process {process_number} [-all]
  Removes a process or processes from the debug process set. If you specify a process
  number, you must specify the debug process number of the process that you want to
  remove. If you specify -all, the request removes all processes in the set. You cannot
  specify both a process number and -all.

- `list_processes
  Lists all processes currently in the debug process set. The output includes the following
  information for each process in the debug set.
  • the mp_debug process number
  • the state of the process. Possible states are running (R), active (*), suspended at
    debug request level (S), and dead (D)
  • the user name of the process
  • the name of the process
  • the name of a program module being executed in the process, if there is one
  • the module name of the module running the process. If all slave processes are on
    the same module, the module name does not appear.

- `mp_login
  Creates and displays a new login process. You must exit that new login process before you
  can continue with other mp_debug requests. The mp_login request displays a new login
  process only when activated on window-terminal devices. If you issue mp_login on a
  nonwindow-terminal device, the request behaves the same way as the ..login command.
Note that if you issue `.login` from within `mp_debug` when using a window-terminal device, `mp_debug` displays the following message.

Using `.login` from within `mp_debug` starts a new login process but does not display it by default. To access this login you have to use the window term CYCLE command to access the new login. To avoid this situation use `mp_login`.

- **use_process process_number**
  Activates the specified process. Enter the debug process number that was assigned by the `start_process` or `include_process` request. If you are unsure of the process number, you can use the `list_processes` request to display it.

- **suspend_process process_number**
  Suspends the activity of the specified process and calls the debugger for the suspended process. This request has the same effect as breaking an OpenVOS process and selecting `debug`. Enter the debug process number that was assigned by the `start_process` or `include_process` request. If you are unsure of the process number, you can use the `list_processes` request to display it.

- **restart [process_number]**
  Restarts a process or processes. If you specify a process number, specify the debug process number of the process that you want to restart. By default, `restart` restarts all suspended processes.

  You can restart a process in a multi-process debugging session by issuing a `continue` command. In this case, only the active process is restarted. However, using the `continue` command in this way is not recommended, since the multi-process debugger will pass the command directly along to the active process without completely updating the command’s state.

- **start_process command_line [-process_name process_name]**
  
  
  [ -output_path output_path_name] [-priority priority]

  [-privileged] [-module module_name] [-current_dir path_name]

  [-wait_time seconds]

  Creates and starts a process. There are two differences between the multi-process debug `start_process` request and the OpenVOS `start_process` command.

  - The `mp_debug` request adds a `debug` command at the beginning of the command line. It calls the debugger before the new process begins to run.

  - With `mp_debug`, you cannot specify `-cpu_limit` as you can when you invoke `start_process` as a command.

  See the description of the `start_process` command for more information on the options that are available with the `start_process` request.
mp_debug

* stop
Removes all processes from the debug process set. This request is equivalent to an
exclude_process -all request.

* quit
Removes all processes from the debug process set, then terminates the multi-process
debugging session and returns you to command level. The quit request has no display
form.

Note: Do not exit mp_debug using the <Control>-c keystroke. This keystroke may
not terminate all slave processes.

* help [-match string]
Lists mp_debug requests. If you specify -match string and specify a character
string, the help request lists the mp_debug requests whose names contain the string.
By default, help lists all mp_debug requests.

Examples
The following example illustrates a multi-process debugging session. In this session, the user
invokes the mp_debug command to observe the overseer process on another module.

mp_debug
mp_debug:  include_process process1 -user Smith.East -module m10
mp_debug:  include_process process1 -user Smith.East -module m21
mp_debug:  list_processes
  1  R     Smith.East process1 (m10)
  2  R     Smith.East process1 (m21)
mp_debug:  use_process 2
mp_debug:  suspend_process 2
  2: Entering debug.
  2: New language is machine.
mp_debug:  list_processes
  1  R     Smith.East process1 (m10)
  2  S     Smith.East process1 (m21)
mp_debug:  trace
  2: # 5:  00FD60D0x s$read_raw (00256BDEx)
  2: # 4:  00FD6098x get_input_char (line 679 in
           module emacs_key_control)
  2: # 3:  00FD6B1Cx get_input_seq (line 541 in
           module emacs_key_control)
  2: # 2:  00FD6F30x emacs (line 874 in module emacs)
mp_debug:  restart
mp_debug:  quit

  ready 12:13:00  3.332  36

Related Information
See also the descriptions of the debug and start_process commands.
**nls_edit_form**

**Purpose**

This command invokes the Forms Editor.

**Display Form**

```plaintext
----------------------------- nls_edit_form-----------------------------
input_path:                
form_path:                
-into:                    no
-prefix:                  no
-library:                 accept_field_definitions
-edit:                    yes
-backup:                  yes
-force_write:             no
-basic:                   no
-cobol:                   no
-fortran:                 no
-pascal:                  no
-pl1:                     no
-pl1_template:            yes
-c:                       no
-processor:               default
-mapping_rules:           default
-sort_into_by_alignment:  yes
-flag_word_size:          32
-produce_symtab:          yes
```
Command Line Form

```
nls_edit_form input_path
  [form_path]
  [-into]
  [-prefix]
  [-library field_definitions_directory_name]
  [-no_edit]
  [-no_backup]
  [-force_write]
  [-basic]
  [-cobol]
  [-fortran]
  [-pascal]
  [-pl1]
  [-no_pl1_template]
  [-c]
  [-processor processor_string]
  [-mapping_rules mapping_string]
  [-no_sort_into_by_alignment]
  [-flag_word_size number]
  [-no_produce_symtab]
```

Arguments

- **input_path**  
  The path name of an input form definition file. The path name cannot be an extended name. A `.form` file is a programming source file. If the file exists, its name must have the suffix `.form`. You can omit the suffix when specifying the name in the command. If the file does not exist, the Forms Editor behaves as though the file exists but is empty. You cannot expect to edit or compile a `.form` file on a release that is earlier than the release on which the `.form` file was originally created unless the `.form` file contains only features supported by the earlier release.

- **form_path**  
  The path name of the file to which the edited form definition is to be written. If `form_path` does not have the suffix `.form`, the command adds that suffix. By default, `nls_edit_form` writes the form definition to a file in the current directory with the same name as the file specified in the `input_path` argument. If the specified file does not exist when you write out the form, the Forms Editor creates it. The `form_path` cannot be an extended name.

The form being edited is given the simple name of the output form definition file without the suffix `.form`. A form name should not exceed 15 characters; otherwise, the names of some automatically generated include files may exceed 32 characters and be truncated.

**Note:** Do not give a form the same name as the program that displays it. Both a form and its related program require that object modules have unique names.
- **-into**  
  Creates a field-values file for each programming language specified by the language arguments. The Forms Editor assigns the name `form_name.incl.language` to the field-values file (an include file) and places the file in the current directory. You can override this argument with the Forms Editor [MENU]-S request.

- **-prefix**  
  Adds a prefix to each field-identifier name in any field-IDs file that the Forms Editor generates. The default prefix is the name of the form followed by an underline. You can specify a different prefix with the Forms Editor [MENU]-S request. You can also override this argument with the Forms Editor [MENU]-S request.

  If you specify -prefix, the Forms Editor also adds the prefix to each variable name in any OpenVOS BASIC or OpenVOS FORTRAN field-values file that it generates. The field-values files for other languages are not affected.

- **-library field definitions directory name**  
  Specifies a directory for storing and retrieving field definition files. The Forms Editor searches the directory for field definition files when you issue the [MENU]-R request, and writes field definition files to the directory when you issue the [MENU]-E request. If you specify -library but do not specify a name, the command uses a subdirectory of your current directory named `accept_field_definitions`. If the directory you specify, either directly or by default, does not exist when you issue a [MENU]-E request, the Forms Editor creates that directory.

- **-no_edit**  
  Creates new language include files and a new object module from an existing form definition file without editing the form. If you also specify -force_write, the Forms Editor also writes a new form definition file. By choosing -no_edit, you can run the Forms Editor in either a batch process or a started process. By default, the Forms Editor reads the form definition file, displays a representation of the form, and allows you to edit it.

- **-no_backup**  
  Specifies that no backup file is to be created for the `input_path` file. By default, if the `input_path` and `form_path` files are in the same directory, the Forms Editor renames the old file to `input_path.form.backup`. The backup file is created each time you write out the form using the [MENU]-W request; it replaces a previous backup file of the same name, if one exists.

- **-force_write**  
  Writes a new form definition file (`form_name.form`) when you invoke `nls_edit_form` with -no_edit. By default, -no_edit produces the object module and specified include files only. Use -force_write with -no_edit to generate a .backup form file or to rename your form without re-editing it.

  **Note:** Do not specify the rename command to rename a form; object and include files must be renamed, and prefixes in include files must be reassigned.
nls_edit_form

- **-basic**
  Creates OpenVOS BASIC versions of the field-IDs file and the field-values file. By default, the Forms Editor does not create OpenVOS BASIC versions of the files. When revising a form, the command uses the already-specified language. You can override the existing value with the Forms Editor [MENU]-S request.

- **-cobol**
  Creates OpenVOS COBOL versions of the field-IDs file and the field-values file. By default, the Forms Editor does not create OpenVOS COBOL versions of the files. When revising a form, the command uses the already-specified language. You can override the existing value with the Forms Editor [MENU]-S request.

- **-fortran**
  Creates OpenVOS FORTRAN versions of the field-IDs file and the field-values file. By default, the Forms Editor does not create OpenVOS FORTRAN versions of the files. When revising a form, the command uses the already-specified language. You can override the existing value with the Forms Editor [MENU]-S request.

- **-pascal**
  Creates OpenVOS Pascal versions of the field-IDs file and the field-values file. By default, the Forms Editor does not create OpenVOS Pascal versions of the files. When revising a form, the command uses the already-specified language. You can override the existing value with the Forms Editor [MENU]-S request.

- **-pl1**
  Creates OpenVOS PL/I versions of the field-IDs file and the field-values file. By default, the Forms Editor does not create OpenVOS PL/I versions of the files. When revising a form, the command uses the already-specified language. You can override the existing value with the Forms Editor [MENU]-S request.

- **-no_pl1_template**
  Specifies that OpenVOS PL/I include files are to be generated as based structures. (See the Explanation section of this command description for details.) When revising a form, the command uses the already-specified language. You can override this argument with the Forms Editor [MENU]-S request.

- **-c**
  Creates OpenVOS C versions of the field-IDs file and the field-values file. By default, the Forms Editor does not create OpenVOS C versions of the files. When revising a form, the command uses the already-specified language. You can override the existing value with the Forms Editor [MENU]-S request.

- **-processor processor_string**
  Specifies the processor for which object code is to be generated. The values of processor_string are as follows:

  - default
  - pentium4

  If you are creating a form that is to run on a module using an IA-32 processor, specify the pentium4 value. By default, processor_string is the processor type of the current module. The default value is default.
- `mapping_rules mapping_string`
  Specifies one of the following data-alignment rules for include files generated by `nls_edit_form`.
  
  - default
  - default/check
  - shortmap
  - shortmap/check
  - longmap
  - longmap/check

  The default value indicates the system-wide default. The default alignment method is site-settable. By default, the data alignment rules specified by `default` are used. (See the Explanation section of this command description for details.) When revising a form, the command uses the already-specified language. You can override this argument with the Forms Editor `MENU`-S request.

- `no_sort_into_by_alignment`
  Specifies that the alignment of data items in the field-values file overrides all other sorting criteria. By default, this argument is `yes`. (See the Explanation section of this command description for details.) When revising a form, the command uses the already-specified language. You can override this argument with the Forms Editor `MENU`-S request.

- `flag_word_size number`
  Specifies the size of the bit flag words in the field-values file. The values of `number` are 8, 16, and 32. By default, `number is 32`. When revising a form, the command uses the already-specified language. You can override this argument with the Forms Editor `MENU`-S request.

- `no_produce_symtab`
  Produces a form object module without a run-time symbol table. Because the form’s run-time symbol table is small, specify `no_produce_symtab` only if there is a shortage of virtual memory.

**Explanation**

The `nls_edit_form` command invokes the Forms Editor. After you issue `nls_edit_form`, your process is at the editor request level. At editor request level, you can make a number of requests. One request, to quit the Forms Editor, returns your process to command level. To issue the `quit` request, press the `[MENU]` key and then type the letter q or Q.

**Note:** The `nls_edit_form` command invokes the new Forms Editor, and the `edit_form` command invokes the old Forms Editor. See the OpenVOS Forms Management System manuals for more information.

If you specify the path name of an existing input form definition file when you issue the `nls_edit_form` command, the Forms Editor reads the file and displays a representation of the defined form. If you are migrating a Forms-based application from a release earlier than VOS Release 15.0.0, see the manual *Migrating VOS Applications from Continuum Systems* (R607) for more information.
The Forms Editor trims trailing spaces from all values you enter into the editor's request forms and from all lines you enter into the form you are constructing. It also deletes all blank lines from the bottom of the form. When you write out the form definition file and the other files described earlier, the files reflect these deletions.

If you specify -into when you invoke the Forms Editor, but do not specify any of the languages at that time, you can specify languages using the Forms Editor requests. If you do not specify any of the language arguments in the command line or in the Forms Editor, -into is ignored.

If you specify -into, -prefix, -mapping_rules, -sort_into_by_alignment, -flag_word_size or any of the language arguments (-basic, -cobol, -fortran, -pascal, -pl1, -pl1_template, or -c) for a particular form, these arguments are saved in the form definition file. You do not have to re-specify these arguments when using the Forms Editor requests or in future invocations of the Forms Editor on that form.

If you specify -pl1_template, all OpenVOS PL/I include files are generated as based structures. Field-values structures can then be declared using the PL/I like attribute. You should specify -pl1_template or the Forms Editor <MENU>-S request to prevent possible longmap or shortmap mismatches in the generated include file.

The -mapping_rules argument allows you to specify the data-alignment rules for include files generated by nls_edit_form. The value default indicates the system-wide default. The default is site-settable. The value shortmap specifies that the shortmap alignment rules are to be used for the include files. The value longmap specifies that the longmap alignment rules are to be used for the include files. The values default/check, shortmap/check, and longmap/check are equivalent to default, shortmap, and longmap, respectively.

Note that -mapping_rules affects include files differently, based on the language specified. In OpenVOS C and OpenVOS Pascal, when you specify -mapping_rules, the include files will contain explicit longmap or shortmap keywords in field-values structures. In OpenVOS PL/I, these keywords are generated only if you specify the -pl1_template argument or the Forms Editor <MENU>-S request. In PL/I, if you do not specify -pl1_template or the Forms Editor <MENU>-S request, you must add the longmap and shortmap keywords manually.

OpenVOS COBOL, BASIC, and FORTRAN do not allow the definitions of structure templates. For these three languages, the Forms Editor generates an include file for the body of the structure, which you include in your program within the structure definition itself. To do this in OpenVOS COBOL, for example, you would specify the following:

```
01   structure1.
    copy 'struct_info.incl.cobol'.
```

The -sort_into_by_alignment argument determines whether the alignment of the data items in the field-values file overrides all other sorting criteria. The default is yes. This argument is used to avoid compiler warnings if you compile with the -mapping_rules default/check, shortmap/check, or longmap/check argument. This argument also minimizes the size of the field-values file because no padding will be required.

The -flag_word_size argument specifies the size of the bit flag words in the field-values file. (The Forms Editor organizes the OpenVOS Pascal boolean data type, the PL/I bit(1) data type, and the C bit data type into bit flag words.) The values of number are 32, 16, and
8. The default is 32. Bit array fields that are larger than the flag word size are not allowed in any language. If you are using FMS with an OpenVOS C program, you should specify the value 32.

Access Requirements
You need read access to a form definition file in order to read it. You need write access to a form definition file, include file, or object module in order to write it.

Related Information
See the OpenVOS Forms Management System manuals for a complete description of the Forms Editor requests.
pascal

Purpose
This command compiles an OpenVOS Pascal source module.

Display Form

```
source_file_name: source_file_name
-define: define
-processor: default
-mapping_rules: default
-list: no
-table: no
-optimize: yes
-mapcase: no
-cpu_profile: no
-check_overflow: no
-silent: no
-nesting: no
-optimization_level: 3

-xref: no
-production_table: no
-check: no
-profile: no
-statistics: no
-check_conformance: no
-full: no
-system_programming: no
-check_uninitialized: no
```
Command Line Form

```
pascal source_file_name
    [-define variable_name...]
    [-processor processor_string]
    [-mapping_rules mapping_string]
    [-list]
    [-xref]
    [-table]
    [-production_table]
    [-no_optimize]
    [-check]
    [-mapcase]
    [-profile]
    [-cpu_profile]
    [-statistics]
    [-check_overflow]
    [-check_conformance]
    [-silent]
    [-full]
    [-nesting]
    [-system_programming]
    [-optimization_level number]
    [-check_uninitialized]
```

Arguments

- `source_file_name` Required
  The path name of an OpenVOS Pascal source module.

- `-define variable_name`
  Defines variables to be used by the preprocessor. These variables are used during
  the preprocessor phase of the compilation. Preprocessor variables can contain
  letters, digits, or the underline character (_), in any position. (See the Explanation
  section of this command description and the description of the
  `preprocess_file` command for details.)

- `-processor processor_string`
  Specifies the processor on which the program module (.pm) is to run. The display form
  for the `-processor` argument restricts the values that you can choose to values for
  the processor family of the current module.

  If the current module uses a processor from the IA-32 family, or if you specify, on the
  command line, the `-processor` argument with the `pentium4` value, the allowed
  `processor_string` values are as follows:

  - `default`
  - `pentium4`

  The default value indicates the system-wide default. Unless your system
  administrator has reset this value, default is `pentium4` for modules using IA-32
  processors. To determine the default value, issue the `display_error`
m$default_processor command. By default, the compiler produces code intended for the processor specified by default.

**-mapping_rules mapping_string**

Specifies one of the following data alignment rules for a given compilation.

- default
- default/check
- shortmap
- shortmap/check
- longmap
- longmap/check

The default value indicates the system-wide default. The default alignment method is site-settable. To determine the default value, issue the display_error m$default_mapping command. By default, the compiler uses the data alignment rules specified by default. (See the Explanation section of this command description for details.)

**-list**

Creates a compilation listing. A compilation listing shows all source statements from the source module and include files, as well as a summary of all data definitions and the path names of include files used. You need not specify -list if you specify -full, -nesting, or -xref, since those arguments create a compilation listing in addition to other listings. By default, the compiler does not generate a compilation listing.

**-xref**

Creates a compilation listing and an alphabetized cross-reference listing of all data actually referenced in the program. By default, the compiler does not generate a cross-reference listing.

**-table**

Creates a symbol table in the object module, for use by the debugger. The compiler also performs some related operations. (See the Explanation section of this command description for details.) In addition, -table suppresses interstatement code optimization, which results in code that is slower than normal. Specifying -table sets the maximum optimization level to 1, unless you explicitly set the level to 0. By default, the compiler does not create a symbol table, suppress interstatement code optimization, or perform any related operations.

Note: A symbol table greatly increases the size of an object module.

If you specify both -table and -production_table, the compiler produces only a production table and sets the maximum optimization level to 3, unless you explicitly specify some other value.

**-production_table**

Creates a symbol table in the object module, for use by the debugger in a production environment. Only variables actually referenced in the program are placed in the symbol table. The compiler also performs some related operations. (See the Explanation section of this command description for details.) Unlike -table,
-production_table does not suppress interstatement code optimization. As a result, invoking the set and continue requests of the debug command can cause unpredictable results. Also, the contents of variables in registers cannot be accurately displayed with the display request of the debug command. In addition, if the optimization level is greater than 2, the contents of any variables may not be accurately displayed with the display request of the debug command. Specifying -production_table sets the maximum optimization level to 3, unless you explicitly specify some other value. By default, the compiler does not create a symbol table, suppress interstatement code optimization, or perform any related operations.

Note: A symbol table greatly increases the size of an object module.

If you specify both -production_table and -table, the compiler produces only a production table and sets the maximum optimization level to 3, unless you explicitly specify some other value.

► -no_optimize

Generates the object code without optimizing it. Optimization produces more compact object code by removing unnecessary or redundant computations. Specifying -no_optimize sets the optimization level to 0. This overrides any other specification of the optimization level. By default, the compiler optimizes the object code.

► -check

Adds code to the object module that checks for the following errors at run time:

- out-of-bounds array subscripts
- out-of-range string references in the predefined procedure substr
- out-of-range assignments to variables of enumerated or subrange types
- more than 256 elements assigned to variables declared to be sets
- out-of-range arguments to the predefined function chr. The range is 0 to 255.

If an array subscript is assigned a constant, the error-checking code may find the error at compile time because of constant folding. Otherwise, the errors are found at run time. By default, the compiler does not check and does not insert the checking code.

► -mapcase

Interprets all uppercase letters except those in character-string constants as lowercase letters. If you specify -mapcase and the source module contains an external variable name or entry name, you may not be able to bind the resulting object module. (See the Explanation section of this command description for details.) By default, the compiler distinguishes between uppercase and lowercase letters.

► -profile

Inserts code in the compiled program that counts the number of times each source statement executes when the program runs. (See the description of the profile and add_profile commands.) By default, the compiler does not insert the counting code. You cannot specify both -profile and -cpu_profile in the same command.

► -cpu_profile

Inserts code in the compiled program that counts the number of times each source statement executes, the amount of CPU time (in milliseconds) spent executing each
statement, and the number of page faults taken executing each statement when the program runs. (See the description of the profile and add_profile commands.) By default, the compiler does not insert the counting code. Note that the code inserted by this argument uses much more CPU time, but provides more useful information, than the code inserted by -profile. You cannot specify both -cpu_profile and -profile in the same command.

► -statistics
   Displays statistics about the compilation as it proceeds. The compiler displays the version number of the compiler as well as the following statistics for each phase.
   
   • disk I/O information
   • elapsed real time
   • amount of storage used
   • number of page faults taken
   • elapsed CPU time
   • time when the compiler completed the phase

The compiler also displays statistical information for the entire compilation, such as the number of source lines and the symbol table size.

You can specify -statistics to see the progress of the compilation and to determine the phase in which an error occurs. If the compiler produces a listing, it puts the statistics in the listing. By default, the compiler does not display compilation statistics.

► -check_overflow
   Generates code that detects the fixed-point arithmetic overflow condition when the program runs. By default, the compiler does not insert the checking code. In that case, the operating system will detect some instances of fixed-point arithmetic overflow at run time.

► -check_conformance
   Verifies that certain constructs conform to ANSI rules. Specifically, it verifies that the type of an actual parameter matches the type of its corresponding pass-by-reference formal parameter if the formal parameter’s data type is a subrange. By default, the compiler does not check for matching subrange types.

► -silent
   Suppresses the warning messages of severity-1 or severity-0 errors on the terminal during compilation. The compiler, nevertheless, puts the messages in an error file and in any listing it produces. By default, the compiler writes all error messages on your terminal.

► -full
   Creates, from the compiled object code, an assembly language listing with added comments, as well as a compilation listing. The compiler uses a disassembler to produce the listing. By default, the compiler does not create an assembly language listing.
-nesting

Produces a compilation listing, with the nesting level of source statements in the listing: the top level is 0, the next level is 1, and so forth. By default, the compiler does not place the nesting level of source statements in any listing it produces.

-system_programming

Diagnoses alignment padding within longmapped records. By default, the compiler does not perform this checking. (See the VOS Pascal Language Manual (R014) for more information on longmapped records.)

-optimization_level number

Specifies the degree of optimization. The possible values are 0, 1, 2, 3, and 4. (See the Explanation section of this command description for details.)

-check_uninitialized

Issues diagnostics for all references to uninitialized variables if you also specify the value of -optimization_level as 3 or 4. If you specify this argument and a value for -optimization_level that is less than 3, the compiler issues an error. This argument is useful when verifying new code or checking for possible bugs, but it can return misleading diagnostics, as in the case of variables that are initialized within a conditional statement. The categories of uninitialized variables diagnosed by the compiler vary, depending on whether you choose both -check_uninitialized and an optimization level of at least 3, or choose only an optimization level of at least 3.

Explanation

The pascal command compiles an OpenVOS Pascal source module into an object module. The source module’s name must have the suffix .pascal. You can either supply or omit the suffix when you specify source_file_name. The compiler generates an object module, puts it in your current directory, and names it. The name of the object module is the name of the source module with the suffix changed from .pascal to .obj.

When you are compiling for an ftServer module at all optimization levels, the module on which you are compiling must have at least 30,000 pages of paging partition available to avoid running out of virtual memory. In addition, the module on which you are compiling should have 64MB of physical memory available to achieve optimal compiler performance.

Using the -define Argument

The -define argument defines variables to be used during the preprocessor phase of the compilation. For example, if you specify the following on the command line, the preprocessor variables var_a and var_b will be initially defined during the preprocessing phase of the compilation:

    pascal prog1 -define var_a var_b
You use preprocessor variables with preprocessor statements to perform conditional compilation on a program. *Conditional compilation* enables you to switch on or off various statements in a program. This is useful, for example, if you want your program to compile different lines of source code on different processors. There are six preprocessor statements.

- `$define`
- `$undefine`
- `$if`
- `$else`
- `$elseif`
- `$endif`

Preprocessor statements must begin in the first column of the source program. Therefore, indentation of nested `$if` statements is not allowed.

A preprocessor statement must be contained on a single line. A line containing a preprocessor statement cannot contain comments or parts of the source language. (An exception is the `$endif` statement, which ignores any text following it on the line, thus allowing you to comment on the source code.)

To comment out a preprocessor statement, the comment delimiters must surround the statement on the same line, or the comment delimiters must open and close on lines surrounding the preprocessor statement. A comment delimiter cannot appear on the same line as the statement if the corresponding comment delimiter appears on a different line. Examples of valid and invalid comments follow.

**Valid:**

```
{ $define pentium4 }
```

**Valid:**

```
(* $define pentium4 *)
```

**Invalid:**

```
{ $endif }
```

For more information on the preprocessor, see the description of the `preprocess_file` command.

**Using the `-processor` Argument**

The `-processor` argument allows you to specify the processor on which the program is to run. The `-processor` argument also allows you to perform cross-compilation on a source module if the Pascal cross compiler is available on your system. *Cross-compilation* occurs when a compiler running on one processor family translates a source module into object code for another processor family. The IA-32 cross compiler generates code to run on ftServer
modules. Specify the value `pentium4` for the `-processor` argument to target an ftServer module.

Depending on the value specified in the `-processor` argument, the compiler automatically defines one preprocessor variable for the processor family and one or more preprocessor variables corresponding to the processor type(s), as shown in Table 2-25.

### Table 2-25. Predefined Preprocessor Variables

<table>
<thead>
<tr>
<th>Processor Value</th>
<th>Preprocessor Variable</th>
</tr>
</thead>
<tbody>
<tr>
<td>default</td>
<td>Varies, depending on the default system processor</td>
</tr>
<tr>
<td>pentium4</td>
<td><code>__PENTIUM4__</code>, <code>__IA32__</code>, and <code>__i386</code></td>
</tr>
</tbody>
</table>

If you specify `-processor pentium4` on the command line, the preprocessor variables `__PENTIUM4__`, `__IA32__`, and `__i386` are defined.

If the value specified in the `-processor` argument indicates the IA-32 processor, the maximum number of bytes available for each function’s initial stack frame is 2,147,483,584 bytes.

The amount of automatic storage you can actually declare is somewhat less than these limits because temporary variables generated by the compiler also count towards the limit.

**Note:** Although the OpenVOS Pascal compiler supports extremely large values (such as 2,147,483,646), the operating system does not support them.

### Using the `-mapping_rules` Argument

The `-mapping_rules` argument allows you to specify the data alignment rules for a given compilation. The value `default` indicates the system-wide default. The default is site-settable. The value `shortmap` specifies that the shortmap alignment rules are to be used for the source module. The value `longmap` specifies that the longmap alignment rules are to be used for the source module. The values `default/check`, `shortmap/check`, and `longmap/check` are equivalent to `default`, `shortmap`, and `longmap`, respectively, except that they also diagnose alignment padding within records. For example, if you specify `default/check`, the compiler displays a severity-0 message stating how many bytes of padding exist between fields within a record. The `%options` mapping directives override `-mapping_rules` values, but alignment padding within records is still diagnosed if you specify one of the checking values.

For more information on data alignment rules, see the *VOS Pascal Language Manual* (R014).

### Using the `-full`, `-nesting`, `-list`, or `-xref` Argument

If you specify the `-list`, `-nesting`, `-full`, or `-xref` argument, the compiler creates a compilation listing file and puts it in your current directory. The name of the compilation listing is `source_file_name.list`. Specifying `-full` creates an assembly language listing in addition to a compilation listing. Specifying `-nesting` adds numbers showing the nesting depth of each source statement in a compilation listing. Specifying `-xref` creates a list of cross-references in addition to a compilation listing.
Using the \texttt{-table} or \texttt{-production_table} Argument

If you specify the \texttt{-table} argument, the compiler creates a symbol table, and allocates storage and generates addresses for all external references, including any that are not used. Symbol-table capacity is $2,147,483,647$ nodes. The compiler generates internal subroutines that calculate size, offset, and bound expressions that determine the characteristics of adjustable data. These subroutines allow the debugger to display and modify variable-length data according to its current length. In addition, the compiler suppresses interstatement code optimization.

If you specify the \texttt{-production_table} argument, the compiler performs all of the same operations, except that it does not suppress interstatement code optimization, and only variables actually referenced in the program are placed in the symbol table. Code produced with \texttt{-table} executes more slowly than code produced with \texttt{-production_table}. Code produced with \texttt{-production_table} can yield unpredictable results if you invoke the \texttt{set} and \texttt{continue} debugger requests.

Using the \texttt{-mapcase} Argument

When you compile a source module specifying the \texttt{-mapcase} argument, and the module contains an external variable name or entry name with one or more uppercase letters, you may not be able to bind the resulting object module. If the binder encounters a reference to the original name (for example, in a binder control file), it does not recognize the original name and its lowercase version as the same name.

Optimizations for ftServer Modules

The \texttt{-optimization_level} argument allows you to optimize programs at different levels. When you are compiling a source module to run on ftServer modules, the levels of optimizations are 1, 2, 3, and 4. Specifying optimization level 3 or 4 causes the compiler to perform level 3 optimizations.

If you specify optimization level 0, the compiler performs the following local optimizations.

- local register allocation
- elimination of unreachable code

If you specify optimization level 1, the compiler performs all level 0 optimizations plus the following other local optimizations.

- local pattern replacement
- short-circuit evaluation of Boolean expressions
- recognition of algebraic identities
- constant folding
- local combination of common subexpressions within a statement
- peephole optimizations within a single statement
- result incorporation

If you specify optimization level 2, the compiler performs all level 1 optimizations plus the following global optimizations.

- branch retargeting
- elimination of unreachable code
If you specify optimization level 3, the compiler performs all level 2 optimizations plus the following global optimizations.

- constant propagation
- removal of invariant assignments from loops
- strength reduction
- linear test replacement
- elimination of dead assignments
- elimination of useless loops
- check for uninitialized variables
- elimination of dead code and dead stores
- instruction scheduling
- no allocation of stack space by automatic variables whose values are kept in registers

Specifying the Optimization Level
The level of optimization is determined by the arguments -no_optimize, -table, and -optimization_level. Specifying -no_optimize sets the optimization level to 0. Specifying -table sets the level to 1, unless you explicitly set the level to 0. The -optimization_level argument sets the level to any of the permitted levels: 0, 1, 2, or 3. The compiler sets the actual level to the lowest level set by any of the three arguments. By default, the level is 3.

Note: If you compile a program with either the -profile or -cpu_profile argument, you must specify an optimization level lower than 3. Otherwise, -profile or -cpu_profile might not return accurate information, since high optimization levels can cause code to be moved from one statement to another.

Using the -check_uninitialized Argument
The optimization level for a source module also affects the functionality of the -check_uninitialized argument.

- If you select the -check_uninitialized argument and an optimization level of at least 3, the compiler diagnoses all instances of uninitialized variables within the source module. In this case, the compiler diagnoses variables that are initialized as part of code executed conditionally.

- If you do not select the -check_uninitialized argument but do select an optimization level of at least 3, the compiler diagnoses instances of variables within the source module that it knows are uninitialized. In this case, the compiler does not diagnose variables that are initialized as part of code executed conditionally.

- If you select an optimization level of less than 3, the compiler issues an error and does not diagnose uninitialized variables within the source module even if you select -check_uninitialized.
Interpreting Compiler Diagnostics

If the compiler discovers any errors in your source module, it displays an error message on your terminal. However, severity-1 and severity-0 messages are not displayed on your terminal when you specify -silent. The compiler also creates an error file named source_file_name.error in the current directory and writes the error messages to the file. The compiler also appends the error messages to a compilation listing if it produces one. The system deletes any .error file if a subsequent compile to the same source file is successful (contains no errors).

The OpenVOS Pascal compiler diagnoses five types of errors.

SEVERITY 0: Advice
SEVERITY 1: Warning
SEVERITY 2: Correctable error
SEVERITY 3: Uncorrectable error: translation can continue
SEVERITY 4: Uncorrectable error: translation cannot continue

The text of the error message explains the cause of the error.

A severity-0 error, although valid Pascal, indicates that improvement is possible, usually in the area of performance. The source module is syntactically correct, so the compiled object module can be bound and executed, but probably with less than optimum efficiency.

A severity-1 error, although valid Pascal, is probably a programming error. Since the source module is syntactically correct at the point of a severity-1 error, however, the compiler continues to compile the source. The compiled object module can be bound and executed, but the program probably will not perform as expected.

A severity-2 error is invalid Pascal, but the compiler can reinterpret the source in such a way that it can continue to compile the program. The compiler proceeds as if the faulty code were replaced with the most likely syntactically correct code. The compiled object module can be bound and executed, but the program probably will not perform as expected.

A severity-3 error is invalid Pascal, and the compiler cannot reinterpret the source in such a way that it can continue to process the program into a usable object module. Nevertheless, the compiler continues to process the program to detect additional errors. However, the object module is not created.

A severity-4 error is invalid Pascal, and the compiler cannot reinterpret the source in such a way that it can continue to process the program from the point of the severity-4 error. The object module is not created.

Note: If the compilation results in more than 100 errors, in any combination (excluding severity-0 errors), compilation terminates.

The compiler always overwrites an existing object module having the same name as the object module it produces.
Access Requirements

You need read access to the source module to compile it. You need modify access to the directory from which you are issuing the compile command and in which the .obj file will be created.

Examples

The following command compiles inv_report.pascal, generates code that will perform a series of range checks when the object module runs, and produces a compilation listing file inv_report.list in your current directory.

```
pascal inv_report -check -list
```

Related Information

See the VOS Pascal Language Manual (R014) for a complete description of the OpenVOS Pascal language.
**pl1**

**Purpose**
This command compiles an OpenVOS PL/I source module.

**Display Form**

```plaintext
source_file_name: pl1

-define:
-processor: default
-mapping_rules: default
-list: no -xref: no
-table: no -production_table: no
-optimize: yes -check: no
-mapcase: no -profile: no
-cpu_profile: no -statistics: no
-fixedoverflow: no -silent: no
-full: no -nesting: no
-system_programming: no -optimization_level: 3
-check_uninitialized: no -max_fixed_bin: 31
-store_args: no
```

2-588
Command Line Form

```
pl1 source_file_name
  [ -define variable_name... ]
  [ -processor processor_string ]
  [ -mapping_rules mapping_string ]
  [ -list ]
  [ -xref ]
  [ -table ]
  [ -production_table ]
  [ -no_optimize ]
  [ -check ]
  [ -mapcase ]
  [ -profile ]
  [ -cpu_profile ]
  [ -statistics ]
  [ -fixedoverflow ]
  [ -silent ]
  [ -full ]
  [ -nesting ]
  [ -system_programming ]
  [ -optimization_level number ]
  [ -check_uninitialized ]
  [ -max_fixed_bin number ]
  [ -store_args ]
```

Arguments

- **source_file_name**  
  The path name of an OpenVOS PL/I source module.

- **-define variable_name**  
  Defines variables to be used by the OpenVOS preprocessor. These variables are used during the preprocessor phase of the compilation. Preprocessor variables can contain letters, digits, or the underline character (_), in any position. (See the Explanation section of this command description and the description of the `preprocess_file` command for details.)

- **-processor processor_string**  
  Specifies the processor on which the program module (.pm) is to run. The display form for the -processor argument restricts the values that you can choose to values for the processor family of the current module.
If the current module uses a processor from the IA-32 family, or if you specify, on the command line, the \texttt{-processor} argument with the \texttt{pentium4} value, the allowed \texttt{processor\_string} values are as follows:

- \texttt{default}
- \texttt{pentium4}

The \texttt{default} value indicates the system-wide default. Unless your system administrator has reset this value, \texttt{default} is \texttt{pentium4} for modules using IA-32 processors. To determine the \texttt{default} value, issue the \texttt{display\_error m$default\_processor} command. By default, the compiler produces code intended for the processor specified by \texttt{default}.

\begin{itemize}
  \item \textbf{\texttt{-mapping\_rules mapping\_string}} \hspace{1cm} \texttt{[CYCLE]}
  \end{itemize}

Specifies one of the following data alignment rules for a given compilation.

- \texttt{default}
- \texttt{default/check}
- \texttt{shortmap}
- \texttt{shortmap/check}
- \texttt{longmap}
- \texttt{longmap/check}

The \texttt{default} value indicates the system-wide default. The default alignment method is site-settable. To determine the \texttt{default} value, issue the \texttt{display\_error m$default\_mapping} command. By default, the compiler uses the data alignment rules specified by \texttt{default}. (See the Explanation section of this command description for details.)

\begin{itemize}
  \item \textbf{\texttt{-list}} \hspace{1cm} \texttt{[CYCLE]}
  \end{itemize}

Creates a compilation listing. A compilation listing shows all source statements from the source module and include files, as well as a summary of all data definitions and the path names of include files used. You need not specify \texttt{-list} when you specify \texttt{-full}, \texttt{-nesting}, or \texttt{-xref}; these arguments create a compilation listing in addition to other listings. By default, the compiler does not generate a compilation listing.

\begin{itemize}
  \item \textbf{\texttt{-xref}} \hspace{1cm} \texttt{[CYCLE]}
  \end{itemize}

Creates a compilation listing and an alphabetized cross-reference listing of all data actually referenced in the program. By default, the compiler does not generate a cross-reference listing.

\begin{itemize}
  \item \textbf{\texttt{-table}} \hspace{1cm} \texttt{[CYCLE]}
  \end{itemize}

Creates a symbol table in the object module, for use by the OpenVOS Symbolic Debugger. The compiler also performs some related operations. (See the Explanation section of this command description for details.) In addition, \texttt{-table} suppresses interstatement code optimization, which results in code that is slower than normal. Specifying \texttt{-table} sets the maximum optimization level to 1, unless you explicitly set the level to 0. By default, the compiler does not create a symbol table, suppress interstatement code optimization, or perform any related operations.

\textbf{Note:} A symbol table greatly increases the size of an object module.
If you specify both -table and -production_table, the compiler produces only a production table and sets the maximum optimization level to 3, unless you explicitly specify some other value.

**-production_table**

Creates a symbol table in the object module, for use by the OpenVOS Symbolic Debugger in a production environment. Only variables actually referenced in the program are placed in the symbol table. The compiler also performs some related operations. (See the Explanation section of this command description for details.) Unlike -table, -production_table does not suppress interstatement code optimization. As a result, the set and continue requests of the debug command can lead to unpredictable results. Also, the contents of variables in registers cannot be accurately displayed with the display request of the debug command. In addition, if the optimization level is greater than 2, the contents of any variables may not be accurately displayed with the display request of the debug command. Specifying -production_table sets the maximum optimization level to 3, unless you explicitly specify some other value. By default, the compiler does not create a symbol table, suppress interstatement code optimization, or perform any related operations.

**Note:** A symbol table greatly increases the size of an object module.

If you specify both -production_table and -table, the compiler produces only a production table and sets the optimization level to 3, unless you explicitly specify some other value.

**-no_optimize**

Generates the object code without optimizing it. Optimization produces more compact object code by removing unnecessary or redundant computations. Specifying -no_optimize sets the optimization level to 0. This overrides any other specification of the optimization level. By default, the compiler optimizes the object code.

**-check**

Checks for out-of-bounds array subscripts and out-of-range substring references. The compiler checks while compiling and inserts code to further check when the program is run. By default, the compiler does not check or insert checking code.

**-mapcase**

Interprets all uppercase letters except those in character-string constants as lowercase letters. If you specify -mapcase and the source module contains an external variable name or entry name, you may not be able to bind the resulting object module. (See the Explanation section of this command description for details.) By default, the compiler distinguishes between uppercase and lowercase letters.

**-profile**

Inserts code in the compiled program that counts the number of times each source statement is executed when the program runs. (See the description of the profile and add_profile commands.) By default, the compiler does not insert the counting code. You cannot specify both -profile and -cpu_profile in the same command.
-cpu_profile

Inserts code in the compiled program that counts the number of times each source statement is executed, the amount of CPU time (in milliseconds) spent executing each statement, and the number of page faults taken executing each statement when the program runs. (See the description of the profile and add_profile commands.) By default, the compiler does not insert the counting code. The code inserted by this argument uses much more CPU time, but provides more useful information, than the code inserted by -profile. You cannot specify both -cpu_profile and -profile in the same command.

-statistics

Displays statistics about the compilation as it proceeds. The compiler displays the version number of the compiler, and the following statistics for each phase:

- disk I/O information
- elapsed real time
- amount of storage used
- number of page faults taken
- elapsed cpu time
- time when the compiler completed the phase

The compiler also displays statistical information for the entire compilation, such as the number of source lines and the symbol table size.

You can specify -statistics to see the progress of the compilation and to determine the phase in which an error occurs. If the compiler produces a listing, it puts the statistics in the listing. By default, the compiler does not display compilation statistics.

-fixedoverflow

Generates code to check for fixed-point arithmetic overflow when the program is run and signals the fixedoverflow condition when it occurs. By default, the compiler does not insert the checking code. Some instances of fixed-point arithmetic overflow may nevertheless be detected when the program runs.

-silent

Suppresses the warning messages of severity-1 or severity-0 errors on your terminal during compilation. The compiler, nevertheless, puts the messages in an error file and in any listing it produces. By default, the compiler writes all error messages on your terminal.

-full

Creates from the compiled object code an assembly language listing with added comments, as well as a compilation listing. The compiler uses a disassembler to produce the listing. By default, the compiler does not produce an assembly language listing.

-nesting

Produces a compilation listing with the nesting level of source statements in the listing: the top level is 0, the next level is 1, and so forth. By default, the compiler does not print the nesting level of source statements in any listing it produces.
-system_programming
    Produces more stringent checking of the program during compilation. The checking
diagnoses the following:
    
    - references to members of a structure without the level-1 structure name
    - some cases of implicit data type conversion
    - missing members in a label array for which no default case exists
    - alignment padding in structures (if the source module’s data alignment method
      is longmap)

-optimization_level number
    Specifies the degree of optimization. The possible values are 0, 1, 2, 3, and 4. (See the
    Explanation section of this command description for details.)

-check_uninitialized
    Issues diagnostics for all references to uninitialized variables if you also specify the
    value of -optimization_level as 3 or 4. If you specify this argument and an
    optimization level of less than 3, the compiler issues an error. This argument is useful
    when verifying new code or checking for possible bugs, but it can return misleading
    diagnostics, as in the case of variables that are initialized within a conditional
    statement. The categories of uninitialized variables diagnosed by the compiler vary,
    depending on whether you choose both -check_uninitialized and an
    optimization level of at least 3, or choose only an optimization level of at least 3.

-max_fixed_bin number
    Specifies the maximum precision (either 31, which is the default, or 63) for fixed binary
    values in a PL/I compilation unit. You can override it with the max_fixed_bin option
    of the %options preprocessor statement. See the OpenVOS PL/I Language
    Manual (R009) for more information about the max_fixed_bin option.

-store_args
    This argument has no effect on programs compiled for ftServer modules but has been
    retained for compatibility with existing software build scripts.

Explanation

The pl1 command compiles an OpenVOS PL/I source module into an object module.

The name of the source module must have the suffix .pl1; you can either supply or omit the
suffix when you give source_file_name. The compiler generates an object module, puts
it in your current directory, and names it. The name of the object module is the same as the
name of the source module with the suffix changed from .pl1 to .obj.

The OpenVOS PL/I compiler invokes two preprocessors: the OpenVOS preprocessor and the
PL/I preprocessor. The -define argument can be used with the OpenVOS preprocessor;
-define does not affect the PL/I preprocessor. See the OpenVOS PL/I Language
Manual (R009) and VOS PL/I User’s Guide (R145) for more information on the PL/I
preprocessor.

When you are compiling for an ftServer module at all optimization levels, the module on
which you are compiling must have at least 30,000 pages of paging partition available to
avoid running out of virtual memory. In addition, the module on which you are compiling should have 64MB of physical memory available to achieve optimal compiler performance.

Using the \texttt{-define} Argument

The \texttt{-define} argument defines variables to be used during the preprocessor phase of the compilation. For example, if you specify the following on the command line, the preprocessor variables \texttt{var\_a} and \texttt{var\_b} will be initially defined during the preprocessing phase of the compilation:

\begin{verbatim}
pl1 prog1 -define var\_a var\_b
\end{verbatim}

You use preprocessor variables with preprocessor statements to perform conditional compilation on a program. \textit{Conditional compilation} enables you to switch on or off various statements in a program. This is useful, for example, if you want your program to compile different lines of source code on different processors. There are six preprocessor statements.

\begin{itemize}
\item \texttt{define}
\item \texttt{undefine}
\item \texttt{if}
\item \texttt{else}
\item \texttt{elseif}
\item \texttt{endif}
\end{itemize}

Preprocessor statements must begin in the first column of the source program. Therefore, indentation of nested \texttt{if} statements is not allowed.

A preprocessor statement must be contained on a single line. A line containing a preprocessor statement cannot contain comments or parts of the source language. (An exception is the \texttt{endif} statement, which ignores any text following it on the line, thus allowing you to comment on the source code.)

To comment out a preprocessor statement, the comment delimiters must surround the statement on the same line, or the comment delimiters must open and close on lines surrounding the preprocessor statement. A comment delimiter cannot appear on the same line as the statement if the corresponding comment delimiter appears on a different line. Examples of valid and invalid comments follow.

\textbf{Valid:}

\begin{verbatim}
/*
 $define
 */
\end{verbatim}

\textbf{Valid:}

\begin{verbatim}
/* $define pentium4 */
\end{verbatim}

\textbf{Invalid:}

\begin{verbatim}
/*
 $endif
 */
\end{verbatim}
For more information on the OpenVOS preprocessor, see the description of the preprocess_file command.

**Using the -processor Argument**

The -processor argument allows you to specify the processor on which the program is to run. The -processor argument also allows you to perform cross-compilation on a source module if the PL/I cross compiler is available on your system. *Cross-compilation* occurs when a compiler running on one processor family translates a source module into object code for another processor family. The IA-32 cross compiler generates code to run on ftServer modules. Specify the value pentium4 for the -processor argument to target an ftServer module.

Depending on the value specified in the -processor argument or the corresponding $options processor option, the compiler automatically defines one preprocessor variable for the processor family and one or more preprocessor variables corresponding to the processor type(s), as shown in Table 2-26.

**Table 2-26. Predefined Preprocessor Variables**

<table>
<thead>
<tr>
<th>Processor Value</th>
<th>Preprocessor Variable</th>
</tr>
</thead>
<tbody>
<tr>
<td>default</td>
<td>Varies, depending on the default system processor</td>
</tr>
<tr>
<td>pentium4</td>
<td><strong>PENTIUM4</strong>, <strong>IA32</strong>, and __i386</td>
</tr>
</tbody>
</table>

If you specify -processor pentium4 on the command line, the preprocessor variables __PENTIUM4__, __IA32__, and __i386 are defined.

If the value specified in the -processor argument indicates the IA-32 processor, the maximum number of bytes available for each function’s initial stack frame is 2,147,483,584 bytes.

The amount of automatic storage you can actually declare is somewhat less than these limits because temporary variables generated by the compiler also count towards the limit.

**Note:** Although the OpenVOS PL/I compiler supports extremely large values (such as 2,147,483,646), the operating system does not support them.

**Using the -mapping_rules Argument**

The -mapping_rules argument allows you to specify the data alignment rules for a given compilation. The value default indicates the system-wide default. The default is site-settable. The value shortmap specifies that the shortmap alignment rules are to be used for the source module. The value longmap specifies that the longmap alignment rules are to be used for the source module. The values default/check, shortmap/check, and longmap/check are equivalent to default, shortmap, and longmap, respectively, except that they also diagnose alignment padding within structures. For example, if you specify default/check, the compiler displays a diagnostic message stating how many bytes of padding exist within a structure. The $options mapping directives override -mapping_rules values, but alignment padding within structures is still diagnosed if you specify one of the checking values.
Using the `-full`, `-nesting`, `-list`, or `-xref` Argument

If you specify the `-list`, `-nesting`, `-full`, or `-xref` argument, the compiler creates a compilation listing file and puts it in your current directory. The name of the compilation listing is `source_file_name.list`. Specifying `-full` gives you an assembly language listing in addition to a program listing. Specifying `-nesting` adds numbers showing the nesting depth of each source statement in a program listing. Specifying `-xref` gives you a list of cross-references in addition to a program listing. See the VOS PL/I User’s Guide (R145) for examples of each type of listing.

Optimizations for ftServer Modules

The `-optimization_level` argument allows you to optimize programs at different levels. When you are compiling a source module to run on ftServer modules, the levels of optimizations are 1, 2, 3, and 4. Specifying optimization level 3 or 4 causes the compiler to perform level 3 optimizations.

If you specify optimization level 0 (or `-no_optimize`), the compiler performs the following local optimizations.

- local register allocation
- elimination of unreachable code

If you specify optimization level 1, the compiler performs all level 0 optimizations plus the following other local optimizations.

- local pattern replacement
- short-circuit evaluation of Boolean expressions
- recognition of algebraic identities
- constant folding
- local combination of common subexpressions within a statement
- result incorporation
- peephole optimizations within a single statement

If you specify optimization level 2, the compiler performs all level 1 optimizations plus the following global optimizations.

- branch retargeting
- global combination of common subexpressions
- removal of invariant expressions from loops
- subsumption
- peephole optimizations across statement boundaries
- global register allocation

If you specify optimization level 3, the compiler performs all level 2 optimizations plus the following global optimizations.

- constant propagation
- removal of invariant assignments from loops
- strength reduction
- linear test replacement
- elimination of dead assignments
- elimination of useless loops
• check for uninitialized variables
• elimination of dead code and dead stores
• instruction scheduling
• inline expansion
• no allocation of stack space by automatic variables whose values are kept in registers

As stated above, unreachable code is eliminated at all optimization levels on ftServer modules. Sometimes, however, you might want your program to contain some code that will be executed only during a debugging session, not during normal program execution. To prevent the compiler from eliminating such unreachable code, you might consider changing your program as follows.

```pl1
declare always_zero    fixed bin(15) volatile static initial (0);
if (always_zero ^= 0) then
    /* Code that should not be eliminated goes here */
```

If you delete the `volatile` attribute from the preceding declaration, the compiler will eliminate the unreachable code. See the *OpenVOS PL/I Language Manual* (R009) for more information on `volatile`.

**Specifying the Optimization Level**

The arguments `-no_optimize`, `-table`, and `-optimization_level` determine the optimization level for a source module. By default, the level is 3.

Table 2-27 describes how each of these compiler arguments affects the optimization level for a source module.

**Table 2-27. Optimization-Related Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Optimization Level for a Source Module</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>-optimization_level</code></td>
<td>Specifies the level of optimization that the compiler uses. Allowed values are 0, 1, 2, 3, and 4. The default level is 3.</td>
</tr>
<tr>
<td><code>-no_optimize</code></td>
<td>Specifies optimization level 0. This argument overrides the <code>-optimization_level</code> argument as well as the optimization level associated with the <code>-table</code> argument if either of these arguments is specified.</td>
</tr>
<tr>
<td><code>-table</code></td>
<td>Specifies a maximum optimization level of 1. This argument overrides the <code>-optimization_level</code> argument if that argument is specified with a value greater than 0. If you specify this argument, the compiler does not perform any global optimizations.</td>
</tr>
</tbody>
</table>

**Note:** If you compile a program with either the `-profile` or `-cpu_profile` argument, you must specify an optimization level lower than 3. Otherwise, `-profile` or `-cpu_profile` might not return accurate information, since high optimization levels can cause code to be moved from one statement to another.
Using the \texttt{-check\_uninitialized} Argument

The optimization level for a source module also affects the functionality of the \texttt{-check\_uninitialized} argument.

- If you select the \texttt{-check\_uninitialized} argument \textbf{and} an optimization level of at least 3, the compiler diagnoses all instances of uninitialized variables within the source module. In this case, the compiler diagnoses variables that are initialized as part of code executed conditionally.

- If you do not select the \texttt{-check\_uninitialized} argument \textbf{but} do select an optimization level of at least 3, the compiler diagnoses instances of variables within the source module that it knows are uninitialized. In this case, the compiler does not diagnose variables that are initialized as part of code executed conditionally.

- If you select an optimization level of less than 3, the compiler issues an error and does not diagnose uninitialized variables within the source module even if you select \texttt{-check\_uninitialized}.

Using the \texttt{-table} or \texttt{-production\_table} Argument

If you specify the \texttt{-table} argument, the compiler creates a symbol table, and allocates storage and generates addresses for all external references, including any that are not used. Symbol-table capacity is 2,147,483,647 nodes. The compiler generates internal subroutines that calculate size, offset, and bound expressions that determine the characteristics of adjustable data. This allows the OpenVOS Symbolic Debugger to display and modify variable-length data according to its current length. In addition, the compiler suppresses interstatement code optimization and only allocates storage for the references that have been used.

If you specify \texttt{-production\_table}, the compiler performs all of the same operations that it performs for \texttt{-table}, except that it does not suppress interstatement code optimization, and only variables actually referenced in the program are placed in the symbol table (most unreferenced variables are from include files). Code produced with \texttt{-table} executes more slowly than does code produced with \texttt{-production\_table}. Code produced with \texttt{-production\_table} can produce unpredictable results if you invoke the OpenVOS Symbolic Debugger set and continue requests.

Using the \texttt{-mapcase} Argument

When you compile a source module with the \texttt{-mapcase} argument, and the module contains an external variable name or entry name with one or more uppercase letters, you may not be able to bind the resulting object module. If the binder encounters a reference to the original name (for example, in a binder control file), it will not recognize the original name and its lowercase version as the same name.

Interpreting Compiler Diagnostics

If the compiler discovers any errors in your source module, it displays an error message on your terminal. Severity-1 and severity-0 messages are not displayed on your terminal when you specify the \texttt{-silent} argument. The compiler also creates an error file named \texttt{source\_file\_name.error} in the current directory and writes the error messages to the file. The compiler also appends error messages to a compilation listing if it produces one. The
system deletes any .error file if a subsequent compile to the same source file is successful (contains no errors).

The OpenVOS PL/I compiler diagnoses five types of errors.

- **SEVERITY 0**: Advice
- **SEVERITY 1**: Warning
- **SEVERITY 2**: Correctable error
- **SEVERITY 3**: Uncorrectable error: translation can continue
- **SEVERITY 4**: Uncorrectable error: translation cannot continue

The text of the error message explains the cause of the error.

A severity-0 error, although valid PL/I, indicates that improvement is possible, usually in the area of performance. The source module is syntactically correct, so the compiled object module can be bound and executed, but probably with less than optimum efficiency.

A severity-1 error, although valid PL/I, is probably a programming error. Since the source module is syntactically correct at the point of a severity-1 error, however, the compiler continues to compile the source. The compiled object module can be bound and executed, but the program probably will not perform as expected.

A severity-2 error is invalid PL/I, but the compiler can reinterpret the source in such a way that it can continue to compile the program. The compiler proceeds as if the faulty code were replaced with the most likely syntactically correct code. The compiled object module can be bound and executed, but the program probably will not perform as expected.

A severity-3 error is invalid PL/I, and the compiler cannot reinterpret the source in such a way that it can continue to compile the program into a usable object module. Nevertheless, the compiler continues to process the program to detect additional errors. However, the object module is not created.

A severity-4 error is invalid PL/I, and the compiler cannot reinterpret the source in such a way that it can continue to process the program from the point of the severity-4 error. The object module is not created.

**Note:** If the compilation results in more than 100 errors, in any combination (excluding severity-0 errors), compilation terminates.

The compiler always overwrites an existing object module having the same name as the object module it produces.

**Access Requirements**

You need read access to the source module to compile it. You need modify access to the directory from which you are issuing the compile command, in which the .obj file will be created.

**Examples**

The command `pl1 sort_reports -xref -table -no_optimize -check` compiles the OpenVOS PL/I source module `sort_reports.pl1` in the current directory, producing
a compilation listing that includes a cross-reference listing and an OpenVOS Symbolic Debugger symbol table so that you can use the OpenVOS Symbolic Debugger in pl1 mode to debug the program. The compiler also inserts out-of-bounds array subscript-checking code. The name of the object module created is sort_reports.obj. It is put in the current directory. If the compiler finds any errors, it creates an error file named sort_reports.error and writes the error messages to it. The compiler puts the error file in the current directory.

Related Information

See the OpenVOS PL/I Language Manual (R009) for a complete description of the OpenVOS PL/I language.
**position_tape**

**Purpose**
This command positions the tape mounted on a specified tape drive, or on a tape drive to which a specified port is attached.

**Display Form**

```
------------ position_tape ---------------
  tape_device_or_port_name: [ ]
  -rewind: no
  -end_of_volume: no
-relative:
-absolute:
-file_id:
-file_number:

```

**Command Line Form**

```
position_tape tape_device_or_port_name
  [-rewind]
  [-end_of_volume]
  [-relative relative_number_of_files]
  [-absolute absolute_number_of_files]
  [-file_id file_id]
  [-file_number file_number]
```

**Arguments**

- **tape_device_or_port_name**  
  The name of a tape device or the name of a port attached to a tape drive, holding the tape to be positioned.

- **-rewind**  
  Rewinds the tape to the beginning of the volume before performing any other positioning. By default, `position_tape` does not rewind the tape.

- **-end_of_volume**  
  Positions the tape to the end of the volume so that it can append a new file to the volume. The tape is rewound first if you specify `-rewind`. Otherwise, the tape is positioned to the end of the volume before any other positioning operation. By default, `position_tape` does not position the tape to the end of the volume.
**position_tape**

- **-relative relative_number_of_files**
  Positions the tape a number of files before or after the current position. Specify the number of files with `relative_number_of_files`. If `relative_number_of_files` is positive, `position_tape` moves the tape forward that many files. If the value is negative, it moves the tape backward the given number of files. The command does this positioning after rewinding or positioning to the end of the volume if you specify either `-rewind` or `-end_of_volume`. The command leaves the tape at the beginning of a file in an unlabeled tape and at the beginning of a file label in a labeled tape. You can specify only one of the following arguments: `-relative`, `-absolute`, `-file_id`, and `-file_number`.

- **-absolute absolute_number_of_files**
  Positions the tape a number of files from the beginning of the tape. Specify the number of files with `absolute_number_of_files`. The command does this positioning after rewinding or positioning to the end of the volume if you specify either `-rewind` or `-end_of_volume`. It leaves the tape at the beginning of a file in an unlabeled tape and at the beginning of a file label in a labeled tape. You can specify only one of the following arguments: `-relative`, `-absolute`, `-file_id`, and `-file_number`.

- **-file_id file_id**
  Positions the tape to a given tape file, specified by its file ID. The tape file must be after the current position of the tape. The tape must be labeled. The command leaves the tape at the beginning of the label of the specified file. It does this positioning after rewinding or positioning to the end of the volume if you specify either `-rewind` or `-end_of_volume`. You can specify only one of the following arguments: `-relative`, `-absolute`, `-file_id`, and `-file_number`.

- **-file_number file_number**
  Positions the tape to a given file, specified by its file number. The tape must be labeled. The command leaves the tape at the beginning of the label of the specified file. It does this positioning after rewinding or positioning to the end of the volume if you specify either `-rewind` or `-end_of_volume`. You can specify only one of the following arguments: `-relative`, `-absolute`, `-file_id`, and `-file_number`.

**Explanation**

The `position_tape` command positions a mounted tape to the beginning or the end of the tape volume or to a given file on the tape volume.

To use the `position_tape` command, you must first explicitly attach a port and explicitly mount a tape. As a convenience, you can specify a tape device or a port name as a value for `tape_device_or_port_name`. The `position_tape` command does not implicitly attach a port or mount a tape. For more information, see the `Explanation` section in the `mount_tape` command description.

After you position the tape, you can issue a command that will perform another operation on the tape.

You can reposition reel tapes and 1/2-inch cartridge tapes. You can also overwrite selected portions of existing data on these tapes. You can only reposition 1/4-inch cartridge tapes to their beginning or end. You **cannot** reposition a 1/4-inch cartridge tape to a file or overwrite
selected portions of existing data on these tapes. You can overwrite all the existing data on a 1/4-inch tape, if you position the tape to its beginning.

Related Information
See also the command descriptions of copy_tape, create_tape_volumes, dismount_tape, display_tape_params, dump_tape, list_save_tape, mount_tape, read_tape, restore_object, set_second_tape, set_tape_drive_params, set_tape_mount_params, set_tape_file_params, verify_save, and write_tape.
posixpath

Purpose
This command converts a relative or full OpenVOS path name into a full POSIX path name.

Display Form
None.

Command Line Form
posixpath [path_name] --help --version

Arguments
- path_name
  The name of a file, or a relative or absolute path name.
- --help
  Displays a brief help message that describes the command.
- --version
  Displays a brief version string.

Explanation
The posixpath command converts the path_name argument, which is a relative or full OpenVOS path name, into a full POSIX path name. The resultant POSIX path name always begins with a slash character (/).

The three arguments are all mutually exclusive.

This command is primarily for use with interactive shells such as bash. Users who are experienced with OpenVOS commands may prefer to use the (posix_path) command function instead of the posixpath command.

Examples
The following command converts an OpenVOS path name into a POSIX path name.

    posix_path %es#mRaid4>system>gnu_library>bin
/system/gnu_library/bin
Related Information

See the description of the `vospath` command and the description of the `(posix_path)` command function.
**preprocess_file**

**Purpose**

This command is a stand-alone preprocessor for text files. The command creates an output text file by copying lines from an input text file into the output text file, based on preprocessor statements specified in the input text file.

**Display Form**

```
------------------------------- preprocess_file --------------------------------
source_file_name: [ ]
output_file_name: [ ]
-define:
```

**Command Line Form**

```
preprocess_file source_file_name
[ output_file_name ]
[ -define variable... ]
```

**Arguments**

- **source_file_name**  
  Required  
  The path name of the source file to be processed. When the source file is processed, an output file named `source_file_name.pout` is created in the current directory.

- **output_file_name**  
  The path name of an output file. The default path name is `current_directory>source_file_name.pout`. If you do not specify the suffix `.pout`, the preprocessor will add it to the output file automatically.

- **-define variable**  
  Defines variables to be used by the preprocessor. These variables are used by the preprocessor, not by the source file itself. Preprocessor variables can contain letters, digits, or underline characters (_), in any position.

**Explanation**

The `preprocess_file` command expands a source file named `source_file_name` into an output file named `source_file_name.pout`. Based on the preprocessor statements specified in the source file, `preprocess_file` copies lines from the source file into an output file that the command creates. You can use this command to preprocess any text file. Stratus recommends that you use the `c_preprocess` command to preprocess OpenVOS C
programs, and that you use the cpp command to preprocess OpenVOS Standard C programs. (See the c_preprocess and cpp command descriptions for more information.)

The output_file_name argument allows you to specify the name of the .pout output file to be created. By default, the file is placed in the current directory.

The -define argument defines variables to be used by the preprocessor. For example, to make the variables var_a and var_b in program1 available to the preprocessor, specify the following:

    preprocess_file program1 -define var_a var_b

Any variables specified with the -define command line argument are established as defined at the beginning of the compilation. All other variables are initially undefined. A source file can contain no more than 100 preprocessor variables.

Before you can compile, bind, and execute a source_file_name.pout source file, you need to rename it source_file_name.language. To avoid the possibility of using the wrong compiler with the preprocessed file, you may want to include the language extension, for example, .pl1, when specifying the source_file_name.

Caution: If you rename the .pout source file, do not give it the same name as the original source file, or the original will be overwritten.

The Preprocessor Statements
Preprocessor statements allow you to conditionally compile a source file. Conditional compilation enables you to switch on or off various statements in a source file. This is useful, for example, if you want a program to compile different lines of source code on different processors.

There are six preprocessor statements.

- $define
- $undefine
- $if
- $else
- $elseif
- $endif

Note that OpenVOS COBOL and assembly language programs require the @ lead-in character for statements, instead of the $ character. The preprocessor statements in OpenVOS COBOL and assembly language programs are @define, @undefine, @if, @else, @elseif, and @endif. Note that where this discussion refers to the $ character, the @ character is also implied.

Preprocessor statements must begin in the first column of the source file. Indentation of nested $if statements is, therefore, not allowed.

A preprocessor statement must be contained on a single line. A line containing a preprocessor statement cannot contain comments or parts of the source language. (An exception is the
$endif statement, which ignores any text following it on the same line, thus allowing you to comment on the source code.)

The following sections describe each preprocessor statement.

**The $define Statement**
The $define statement defines a preprocessor variable inside a source file. The syntax is as follows:

```
$define identifier
```

An identifier can contain letters, digits, or an underline character (_), in any position. For more information on defining preprocessor variables, see the description of the -define argument earlier in the Explanation section.

**The $undefine Statement**
The $undefine statement undefines a preprocessor variable. The syntax is as follows:

```
$undefine identifier
```

**The $if Statement**
The $if statement conditionally evaluates an expression as true or false. The syntax is as follows:

```
$if expression
```

Expressions are simple expressions using the NOT, AND, and OR operators. Parentheses have their usual grouping meaning. Table 2-28 lists the operators and lead-in character for each command.

**Table 2-28. Preprocessing Operators and Lead-In Characters**

<table>
<thead>
<tr>
<th>Command</th>
<th>AND</th>
<th>OR</th>
<th>NOT</th>
<th>Lead-In Character</th>
</tr>
</thead>
<tbody>
<tr>
<td>bind</td>
<td>&amp;</td>
<td></td>
<td></td>
<td>^</td>
</tr>
<tr>
<td>cobol</td>
<td>and</td>
<td>or</td>
<td>not</td>
<td>^</td>
</tr>
<tr>
<td>create_table</td>
<td>&amp;</td>
<td></td>
<td></td>
<td>^</td>
</tr>
<tr>
<td>fortran</td>
<td>.and.</td>
<td>.or.</td>
<td>.not.</td>
<td>$</td>
</tr>
<tr>
<td>pascal</td>
<td>and</td>
<td>or</td>
<td>not</td>
<td>^</td>
</tr>
<tr>
<td>pl1</td>
<td>&amp;</td>
<td></td>
<td></td>
<td>^</td>
</tr>
</tbody>
</table>

The order of operator precedence is NOT, AND, and OR.

In an expression, preprocessor variables that are defined evaluate to true. If the expression evaluates to true, all statements up to the next $else, $elseif, or $endif statement are processed, unless processing is explicitly disabled by another preprocessor statement; otherwise, they are skipped.

You cannot specify a preprocessor variable by itself. Instead, you must pass it as an argument to the defined function. There cannot be any spaces between the lead-in character and the
preprocessor statement. All other tokens must be separated by at least one space. See the Examples section for examples of the defined function.

The limit for nested $if and $elseif statements is 32.

**The $else Statement**
The $else statement conditionally processes the lines up to the next $endif statement. The syntax is as follows:

```plaintext
$else
```

**The $elseif Statement**
The $elseif statement conditionally evaluates an expression as true or false. The syntax is as follows:

```plaintext
$elseif expression
```

See “The $if Statement” for more information.

**The $endif Statement**
The $endif statement closes the most recent $if or $elseif statement. The syntax is as follows:

```plaintext
$endif [ignored_text]
```

An $endif statement is required for each $if and $elseif statement specified.

You can optionally place text after the $endif to comment the source code. The preprocessor ignores the text if it is preceded by a space.

**Differences between preprocess_file and Compiler Preprocessing**
This section explains the differences between the preprocessing performed by the preprocess_file command and the preprocessing performed by the compilers.

**Comments**
All of the compilers recognize comments in source files. The preprocess_file command, however, treats comments as if they were normal text. This difference can affect your results. For example, the following text would be treated as a comment (that is, ignored) by the PL/I compiler, but not by the preprocess_file command.

```plaintext
/ *
 $define abc
 */
```

The preprocess_file command would not recognize /* and */ as comment delimiters and would consider $define abc to be executable code. Therefore, you should be careful when commenting out code in source files that you plan to preprocess with the preprocess_file command.

See the descriptions of the compiler commands for examples of valid and invalid comments.
Output Files
Just as the .pout file created by `preprocess_file` contains only the lines of source code to be executed, the .list file created by the compiler commands also shows you which lines of code will be executed. If you specify the -list argument in a compiler command, the compiler inserts three plus signs (++) in front of each line in the .list file that will not be executed. (An asterisk, rather than three plus signs, is inserted in the .list files for the bind command.) For example, the plus signs in the following .list file designate which lines of code will not be executed if you specify the command `pl1 test -list -define a_code`. All unmarked lines will be executed. (Note that only the pertinent part of the .list file is shown, not the entire file.)

```
1        a:
2             procedure;
3        declare i fixed bin(15);
4
5   +++$if defined (a_code) | defined (b_code)
6             i = 1;
7   +++$else
8   +++          i = 2;
9   +++$endif
10
11        put list (i);
12        end a;
```

See the Examples section of this command description to compare the preceding output against the .pout file created when `preprocess_file` preprocesses the same PL/I program.

Access Requirements
You need read access to the source file to process it. You need modify access to the directory in which the .pout file will be created.

Examples
Example 1.
The following OpenVOS PL/I program, `test.pl1`, uses preprocessor statements to perform conditional compilation.

```
a:    
    procedure;
    declare i fixed bin(15);

$if defined (a_code) | defined (b_code)
    i = 1;
$else
    i = 2;
$endif

    put list (i);
end a;
```
If you specify the `preprocess_file test.pl1 -define a_code` command, the command produces the following file, `test.pl1.pout`.

```plaintext
a:
  procedure;
  declare i fixed bin(15);

  i = 1;

  put list (i);
end a;
```

The `.pout` file contains only the source code that will be compiled. Since `a_code` was defined, the line `i = 1;` will be compiled.

If you specify the `preprocess_file test.pl1` command without defining `a_code` or `b_code`, the line `i = 2;` is processed rather than `i = 1;`.

Example 2.
The following OpenVOS COBOL program, `test.cobol`, uses preprocessor statements to perform conditional compilation.

```plaintext
data division.
  working-storage section.
  @if defined (use_i)
    01 i pic 9.
  @elseif defined (use_j)
    01 j pic 9.
  @endif

  procedure division.
  main.
  @if defined (use_i)
    move 1 to i.
    display i.
  @elseif defined (use_j)
    move 2 to j.
    display j.
  @endif
  stop run.
```

If you specify the command `preprocess_file test.cobol` without defining `use_i` or `use_j`, the command produces the following file, `test.cobol.pout`.

```plaintext
data division.
  working-storage section.

  procedure division.
  main.

  stop run.
```
Since neither preprocessor variable was defined, the .pout file contains no executable code. As a result of preprocessing, during compilation, the compiler will not allocate any storage for the variables i and j, and will not execute any move or display statements. In this way, preprocessor statements can save system resources.

**Error Codes**

The following table explains some error codes this command might return.

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>e$pp_too_many_variables (4693)</td>
<td>Too many variables were defined. The limit is 100 variables.</td>
</tr>
<tr>
<td>e$pp_preprocess_failed (4694)</td>
<td>An error occurred during preprocessing.</td>
</tr>
<tr>
<td>e$pp_control_line_error (4695)</td>
<td>An error was encountered. Typically, this error indicates a misspelled control name.</td>
</tr>
<tr>
<td>e$pp_syntax_errors_in_expression (4699)</td>
<td>An expression in an $if or $elseif statement could not be parsed successfully.</td>
</tr>
<tr>
<td>e$pp_invalid_variable_name (4698)</td>
<td>An invalid variable name was encountered in a $define or $undefine statement or within a defined() function reference.</td>
</tr>
<tr>
<td>e$pp_expression_stack_overflow (4700)</td>
<td>Too many nested $if statements were specified. The nesting limit is 32.</td>
</tr>
<tr>
<td>e$pp_expression_stack_underflow (4701)</td>
<td>An unexpected $endif statement was encountered (no unclosed $if statement preceded the $endif statement).</td>
</tr>
<tr>
<td>e$pp_unclosed_if (4702)</td>
<td>The source program ended with an unclosed $if statement.</td>
</tr>
<tr>
<td>e$pp_unexpected_elseif (4703)</td>
<td>An $elseif statement was encountered, but no $if statement was currently active.</td>
</tr>
<tr>
<td>e$pp_unexpected_else (4704)</td>
<td>An $else statement was encountered, but no $if statement was currently active.</td>
</tr>
<tr>
<td>e$pp_unexpected_end_expression (4705)</td>
<td>An $endif statement was encountered, but no $if statement was currently active.</td>
</tr>
<tr>
<td>e$pp_missing_left_paren (4696)</td>
<td>An expression, $define statement, or $undefine statement is missing a left parenthesis.</td>
</tr>
<tr>
<td>e$pp_missing_right_paren (4697)</td>
<td>An expression, $define statement, or $undefine statement is missing a right parenthesis.</td>
</tr>
<tr>
<td>Error Code</td>
<td>Explanation</td>
</tr>
<tr>
<td>------------------------------------------------</td>
<td>-------------------------------------------------------</td>
</tr>
<tr>
<td>e$pp_no_value_defined(4706)</td>
<td>A preprocessor variable is expected to be defined.</td>
</tr>
<tr>
<td>e$pp_unexpected_syntax(4707)</td>
<td>The syntax is unexpected; the word defined is expected.</td>
</tr>
<tr>
<td>e$pp_too_many_tokens(4708)</td>
<td>An expression exceeds the limit of 100 tokens.</td>
</tr>
<tr>
<td>e$pp_left_over_tokens(4709)</td>
<td>An expression has been specified incorrectly.</td>
</tr>
</tbody>
</table>

**Related Information**

See also the command descriptions of `c_preprocess`, `cpp`, `vcpp`, `cobol`, `fortran`, `pascal`, and `pl1`. For a description of the `create_table` command, see *OpenVOS System Administration: Configuring a System* (R287).
Purpose
This command puts a print request into a print queue for printing.

Display Form

```
file_names:   print
-queue:       standard
-title:
-destination:
-module:
-device:
-header:
-footer:
-index:
-defer_until:
-interpret_tabs:  replace
-exception_handling:  replace
-copies:       1  -line_numbers:  no
-delete:       no  -raw:  no
-page_breaks:  yes  -use_fortran_controls:  no
-indentation:  0  -page_size:
-top_margin:   3  -bottom_margin:  3
-line_length:  
-queue_priority:
-first_page:   1  -notify:  no
-pass_thru:    no  -last_page:  0
```

Command Line Form

```
print file_names...
  [-queue queue_name]
  [-title title_string]
  [-destination destination_string]
  [-module module_name]
  [-device device_name]
  [-header header_string]
  [-footer footer_string]
  [-index index_name]
  [-defer_until date_time]
  [-interpret_tabs start_column, spacing]
  [-exception_handling exception_string]
  [-copies number]
  [-line_numbers]
  [-delete]
  [-raw]
  [-no_page_breaks]
  [-use_fortran_controls]
  [-indentation number]
  [-page_size number]
  [-top_margin number]
  [-bottom_margin number]
  [-line_length number]
  [-wrap]
  [-queue_priority number]
  [-notify]
  [-first_page page_number]
  [-last_page page_number]
  [-pass_thru]
```

Arguments

- **file_names**  
  One or more names or star names of files to be printed. You can specify any type of file.

- **-queue queue_name**  
  Prints the file on the printer controlled by the print queue *queue_name*. By default, `print` puts the file into the *standard* queue. The *standard* queue is usually associated with a default printer on which the file is printed, but this depends on the site configuration.

- **-title title_string**  
  Prints the character string *title_string* on the header page that precedes the printout of your file. In the command line, if the string contains spaces, you must enclose it in apostrophes. By default, the file name appears as the title on the header page. The *title_string* value has a maximum length of 132 characters.
-destination destination_string

Prints the character string destination_string at the top of the header page that precedes the printout of your file. In the command line, if the string contains spaces, you must enclose it in apostrophes. By default, your person name appears as the destination on the header page.

-module module_name

Specifies the module containing the specified print queue. By default, print uses your current module.

-device device_name

Specifies a device name or star name. If you specify a device, the request is printed only on a device that matches the specified name or star name. If you do not specify a device, the request is printed on the first available printer controlled by the specified queue. If you specify a device that is not available, the request remains in the queue until the specified device becomes available. If you specify a device name unknown to the operating system, the following error message is displayed:

Device name is not known to the system.

-header header_string

Prints the character string header_string on every page of the output. For top margins of three or more lines, header_string appears at the left margin on the second line; for top margins of one or two lines, header_string appears on the first line. By default, no header is printed. If you print a file that incorporates formatting features of the edit command, or if you specify -no_page_breaks, print ignores this argument. The header_string value has a maximum length of 132 characters.

-footer footer_string

Prints the character string footer_string on every page of the output. For bottom margins of three or more lines, footer_string appears at the left margin on the second line from the bottom; for bottom margins of one or two lines, footer_string appears on the bottom line. By default, no footer is printed. If you print a file that incorporates formatting features of the edit command, or if you specify -no_page_breaks, print ignores this argument. The footer_string value has a maximum length of 132 characters.

-index index_name

Specifies an index that controls the order in which the records in a specified file are printed. By default, the records are printed in the order in which they appear in the file.

-defer_until date_time

Defers printing the file until some time after date_time. The date_time value can be a character string in the standard form.

yy-mm-dd hh:mm:ss

It can also be a character string in any form accepted by the (date_time) command function. In this case, the string must be enclosed in apostrophes. See Chapter 1, “OpenVOS Command Functions,” for examples of acceptable date/time input strings.
- **-interpret_tabs** `start_column, spacing`  
Interprets occurrences of the ASCII tab character. You must specify the column number `start_column` of the first tab stop and the number `spacing` of positions between tab stops. A comma must separate the two numbers. You cannot specify `-interpret_tabs` and `-raw` in the same command.

- **-exception_handling** `exception_string`  
Specifies how to handle nonprinting characters in the text. The possible values are `replace`, `ignore`, and `abort`. If you specify `replace`, the operating system prints the hexadecimal number representing the ASCII code for the nonprinting character. If you specify `ignore`, the operating system ignores nonprinting characters. If you specify `abort`, the operating system cancels this print request if a nonprinting character is encountered, and continues with the next request in the queue. By default, the `print` command uses `replace`. You cannot specify `ignore` and `-raw` in the same command.

- **-copies** `number`  
Prints multiple copies of each specified file. By default, the command prints one copy.

- **-line_numbers**  
Prints the file with line numbers. By default, the command prints the file without line numbers.

- **-delete**  
Deletes the file after it is printed.

- **-raw**  
Prints the file literally; all character sequences that are normally control sequences for the printer (and not printed) are replaced with the ASCII digits representing the hexadecimal value of the bytes. You cannot specify `-raw` and either `-exception_handling ignore` or `-interpret_tabs` in the same command.

- **-no_page_breaks**  
Prints the file without page breaks and automatically sets the top and bottom margins to 0. By default, files are printed with page breaks.

- **-use_fortran_controls**  
Interprets any of the following characters as a FORTRAN printing control character when it appears in column 1 of a file. The command treats all other characters as space characters.

<table>
<thead>
<tr>
<th>Character</th>
<th>Printing Instruction</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Skip to the next page</td>
</tr>
<tr>
<td>0</td>
<td>Double space</td>
</tr>
<tr>
<td>+ †</td>
<td>Overstrike the previously written record</td>
</tr>
<tr>
<td>space character</td>
<td>Skip to the next line</td>
</tr>
</tbody>
</table>

†The command does not support the - (minus) FORTRAN printing control character.
The command disregards any generic/canonical control sequences it encounters in the first column. (Generic/canonical control sequences are those that a user has entered in the body of text of a file.)

- **-indentation number**
  Sets the left margin for the body of the document as well as the headers and footers to the column designated by *number*. By default, `print` sets the left margin to the first printing position on the line.

- **-page_size number**
  Sets the number of lines on a page. After printing *number* lines, including top and bottom margin lines, the printer skips to a new page. By default, the page size default value for the print queue is used. The `-page_size` value can be a maximum of 254 lines long.

- **-top_margin number**
  Sets the number of lines in the top margin of each printed page. By default, the top margin value is 3. The first line of the file that appears on each page is printed on the first line after the top margin (line 4). If you print a file that incorporates formatting features of the `edit` command, or if you specify `-no_page_breaks`, `print` ignores this argument. The `-top_margin` value can contain a maximum of 254 lines.

- **-bottom_margin number**
  Sets the number of lines in the bottom margin of each printed page. By default, the bottom margin value is 3. The last line of the file that appears on each page is printed on the line immediately before the bottom margin. If you print a file that incorporates formatting features of the `edit` command, or if you specify `-no_page_breaks`, `print` ignores this argument. The `-bottom_margin` value can contain a maximum of 254 lines.

- **-line_length number**
  Specifies the number of character positions per line. The line length includes any indentation. By default, the line length is set to the default value for the print queue.

- **-wrap**
  Continues the printing of overflow from a long line on a subsequent line or lines. Overflow consists of the character or characters at the end of a line that will not fit into the positions available within the specified margins. By default, long lines are truncated.

- **-queue_priority number**
  Sets the print request’s priority in the print queue. The value of `-queue_priority` can be from 0 to 9, with 9 representing the highest queue priority. If you assign a queue priority to a print request, `print` inserts the request in the queue before all requests with lower queue priority. By default, the queue priority of a print request with a size less than or equal to 20 disk blocks is 5, and that of a print request with a size greater than 20 disk blocks is 4.
- **-notify**
  Tells the `print` command to send you a message when the printing of your job is complete. By default, the command does not notify you when your print job is complete.

- **-first_page page_number**
  Specifies that the request should start printing at the given page. All pages prior to the specified page are ignored.

- **-last_page page_number**
  Specifies that the request should stop printing at the given page. All pages after the specified page are ignored.

- **-pass_thru**
  Passes any new line control codes embedded in the file through to the printer. The command appends the new line control code to each record in the sequential file. You can use `-pass_thru` and `-device` to ensure that the file is printed on the device for which it was generated. The new line control codes are defined by your system administrator in the `spooler_configuration.v1.tin` file. For more information on this file, see the manual *VOS System Administration: Administering the Spooler Facility* (R286).

**Explanation**

The `print` command puts a print request into a print queue for printing on a line printer or a letter-quality printer. A print request includes the path name of a file or the path names of several files to be printed.

You can print any type of file with the `print` command.

At the beginning of a print job, the printer produces header pages that give information about the printout that follows. If you specify `-title` or `-destination`, you can specify an alternative title or destination for the printout of your file. Since `title_string` and `destination_name` can be any character string, you can specify any title and person, office, location, or other entity that you want. This is particularly useful if you are remotely situated from the printer that produces your printout.

When the file is a formatted file produced by the word processing editor, the `print` command sets the page size. Other text formatting is dependent on the formatting controls in the file.

Use `-top_margin` and `-bottom_margin` to specify margins of less than or more than 3 lines. With a margin of 1 or 2, if you also supply a header or footer string, the header or footer is printed on the first line of the margin. If you specify a margin of 0, no header or footer is printed. A margin of 0 is useful for printing unbroken output, such as address labels. The header and footer strings can contain the following variables, which the operating system replaces with current values when it prints the file:

- `&file_name&` (the path name of the file)
- `&page_number&` (the page number of the printed file)
- `&date_time&` (the current date and time)
- `&user_name&` (the user name of the requester)
print

The page-number variable always begins at 1 for each copy, and is incremented each time a page is printed. You can restrict the portion of a file the system prints by giving the -first_page and/or -last_page arguments to the command. To begin the print job somewhere other than the beginning of a file, use the -first_page argument, and all pages prior to the specified page are not printed. Similarly, to cause the system to stop printing before the end of a file, use the -last_page argument, and all pages after the specified page are not printed.

The date-and-time variable is the date and time that the operating system starts to print the file. A separate print job is started for each file you want printed.

If you specify -header, -footer, top_margin, or bottom_margin when printing a file that incorporates formatting features of the edit command, the operating system disregards your choices. Some combinations of print command arguments may yield unpredictable results when you use them on formatted files (for example, the -wrap and -line_numbers arguments).

You can print a file on another system to which you have access if:

- the queue and the file are on the same system
- the requesting process and the file are on the same system
- the system that the file is on does not require that users of your system be registered

Note that on a Remote 3270 printer you should not use the -pass_thru option to print a file including control characters (for example: LF or CR). Otherwise, a spooler will go offline due to a command reject operation.

Access Requirements

You need read access to a file in order to print it. You need modify access to the directory that contains the file in order to specify -delete.

Examples

Example 1.
To print two copies of the file make_reports.list in the current directory and delete the file after it is printed, use the following command.

```
print make_reports.list -delete -copies 2
```

Example 2.
The following command prints the specified files after 11 P.M.

```
print *.cobol *.list -defer_until 23:00:00
```

Example 3.
To print the file memos in the current directory with the specified header, use this command.

```
print memos -header 'The memos file of &user_name&.'
```

The name of the user making the request is filled in when the file is printed.
Example 4.
To print five pages from the middle of the file sales_reports, use this command.

```
print sales_reports -first_page 5 -last_page 10
```

Related Information
The `list_print_requests` command displays the status of all print requests you have made with the `print` command. The `cancel_print_requests` command cancels one or more print requests. The `display_print_status` command provides information about the status of the available spoolers. The `display_print_defaults` command describes the print defaults in effect for a specified queue. The `list_devices` `-type printer` command shows a list of available printers.
profile

Purpose
This command generates a file that contains performance information about the statement execution of one or more program modules.

Display Form

```
----------------------- profile -----------------------
profile_file_names: [value]
-pm_paths: [value]
-list: no
-sort: [value]
-wrap: yes
-module_name: [value]
-first_line: 1
-last_line: [value]
-threshold: 100
-includes: yes
-changes_only: no
-output_path: [value]
```

Command Line Form

```
profile profile_file_name... 
[ -pm_paths new_path_name...] 
[ -list ] 
[ -sort sort_code ] 
[ -no_wrap ] 
[ -module_name object_module_name ] 
[ -first_line number ] 
[ -last_line number ] 
[ -threshold number ] 
[ -no_includes ] 
[ -changes_only ] 
[ -output_path path_name ]
```
Arguments

- **profile_file_names**
  The path name of one or more profile files. You cannot use star names. If the system automatically generates a profile file, the file name has the suffix `.process_id.profile` (process_id is the program’s process ID, or PID). If you manually create a profile file, the file name usually has the suffix `.profile`. Program modules can have either a fixed or stream file organization. You must have compiled at least one object module in each program module and specified the `-ql` or `-qc` option (for OpenVOS Standard C programs) or the `-profile` or `-cpu_profile` argument (for programs compiled with one of the other OpenVOS compilers). To include alignment fault instead of page fault information, you must have bound the object module(s) with the `-profile_alignment_faults` argument of the bind command.

  **Note:** When the system automatically generates a profile file, it creates a new profile file each time, rather than overwriting the old files. To prevent the disk from filling up with old profile files, a system administrator should be aware of this issue and develop a plan for deleting these files.

  If you specify more than one `.profile` name, you must also give the `-output_path` argument.

- **-pm_paths new_path_name...**
  Specifies the new path name of one or more program modules used to create the profile files, if you have moved one or more program modules to another directory. You cannot use star names to specify the path names.

  If you specify this argument, you must provide the same number of program module paths as there are profile file names, even if you moved only one program module to another directory. Also, you must list them in the same order as the profile file names.

- **-list**
  Combines the performance information with a source listing in a side-by-side format. By default, the command combines only the statement line numbers with the performance data from the `.profile` file in the output. The executing program produces this `.profile` file.

  **Note:** You cannot specify `-list` and `-sort` at the same time.

- **-sort sort_code**
  Sorts the output by frequency, by cost in CPU time, or by coverage of statement execution. There are five possible values for `sort_code`.

  - count
  - cpu
  - faults
  - coverage
  - coverage_summary

  Specify `count` to select frequency, `cpu` to select CPU execution time, or `faults` to select page faults or alignment faults. Specify `coverage` or `coverage_summary` to
display information about the number and percentage of statements that actually executed.

If you do not specify sort_code, profile sorts the source lines and performance data in the order of the statements in the source module. If the object module was compiled with -profile specified, you can only sort by count.

**Note:** You cannot specify -sort and -list at the same time.

- **-no_wrap**
  Truncates to 79 characters any lines in the output file that exceed 79 characters. By default, no lines are truncated, and if output is written to a device only 80 columns wide, any lines longer than 80 characters wrap.

- **-module_name object_module_name**
  Specifies an object module that is one of several in a program module. The command reports on the performance of that one object module within the program module. You cannot supply the .obj suffix when specifying an object_module_name.

- **-first_line number**
  Specifies a line number designating the first line of a section of source code. By default, the value is 1. If you do not specify object_module_name, -first_line is ignored.

- **-last_line number**
  Specifies a line number designating the last line of a section of source code. If you do not specify object_module_name, -last_line is ignored.

- **-threshold number**
  Specifies the value for the CUM % column that determines when profile output is stopped. The threshold argument can have a value between 1 and 100; the default is 100. The value of threshold is meaningful only when the output includes a CUM % column, that is, when sort_code is cpu, count, or faults, and the program is compiled with -cpu_profile specified.

  For example, if you set threshold to 90 and specify -sort cpu, the command output stops when the CUM % column reaches 90 percent.

- **-no_includes**
  Prevents the expansion of include files when you specify -list. By default, if you specify -list, all included files will be expanded. If you specify -no_includes and executable code exists in the include files, the profile information for the executable code will not be shown.

- **-changes_only**
  Specifies that changed source lines only are used to compute the following values in the .plist file:
  - number of statements executed
  - number of statements not executed
  - total number of statements
If you do not specify this argument, unchanged source lines are added when these values are computed.

**Note:** You cannot specify this argument unless you specify the `coverage` or `coverage_summary` value in the `-sort` argument.

```bash
-> output_path path_name
```

Specifies the path name of the output file if you specify more than one profile file name. You **must** specify a path name if you specify more than one profile file name. The command builds the output file name by adding `.plist` to `path_name`.

If you specify only one profile file name, by default the command creates an output file in the current directory and builds the output file name by replacing the `.profile` suffix of the profile file with `.plist`.

**Explanation**

The `profile` command generates a file that contains performance information about the statement execution of one or more object modules in one or more program modules. The command uses as input one or more profile files created by executing program modules that you have compiled with certain arguments or options. The command operates in one of four modes, and generates a `.plist` output file, which you can display with a text editor or the `display` command.

**Preparing to Use the profile Command**

Before you can use the `profile` command, you must perform the following steps.

1. Compile one or more object module(s) and specify the `-profile` or `-cpu_profile` argument, or specify the `-ql` or `-qc` option if you are using the OpenVOS Standard C compiler.

When you specify `-profile` or `-ql`, the compiler inserts code to count the number of times each source statement is executed when the program runs. The compiler also marks the object module so that the object module produces one profile file every time it runs. If the operating system spawns more than one task containing the object module, only the **first** task generates its own profile file.

If you compile a source module with `-profile` or `-ql`, the `profile` command output contains the statement numbers and the number of times the statement was executed.

When you specify `-cpu_profile` or `-qc`, the compiler inserts code to obtain the amount of CPU time spent executing each statement, the number of page or alignment faults taken for each statement, and the number of times each statement executes. The compiler also marks the object module so that the object module produces one profile file every time it runs. If the operating system spawns more than one task containing the object module, all of the execution information is stored in **one** profile file.

If you compile a source module with `-cpu_profile` or `-qc`, the `profile` command output contains the statement numbers, the number of times the statement was executed, the CPU time (in milliseconds) spent executing the statement, and the
number of page or alignment faults taken while the statement was being executed. For more information on profiling alignment faults, see the section “Using the -profile_alignment_faults Compiler Argument.”

2. Bind these object modules into one or more program modules. If you specify the -profile_alignment_faults binder argument on an ftServer module, the profile command determines the number of alignment faults instead of the number of page faults.

3. Execute the program module(s). When you execute the program module, it produces a profile file. A profile file contains performance information about all of the object modules compiled with -profile or -cpu_profile and bound together in one or more executing program modules. If the system automatically generates a profile file, the file name has the suffix .process_id.profile (process_id is the program’s process ID, or PID). If you manually create a profile file, the file name usually has the suffix .profile. When you run the program, the operating system puts the profile file in your current directory and overwrites existing profile files with the same name.

4. If you plan to use the profile command in differential mode, you must run the compare_files command using the current version of the source file and an older version. See the following section, “Modes of Operation of the profile Command,” for more information about differential mode.

Modes of Operation of the profile Command
As described in the following sections, you can use the profile command in the following modes:

- uncombined, non-differential mode
- combined mode
- differential mode
- combined and differential mode

Uncombined, Non-differential Mode
Use this mode to examine statement execution of one or more object modules in one program module. To use this mode, specify one value for profile_file_names. When you specify only one value for profile_file_names, by default, the command creates an output file with the extension .plist in the current directory.

Combined Mode
Use this mode to merge code coverage information about object modules that are common to more than one program module. This information may be helpful if a single program module does not execute every line in an object module. To use this mode, specify two or more profile_file_names (including their path names) of program modules that contain common object modules. When you specify two or more values for profile_file_names, you must also specify a value for the -output_path argument that contains an output file name with the extension .plist. If you do not add the .plist extension, the command adds it to the output path name.

Note: Common object modules have the same name and same compile date and time. You can check the compile date and time of an object module used by a program
module by issuing the `display_program_module -module_map` command. You can also check the compile date and time of an object module by issuing the `display_object_module_info` command.

In combined mode, the file `output_path.plist` contains per-line execution counts for all object modules unique to any of the specified program modules. Also, for all object modules common to two or more of the specified program modules, it contains combined per-line execution counts.

**Differential Mode**

Use this mode to directly measure such data as the percentage of code coverage for specific numbers of lines of changed code in a program module.

**Note:** Before using the `profile` command in differential mode, you must have the profile file generated by the program module, the source file or files for each object module in the program module, and a comparison file for a current and older version for all object files being profiled.

Create the comparison files with the `compare_files` command. Specify the current source file as `path_B`. Specify that the comparison file created by `-output_path` be in the same directory as the current version of the source file, and that the comparison file have the extension `.cmpf`, as shown in the following examples. Note that if a source file name is longer than 32 characters with the `.cmpf` extension, you must truncate it to accommodate the `.cmpf` extension.

<table>
<thead>
<tr>
<th>Source File Name</th>
<th>Name of compare_files Output</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>object_module.x.pl1</code></td>
<td><code>object_module_x.pl1.cmpf</code></td>
</tr>
<tr>
<td><code>setalignment_fault_mode.pascal</code></td>
<td><code>setalignment_fault_mode.pascal.cmpf</code></td>
</tr>
</tbody>
</table>

When you specify the `-changes_only` argument and the `coverage` or `coverage_summary` value of the `-sort` argument, the `profile` command performs differential profiling. In this case, you do not specify the comparison file name. Instead, the command assumes that the comparison file(s) follow the previously described naming conventions. The command checks that each pair of source files and comparison files has a consistent name and that the comparison file has a later creation date than the source file.

If a properly named comparison file does not exist in the same directory as the current version of a source file, the command displays the message:

```
profile (advisory): error_code. Continuing...
```

and performs non-differential profiling for that object module. The `error_code` may indicate lack of access to the comparison file, a circular link, a module being off-line, or one of many other possible problems. If the `profile` command encounters a zero-length comparison file, the command assumes that there are no source changes.

In this mode, the command writes performance information only about lines that are different in the current and older versions of the source files lines to the `.plist` file.
**Combined and Differential Mode**
Use the combined and differential modes together when you want to examine the execution history of changed lines in an object module or modules that is included in more than one program module. Before using the `profile` command in combined and differential mode, issue the `compare_files` command to generate a comparison file for the current and older versions of the object modules’ source code. To issue the `profile` command in combined and differential mode, specify `profile_file_names` of the program modules that include the common object module(s), and specify the `-changes_only` argument and the `coverage` or `coverage_summary` value of the `-sort` argument.

**Using the `-pm_paths` Argument**
If you specify the `-pm_paths` argument, exactly the same number of `pm` paths must exist as there are profile file names, and the order that you specify the `pm` paths must correspond to the order in which you specify the profile file names. In addition, when you issue the `profile` command, the program module must be the same version as the executing program (not a re-bound version).

**Controlling the Output of the `profile` Command**
As described in the following sections, you can control the output of the `profile` command in any mode by specifying the `-profile` or `-cpu_profile` compiler arguments, and by specifying one or more of the following `profile` command arguments:

- `-list`
- `-no_includes`
- `-sort`
- `-module_name`
- `-first_line`
- `-last_line`

**Using the `-profile` or `-cpu_profile` Compiler Argument**
If you compile the source module with `-profile` (or `-ql`) compiler argument, the `profile` command lists the statement numbers and the number of times the statement was executed.

If you compile the source module with `-cpu_profile` or `(-qc)` compiler argument, the command lists the statement numbers, the number of times the statement was executed, and the CPU time (in milliseconds) spent executing the statement. The command also lists the number of page faults taken while the statement was being executed unless you are using an ftServer module and you specify the `-profile_alignment_faults` binder argument after compiling the object modules, in which case the command lists the alignment faults taken while the statement was being executed.

**Note:** If a program module contains a call to a subroutine, the `.plist` entry for the calling statement includes the subroutine only if the subroutine is not itself profiled.

**Using the `-profile_alignment_faults` Compiler Argument**
If you bind the object modules with the `-profile_alignment_faults` argument on an ftServer module, the `profile` command lists the number of alignment faults (instead of page faults) taken while the statement was being executed.
Using the -list or -no_includes Argument
To display the code for each statement number in the .plist file, give the -list argument to the command. You can also use the -no_includes argument to suppress the display of the header files in the .plist file. If you specify -list, the command adds the actual source code next to the execution history for line number in the .plist file. The profile command can only find the source module if it is in the same directory in which it was compiled. The source module must be the same one that was used to compile the program module.

Using the -sort Argument
You can specify one of five values for the -sort argument to control how and what type of execution history information is sorted in the .plist file. Note that the -sort argument only examines executable statements; it ignores comments and blank lines. Note also that the -list and -sort arguments are mutually exclusive.

- If you specify -sort count, the command sorts the execution history of each source statement in order of decreasing frequency of statement execution.

- If you specify -sort cpu, the command sorts the execution history of each source statement in order of decreasing CPU time consumed.

- If you specify -sort faults, the command sorts the execution history of each source statement in order of decreasing numbers of page or alignment faults taken during the execution of each statement.

- If you specify -sort coverage, the command lists the following information.
  - the number of statements in the program
  - the number and percentage of statements that executed
  - the line numbers of the statements that are not executed
  - the number of statements contained on those lines

- If you specify -sort coverage_summary, the command lists the following information.
  - the number of statements in the program
  - the number and percentage of statements executed for each module
  - the total number and percentage of statements that executed

Using the -module_name Argument
If you specify a value for the -module_name argument, performance information is displayed for the object module identified by object_module_name rather than for the entire program module in which the specified object module is contained.
Using the **-first_line** or **-last_line** Argument

If you specify a value for the **-first_line** argument, performance information is displayed for source statements starting with the line identified by the **-first_line number** value. If you specify a value for the **-last_line** argument, performance information is collected for source statements up to the line identified by the **-last_line number** value. The **-first_line** and **-last_line** arguments only apply if you specify the **-module_name** argument.

Using the **-threshold** Argument

If you specify a value for the **-threshold** argument, you can reduce the size of the .plist file by eliminating statements you do not need to include. For example, if you specify a value of 90 for **threshold**, the listing continues until the **CUM %** column reaches 90 percent.

Access Requirements

To use the **profile** command, you need modify access to the current directory. You also need read access to the files you specify in the **profile_file_names** argument and to the program module used to create the profile file.

Examples

**Example 1.**

This example uses the **profile** command in non-differential, non-combined mode to examine the execution history of the following **sample.pl1** source module. This source module converts a set of Celsius temperatures to Fahrenheit.

```plaintext
1 sample:
2   procedure;
3
4   declare   fahrenheit float bin(24);
5   declare   celsius    float bin(24);
6   declare   degrees    fixed bin(15);
7
8   put edit ('   CELSIUS', ' FAHRENHEIT')
9       (a (10), x (1), a (11));
10   put skip;
11
12   do degrees = 0 to 100 by 10;
13       celsius = degrees;
14       fahrenheit = 9 * celsius / 5 + 32;
15   put edit (celsius, fahrenheit)
16       (f (10), x (1), f (11, 1));
17   put skip;
18 end;
19
20 end sample;
```
Perform the following steps to examine the execution history of the `sample.pl1` source module.

1. Compile the `sample.pl1` source code using the `-profile` or `-cpu_profile` option.
   
   pl1 sample -profile

2. Bind the sample object module.
   
   bind sample

3. Execute the `sample.pm` program module. When you execute the `sample.pm` program module, it generates a `sample.profile` file that contains the execution history for the Celsius to Fahrenheit conversion.
   
   sample

<table>
<thead>
<tr>
<th>CELSIUS</th>
<th>FAHRENHEIT</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>32.0</td>
</tr>
<tr>
<td>10</td>
<td>50.0</td>
</tr>
<tr>
<td>20</td>
<td>68.0</td>
</tr>
<tr>
<td>30</td>
<td>86.0</td>
</tr>
<tr>
<td>40</td>
<td>104.0</td>
</tr>
<tr>
<td>50</td>
<td>122.0</td>
</tr>
<tr>
<td>60</td>
<td>140.0</td>
</tr>
<tr>
<td>70</td>
<td>158.0</td>
</tr>
<tr>
<td>80</td>
<td>176.0</td>
</tr>
<tr>
<td>90</td>
<td>194.0</td>
</tr>
<tr>
<td>100</td>
<td>212.0</td>
</tr>
</tbody>
</table>

4. Issue the `profile` command to generate the `sample.plist` file from the `sample.profile` file.
   
   profile sample.profile
5. You would then issue the `display_file` command to display the `sample.plist` file. This file shows the frequency of execution of statements 1 through 20.

```
display_file sample.plist
```

Profile of: sample

Number of statements: 10
Statements Executed: 10 (100.00% of statements)

<table>
<thead>
<tr>
<th>STATEMENT</th>
<th>COUNT</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>8</td>
<td>1</td>
</tr>
<tr>
<td>10</td>
<td>1</td>
</tr>
<tr>
<td>12</td>
<td>1</td>
</tr>
<tr>
<td>13</td>
<td>11</td>
</tr>
<tr>
<td>14</td>
<td>11</td>
</tr>
<tr>
<td>15</td>
<td>11</td>
</tr>
<tr>
<td>17</td>
<td>11</td>
</tr>
<tr>
<td>18</td>
<td>11</td>
</tr>
<tr>
<td>20</td>
<td>1</td>
</tr>
</tbody>
</table>

TOTALS: 60

Example 2.

The following are examples of the difference between non-differential and differential output. Non-differential output shows the total number of statements in a program module and the total number of statements executed in a program module.

Coverage of: sample_new

Number of statements: 17
Statements Executed: 12 (70.58% of statements)

<table>
<thead>
<tr>
<th>STATEMENTS NOT EXECUTED</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>LINE</th>
<th>NUMBER OF NUMBER STATEMENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>24</td>
<td>30</td>
</tr>
</tbody>
</table>
Differential output shows the total number of **changed** statements in a program module and the total number of changed statements executed in a program module.

Coverage of : sample_new

- Number of statements: 10 (changed statements only)
- Statements Executed: 5 (50.00% of changed statements)

**CHANGED STATEMENTS NOT EXECUTED**

<table>
<thead>
<tr>
<th>LINE</th>
<th>NUMBER OF STATEMENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>24 - 30</td>
<td>5</td>
</tr>
</tbody>
</table>

**Related Information**

For information about accumulating profile data, see the description of the `add_profile` command. For information about comparing old and current versions of a file, see the `compare_files` command. For information on using the program counter to measure performance, see the description of the `harvest_pc_samples` and `analyze_pc_samples` commands.
propagate_access

Purpose
This command copies entries in the access control list and the default access control list of a directory to all subordinate directories.

Display Form

```
-------------------------- propagate_access --------------------------
directory_name: current_directory
-exclude:
-user:
```

Command Line Form

```
propagate_access [directory_name]

  [-exclude subdirectory_names...]
  [-user [user_name]]
```

Arguments

- `directory_name`
The name of a directory. The command replaces entries in the access control lists and default access control lists of subordinate directories with entries in the access control list and default access control list of the directory `directory_name`. By default, the command uses the current directory.

- `-exclude subdirectory_names`
  Specifies the names or star names of subordinate directories to be excluded from this command’s actions. The command does not replace entries in the access control lists and default access control lists of these directories. By default, the command uses all subordinate directories.

- `-user user_name`
  Propagates only one access control list entry or one default access control list entry. The `user_name` value can be a user star name. If you specify `-user` but omit a value for `user_name`, `propagate_access` uses your user name. By default, the command completely replaces the access control lists and default access control lists of the subordinate directories with the access control list and default access control list of the directory `directory_name`. 
**Explanation**

The `propagate_access` command adds or replaces entries in the access control lists and default access control lists of specified directories that are subordinate to the directory `directory_name`.

When you create a directory, the directory inherits the access control list and the default access control list of the containing directory. But the operating system does not update the access control lists and default access control lists of subordinate directories when you subsequently change the access control list or default access control list of a containing directory. You can use the `propagate_access` command to update these lists so they match the new lists on the superior directory.

**Access Requirements**

To propagate the access control list and default access control list of a directory to subordinate directories, you need modify access to the directory.

**Examples**

Suppose that this is the directory hierarchy of `%s1#d02`.

![Diagram of directory hierarchy]

The following command replaces the access control lists and default access control lists of the directories b, c, and d, but not those of directories e, f, and g, with the access control list and default access control list of directory a.

```
propagate_access (current_dir) -exclude e
```
**Related Information**

For more information about access control, see the command descriptions of `display_access`, `display_default_access_list`, `give_default_access`, `remove_access`, and `remove_default_access`. For a detailed discussion of access, see *OpenVOS Commands User’s Guide* (R089) and *OpenVOS System Administration: Registration and Security* (R283).
**read_tape**

**Purpose**

This command reads tape files to disk files in your directory hierarchy.

**Display Form**

```
+----------------------------------+----------------------------------+
|  tape_device_or_port_name:       |  file_names:                     |
|  -file_ids:                      |  -delete: no                     |
|  -delete: no                    |  -truncate: no                   |
|  -relative: no                  |  -relative record_size           |
|  -sequential: no                |  -sequential                     |
|  -stream: no                    |  -stream_output_type             |
|  -multi_reel: no                |  -multi_reel: no                 |
+----------------------------------+----------------------------------+
```

**Command Line Form**

```
read_tape  tape_device_or_port_name  
           file_names [ ] ...            
           [ -file_ids file_ids ... ]    
           [ -delete ]                    
           [ -truncate ]                  
           [ -relative record_size ]      
           [ -sequential ]               
           [ -stream output_type ]        
           [ -multi_reel ]                
```

**Arguments**

- **tape_device_or_port_name**  
  The name of the tape device, or the name of the port attached to the tape drive, holding the tape from which the files are to be read.

- **file_names**  
  Specifies one or more disk files to which read_tape is to read the tape files. By default, the command reads a file specified in the -file_ids argument to the current directory and gives it the same name you specified in the -file_ids argument. If you do not specify a file ID, it reads the tape file at the current position of the tape to a disk file in the current directory with the same name as the file ID of the tape file. In this case, the tape must be a labeled tape. The file names cannot be extended names.
-file_ids file_ids
  Specifies one or more file IDs of the tape files that read_tape is to read. The operating system disregards this argument if the tape is unlabeled. The tape must be positioned to the tape file having a file ID equal to the first file_ids term, and the file IDs of the subsequent tape files must be the same as the subsequent file_ids terms. If you specify a star name, the operating system reads all the files that match the star name on the tape, beginning at the current file. The command names the disk files with the tape file IDs if you omit file_names. By default, the command does not check the tape file IDs.

-delete
  Deletes, without asking, an existing disk file with the same name as a file to be created by the command. By default, the command asks you whether to delete a disk file that would have a conflicting path name with a file created by this command.

-truncate
  Truncates an existing disk file with the same file name as a file to be created by the command.

-relative record_size
  Specifies the organization of the disk file(s) that read_tape is to create. The record_size is the maximum record size. If you specify -relative, you cannot specify -sequential or -stream. By default, the command creates a fixed disk file if the tape file is type f (fixed) or type fb (fixed block), and creates a sequential file otherwise.

-sequential
  Specifies the organization of the disk file(s) that read_tape is to create. If you specify -sequential, you cannot specify -relative or -stream. By default, the command creates a fixed disk file if the tape file is type f (fixed) or type fb (fixed block), and creates a sequential file otherwise.

-stream output_type
  Specifies the organization of the disk file(s) that read_tape is to create. The possible values are output_by_record and output_raw. If you specify output_by_record, the command inserts a line-feed character at the end of each record it reads to the disk file; otherwise, it does not insert the line-feed character. If you specify -stream, you cannot specify -relative or -sequential. If you specify -stream but do not specify an output type, the value of output_type is output_by_record. By default, the command creates the following: a fixed disk file if the tape file is type f (fixed) or type fb (fixed block); a stream file if the tape is a UNIX format; or a sequential file if the tape is of any other format.

-multi_reel
  Allows the read_tape command to read files that span tape volumes; also, the volume_id differs between succeeding reels. If all reels have the same value for the volume_id field, this argument is not needed for read_tape to process the tapes correctly.
read_tape

Explanation

The `read_tape` command reads the files selected by `file_ids` from the tape volume mounted on the specified tape drive or on the tape drive connected to the specified port.

If the tape is labeled, `read_tape` reads as many files as the number of file names you give or the number of file IDs you give, whichever is greater. If you specify an asterisk for the `-file_ids` argument, `read_tape` reads all the files on the tape. If you specify a star name for one of the file IDs, the command reads all the files on the tape that match any of the file IDs, beginning at the current file. If you try to read more files than the tape contains, the command reads as many files as are available, and returns the message `File not found`.

If the tape is unlabeled, `read_tape` reads as many files as the number of file names you give. For unlabeled tapes, it disregards a `-file_ids` argument; therefore you must specify `file_names`. Unlabeled tapes can be read past the end-of-volume mark, if there is additional data.

The `read_tape` command reads the character set information, that is, character set and shift mode, stored with each file on tape, and it sets the correct character set information on each file that it reads.

If you have not yet used the `attach_port` command to attach a port, the `read_tape` command implicitly attaches the port. If you have not yet used the `mount_tape` command to mount a tape, the `read_tape` command implicitly mounts the tape before executing. When execution is completed, if `read_tape` implicitly mounted a tape, it implicitly dismounts the tape. If it implicitly attached a port, it implicitly detaches the port. For more information, see the `Explanation` section in the `mount_tape` command description.

Access Requirements

By default, you have write access to the tape device, with which you can read from and write to a tape. If your system administrator restricts access to the tape device, you need read access to use `read_tape`.

Related Information

See also the command descriptions of `copy_tape`, `create_tape_volumes`, `dismount_tape`, `display_tape_params`, `dump_tape`, `list_save_tape`, `mount_tape`, `position_tape`, `restore_object`, `set_second_tape`, `set_tape_drive_params`, `set_tape_mount_params`, `set_tape_file_params`, `verify_save`, and `write_tape`.  

OpenVOS Commands Reference Manual (R098)  2-639
Purpose
This command displays a prompting message on your terminal.

Display Form

```
ready

-format: medium
```

Command Line Form

```
ready [-format format_code]
```

Arguments
- `-format format_code`
  Specifies the format of the message. The permitted values are `brief`, `medium`, and `long`. By default, the value is `medium`.

Explanation
The `ready` command displays a prompting message on the `default_output` port of your process.

The `brief` format of the message is as follows:

```
ready:
```

The `medium` format of the message is as follows:

```
ready time_of_day
```

The `long` format of the message is as follows:

```
ready time_of_day cpu_time page_faults
```

The `time_of_day` value has the form `hh:mm:ss`, where `hh` is the hour, `mm` is the minute of the hour, and `ss` is the second of the minute. The `cpu_time` value is the amount of central processor time in seconds that your process used since the previous ready message. The `page_faults` value is the number of page faults your process took since the previous ready message.
The `ready` command can be used in command macros to tell you when execution has reached checkpoints in the macro at which the `ready` command is issued. In this context, prompting messages are usually disabled.

**Related Information**

See also the description of the `set_ready` command.
**remove_access**

**Purpose**

This command removes entries from the access control lists of files, directories, and devices.

**Display Form**

```
<table>
<thead>
<tr>
<th align="left">path_names:</th>
</tr>
</thead>
<tbody>
<tr>
<td align="left">-user:</td>
</tr>
<tr>
<td align="left">-all:</td>
</tr>
</tbody>
</table>
```

**Command Line Form**

```
remove_access path_names...
   [-user user_names ...]
   [-all]
```

**Arguments**

- **path_names**  
  Required
  One or more names or star names of files, directories, or device access lists. The path name for device access lists is `>system>acl>access_list_name`. The command removes the specified entry or entries from the access control lists of each of these files, directories, or devices.

- **-user user_names**
  Specifies one or more user names or star names. The command removes entries of these user names from the access control lists. By default, the command uses your user name.

- **-all**
  Removes all entries from the access control lists. If you also specify one or more user names, -all takes precedence, and all entries will be removed from the access control lists of the specified path names. By default, the command uses your user name.

**Explanation**

The `remove_access` command removes entries from the access control lists of the files, directories, and devices specified by the `path_names` argument. The -user and -all arguments determine which entries the command removes.

The `path_names` argument for device access lists contains the file name `access_list_name`. This file name matches the value for the `access_list_name` field for each device in a set in the `devices.tin` file. The `access_list_name` file contains the
access control list (ACL) for a device or set of devices. The operating system creates the ACL from the `devices.txt` file. A user can remove user names from the ACL by issuing the `remove_access` command. The system then automatically updates the ACL. If the `access_list_name` field for a device does not have a value, no ACL is created, and all users can access that device.

Access Requirements

To remove entries from the access control list of a file, you need modify access to the directory containing the file.

To remove entries from the access control list of a directory, you need modify access to the directory containing the directory.

To remove entries from the access control list of a device, you need modify access to the directory that lists the device.

Examples

Suppose the file `make_reports.cobol` in the current directory has the following access control list.

- `Smith.Sales` -- write
- `*.Sales` -- read
- `.*` -- null

The following command removes the last entry in the list.

```
remove_access make_reports.cobol -user *.*
```

Related Information

For more information about access, see the command descriptions of `display_access`, `display_access_list`, `display_default_access_list`, `give_default_access`, `propagate_access`, and `remove_default_access`. For a detailed discussion of access, see OpenVOS Commands User’s Guide (R089) and OpenVOS System Administration: Registration and Security (R283).
remove_default_access

Purpose
This command removes entries from the default access control lists of directories.

Display Form

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>directory_names</td>
<td>Required One or more names or star names of directories. The command removes entries from the default access control list of each directory.</td>
</tr>
<tr>
<td>-user user_names</td>
<td>Specifies one or more user names or star names. The command removes entries of these user names from the default access control lists. By default, the command uses your user name.</td>
</tr>
<tr>
<td>-all</td>
<td>Removes all entries from the default access control lists of each directory. If you also specify one or more user names, -all takes precedence, and all entries will be removed from the default access control lists of the specified directories. By default, the command uses your user name.</td>
</tr>
</tbody>
</table>

Command Line Form

```
remove_default_access directory_names...
  [-user user_names ...]
  [-all]
```

Arguments

- `directory_names` **Required**
  One or more names or star names of directories. The command removes entries from the default access control list of each directory.

- `user_names`
  Specifies one or more user names or star names. The command removes entries of these user names from the default access control lists. By default, the command uses your user name.

- `-all` **[CYCLE]**
  Removes all entries from the default access control lists of each directory. If you also specify one or more user names, `-all` takes precedence, and all entries will be removed from the default access control lists of the specified directories. By default, the command uses your user name.

Explanation

The `remove_default_access` command removes entries from the default access control lists of the directory you specify in the `directory_names` argument. The `-user` and `-all` arguments determine which entries the command removes.

Access Requirements

To remove entries from the default access control list of a directory, you need modify access to the directory.
Examples
Suppose the following is the default access control list of the current directory.

    Smith.Sales    --   write
    Jones.*        --   write
    *.Accounting   --   read
    *.              --   null

The following command removes the last entry in the list.

    remove_default_access -user *.*

Related Information
For more information about access, see the command descriptions of display_access, display_default_access_list, give_default_access, remove_access, and propagate_access. For a detailed discussion of access, see OpenVOS Commands User’s Guide (R089) and OpenVOS System Administration: Registration and Security (R283).
**rename**

**Purpose**
This command changes the name of a file, directory, or link.

**Display Form**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>old_name:</td>
<td></td>
</tr>
<tr>
<td>new_name:</td>
<td></td>
</tr>
<tr>
<td>-delete:</td>
<td>no</td>
</tr>
<tr>
<td>-files:</td>
<td>yes</td>
</tr>
<tr>
<td>-dirs:</td>
<td>no</td>
</tr>
<tr>
<td>-links:</td>
<td>no</td>
</tr>
<tr>
<td>-brief:</td>
<td>no</td>
</tr>
</tbody>
</table>

**Command Line Form**

```
rename old_name
     new_name
       [-delete]
       [-no_files]
       [-dirs]
       [-links]
       [-all]
       [-brief]
```

**Arguments**

- **old_name**
  
  Required
  The path name or star name of objects to be renamed. If `old_name` is not a star name, the command renames the specified object, regardless of its type. If `old_name` is a star name, the command renames all objects that match, subject to the control arguments `-files`, `-dirs`, `-links`, and `-all`.

- **new_name**
  
  Required
  The new name or star name of the objects. The file name must conform to the rules for valid file names on the module where the file is being renamed.

  If you use a star name and the character strings that replace the asterisks cause `new_name` to exceed the maximum length of a file name, the leftmost characters beyond that limit are truncated.
(rename)

- delete
  Deletes a file or directory or unlinks a link that has the same path name as the new name of the renamed object. By default, the command asks you whether to delete the existing object. If you do not delete the existing object, the command does not rename the specified object.

- no_files
  Suppresses the renaming of all files whose names match old_name. This has an effect only if old_name is a star name. By default, the command renames only the specified files. If you specify -dirs or -links in the lineal form, the default becomes -no_files. In that case, you must also specify -files if you also want to rename files.

- dirs
  Renames all directories whose names match old_name. This has an effect only if old_name is a star name. By default, the command renames no directories except when you specify -all.

- links
  Renames all links whose names match old_name. This has an effect only if old_name is a star name. By default, the command renames no links unless you specify -all.

- all
  Renames all files, directories, and links whose names match old_name. This has an effect only if old_name is a star name. By default, if you do not specify the -dirs or the -links argument and old_name is a star name, the command renames only files.

- brief
  Suppresses the display of each object that matches a star name before the object is renamed. By default, the command displays the names.

Explanation

The rename command renames a file, directory, or link.

The argument old_name identifies the object or objects to be renamed, and the argument new_name specifies the new name(s). Either name can be a star name. If the asterisks in the star name can be replaced with definite character strings, and the result is the object name, then the star name matches the object name. For example, the star name S*s matches the objects Sales, September.Sales, Smith.wages, but does not match Sales.Sebertemper.

If new_name contains an asterisk, when an object is renamed, the asterisk is replaced either by the characters that an asterisk in old_name represents, or by the entire old name, if old_name does not contain an asterisk.

Note: If old_name is a star name that matches the name of more than one object of the specified type, then new_name must be a star name.

See the copy_file command for a description of how star names function.
If you specify -delete, the operating system deletes a file or a directory or unlinks a link whose path name matches the path name of the renamed object.

Use -no_files to rename directories and links, but not files. The standard usage of the command is to rename files, but it can also rename other objects. Use -no_files in conjunction with either -dirs or -links or both.

Access Requirements
To rename an object, you need modify access to the directory in which it resides.

Examples
The following command renames the file make_report.cobol.

   rename make_report.cobol make_report.old.cobol -delete

The new name of the file is make_report.old.cobol. The command also deletes a file or directory or unlinks a link named make_report.old.cobol, if one exists in the current directory.
reserve_device

Purpose
This command reserves a device for a batch process.

Display Form

reserve_device

Arguments

- **device_path_name**
  Required
  The path name of a device, such as a tape drive or a disk, to be reserved for a batch process.

Explanation
The reserve_device command reserves a device for a batch process to be started by the current process. If you specify (terminal_name) as the device_path_name, messages will not appear on the message line (usually line 25) of the screen. Use the cancel_device_reservation command to reactivate messages.

Access Requirements
By default, you have write access to a tape device. If your system administrator restricts access to the tape device, you need read access to read from tapes, or write access to read from and write to tapes.

Related Information
For more information on device reservations and batch processes, see the command descriptions of batch, cancel_batch_requests, cancel_device_reservation, display_batch_status, list_batch_requests, and move_device_reservation.
reset_eof

Purpose
This command truncates or extends a stream file to the specified offset.

Display Form

Command Line Form

reset_eof source_file
[offset]
-ask
-all
-brief

Arguments

- source_file
  Required
  The path name of a stream file. If source_file is a star name, the command affects only 64-bit stream files.

- offset
  A value between 0 and 549,235,720,192 that indicates the byte offset of the new end-of-file (EOF). The value 549,235,720,192 requires that the 64-bit stream file in question has a fixed 256-block extent. If the EOF is set at this value, the command cannot add bytes past this location.

  If offset is 0, this command is equivalent to the truncate_file command.

- ask
  Controls whether the command asks you before modifying the file. If source_file is a star name, the default is yes; otherwise, the default is no. Unless you specify -no_ask explicitly, the command always asks in situations where the command would result in extending an ordinary stream file so that it grows by more than 1024 blocks.
-all
Determine whether the command modifies ordinary stream files as well as 64-bit stream files. If source_file is a star name, the default is no; otherwise, the default is yes.

-brief
Suppresses providing details about the result of each completed operation. If source_file is a star name, the default is no; otherwise, the default is yes.

Explanation
The reset_eof command extends or truncates a stream file to a specified offset.

If the file is longer than the value of offset, the command truncates it. If it is shorter, the command extends it with binary zero fill. Extending a 64-bit stream file is quick, and even extending to the maximum value, takes very little time and disk space. This is because these files may be sparse, meaning that blocks containing all binary zeros are not allocated. This is not true for normal stream files; therefore, extending them may take a long time and use significant disk space. The command can truncate normal stream files located on modules running any release prior to OpenVOS 17.2.x, but the command can extend only those files located on modules running Release 17.2.x or later.

The command displays an error message for each eligible stream file for which the truncation/extension operation fails. If the file is not a star name, the command also displays a message if source_file is not an eligible stream file. If the operation succeeds, the command does not display a message unless you specify -no_brief. In that case, the message has one of the following formats:

filename has been truncated by N bytes.

or

filename has been extended by N bytes.

If the file is already the size indicated by offset, the command displays the following message:

filename is already N bytes long.

If you specify a star name, the command displays a similar message for each file modified as the result of the operation unless you specify -brief. In that case, the command displays messages only when it cannot perform the operation (for example, due to access of the underlying capacity of the file). Unless you specify -no_brief, the command does not display messages for files matching the star pattern that are not eligible stream files or that already have the requested length.

The offset is always relative to the beginning of the file, not the current size. Therefore, specifying an offset of 0 truncates the file completely and is equivalent to using the truncate_file command. If you want to truncate the file and retain allocated blocks, you must use truncate_file because reset_eof does not provide the -retain argument.

Table 2-29 lists the maximum offset values for dynamically allocated extent (DAE) files of various extents. (You cannot specify this command for statically allocated extent (SAE) files.)
Examples

The following example is equivalent to specifying the `truncate_file` command.

```
reset_eof *x* 0
Do you want to truncate flex by 54010000011 bytes? (yes, no) n
Do you want to truncate flex.pm by 16384 bytes? (yes, no) n
Do you want to truncate flex1.pm by 16384 bytes? (yes, no) y
```

The following example adds 10,000,000 blocks to the `stream1` file.

```
reset_eof stream1 10000000
Extending eof to 10000000 will add 1220 blocks to the file.
Do you want to extend stream1 by 5000000 bytes? (yes, no) y
```

Related Information

See also `truncate_file`.

---

Table 2-29. Maximum Offset Values for DAE Files

<table>
<thead>
<tr>
<th>Extent Size</th>
<th>Maximum Offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2,145,452,032 (applies to all normal stream files regardless of their extent size)</td>
</tr>
<tr>
<td>8</td>
<td>17,163,616,256</td>
</tr>
<tr>
<td>16</td>
<td>34,327,232,512</td>
</tr>
<tr>
<td>32</td>
<td>68,654,465,024</td>
</tr>
<tr>
<td>64</td>
<td>137,308,930,048</td>
</tr>
<tr>
<td>128</td>
<td>274,617,860,096</td>
</tr>
<tr>
<td>256</td>
<td>549,235,720,192</td>
</tr>
<tr>
<td>flex</td>
<td>540,142,534,656</td>
</tr>
</tbody>
</table>
restore_object

Purpose
This command restores directories, files, or links saved with the `save` or `save_object` command.

Display Form

```
+-------------------------------------+-----------------------------------------------------+
| tape_device_or_port_name:           | restore_object                                      |
| path_name:                         |                                                    |
| object_type: file                  |                                                    |
| destination_dir: current_dir        |                                                    |
| -volume_id:                        |                                                    |
| -restore_acls: yes                 |                                                    |
| -pack: no                          |                                                    |
| -first: no                         |                                                    |
| -replace: no                       |                                                    |
| -unattended: no                    |                                                    |
| -keep_dates: no                    |                                                    |
```

Command Line Form

```
restore_object tape_device_or_port_name path_name object_type
  [destination_dir]
  [-volume_id volume_id]
  [-no_restore_acls]
  [-pack]
  [-first]
  [-replace]
  [-unattended]
  [-keep_dates]
```

Arguments

- **tape_device_or_port_name**
  
  The name of a tape device, or the name of a port attached to a tape drive holding the object to be restored, or the name of a port attached to a disk file.

- **path_name**
  
  The relative or full path name of the saved object. A relative path name is evaluated relative to the current directory. The command restores an object if the path name saved with the object matches the path name you specify.
**restore_object**

- **object_type**
  The type of object to be restored. The possible object types are file, directory, and link. By default, the type is file.

- **destination_dir**
  Specifies the path name of the destination directory. The command restores the saved object to the destination directory. By default, the command uses your current directory.

- **-volume_id  volume_id**
  Specifies the tape volume ID of the first save tape in a set of save tapes. When you restore an object saved on tape, you must supply the volume ID of the first save tape volume. Supply the volume ID either with `volume_id` or when asked by the command. (The `save_object` command stores the volume ID of any additional save tape on the preceding save tape, so it is not necessary to specify more than the first volume ID.)

  When the tape is already mounted, `restore_object` checks the volume ID specified against the volume ID of the tape. If they differ, the command asks if you want to use the volume ID of the tape. If so, it replaces the volume ID specified with the volume ID of the tape; otherwise the command aborts.

  When the port specified in `tape_device_or_port_name` is attached to a disk file, the command disregards this argument.

- **-no_restore_acls**
  Omits the saved access control list of an object when restoring the object. By default, the command restores any saved access control list.

- **-pack**
  Packs a file being restored and discards deleted records. You cannot specify `-pack` if the file has separate-key or item indexes. By default, the command does not pack the file.

- **-first**
  Restores the earliest copy of the saved object. By default, the command restores the latest version of the saved object.

- **-replace**
  Deletes and replaces (with the restored object) an already existing object that has the same name. The command overrides the safety switch and/or expiration date on any file or link to be replaced. By default, the command does not delete the existing object. If the object is a file or a link, you get a message that it already exists; if the object is a directory, you get a message that information can be restored to that object.

- **-unattended**
  Causes tape drives with automatic loaders to switch from one tape to the next, without user intervention. This argument has no effect on tape drives for ftServer modules.

- **-keep_dates**
  Specifies whether to restore the original values of the date-time-used (DTU), date-time-modified (DTM), and date-time-saved (DTS) attributes of the saved object.
If it is set, the DTS value is set to the current time. By default (no), the value of the DTS attribute is 0, and the value of the DTM attribute is the time that the object was restored. Specify yes to restore the original values of these attributes.

**Explanation**

The `restore_object` command restores an object saved earlier with a `save` or `save_object` command. An object is a directory, file, or link. You must have saved the object with the `save` command or the `save_object` command in order to use `restore_object`.

The saved object can be on a save tape or in a disk file.

If you specify the name of a directory as the object to be restored, `restore_object` restores the directory and all directories, files, and links in it.

If you are restoring an object to a tape and have not yet attached a port with the `attach_port` command, the `restore_object` command implicitly attaches a port. If you have not yet mounted a tape with the `mount_tape` command, the `restore_object` command automatically mounts a tape before executing. When execution is completed, if `restore_object` implicitly attached a port, it implicitly detaches the port, forcing the tape to be unloaded. For more information, see the **Explanation** section in the `mount_tape` command description.

If you are restoring an object to a disk file, you must first attach a port with the `attach_port` command. Then specify the port name for the `tape_device_or_port_name` argument of the `restore_object` command. When execution is completed, you must detach the port with the `detach_port` command.

You must switch tapes manually when restoring saved objects on more than one tape.

The `path_name` is the path name of the object in the file system at the time it was saved. This path name identifies the object on the save tape or in the disk file. The path name may be changed after the object is restored. The `restore_object` command matches the `path_name` you give with the command, to the path name of the object on the save tape or in the disk file. The path name matched in each case is a full path name.

If you give the `restore_object` command a relative `path_name`, the command expands it before comparing it to the path name on tape or disk. You must be careful, therefore, to give a relative `path_name` only when your current directory is the directory from which the object was originally saved. In this case, the resulting full path name, expanded relative to your current directory, is the same as the path name the object had when it was saved.

If the saved object is a directory, the `restore_object` command uses the `path_name` you give with the command as the common path name of the files, links, and subdirectories (if any) that it is to restore.

The path name of the object after restoration is the `restore path name`. If the restore path name of a saved file conflicts with the path name of an existing link or directory, the `restore_object` command creates a new name for the file by generating a unique string and adding the suffix `.restore` to it. The command creates a new name in the same way for
a link whose restore path name conflicts with the path name of an existing file or directory. See the description of the `replace` argument for more information.

**Note:** The `display_file_status` command shows index names in order of the index address inside the file. This order may change if the file is specified as the subject of the `copy_file`, `move_file`, `restore_object`, or `save_object` command.

When you specify `-pack`, all indexes are re-created, regardless of file organization, though in some cases the resulting indexes are empty. It is not possible to delete a record from a fixed file with no record index. If you ask the operating system to delete a record from such a file, it updates embedded-key and deleted-record indexes appropriately, but does not actually delete the record. Therefore, such records reappear if their file is packed.

If you specify `-first`, the object restored is the earliest version of the saved directory, file, or link; otherwise, it is the latest version.

**Access Requirements**

By default, you have write access to a tape device, with which you can read from and write to a tape. If your system administrator restricts access to the tape device, you need read access to use `restore_object`. You also need modify access to the directory in which the `restore_object` command restores the object.

**Examples**

**Example 1.**

Suppose `#tape.2.0` is a tape drive on which you have mounted a tape volume. The following command saves your current directory on the tape.

```
save_object #tape.2.0
```

The `restore_object` command asks you for the volume ID of the tape volume mounted on `#tape.2.0`. To restore the directory to its original place in the hierarchy, mount the save tape and issue this command.

```
restore_object #tape.2.0
```

**Example 2.**

In this example, the port `a_port` is attached to the drive on which you have mounted the save tape. You must supply the volume ID of the save tape when the command prompts you for it.

Generally, you will supply a path name for the `save_object` and `restore_object` commands. The following commands are equivalent if the full path name of your current directory is `%s1#d02>Sales>Smith`.

```
restore_object a_port %s1#d02>Sales>Smith>weekly_report
restore_object a_port weekly_report
restore_object a_port (current_dir)>weekly_report
```

**Example 3.**

The following command restores a large volume of weekly reports in the directory `%s1#d02>Rpts>Fred`. These reports are saved on several tapes, and the system in use is
equipped with automatic tape loaders that switch from tape to tape, as needed, to restore the contents of the entire directory.

```
restore_object a_port -unattended %s1#d02>Rpts>Fred>weekly_report*
```

**Related Information**

See also the command descriptions of `copy_tape`, `create_tape_volumes`, `dismount_tape`, `display_tape_params`, `dump_tape`, `list_save_tape`, `mount_tape`, `position_tape`, `read_tape`, `save_object`, `set_second_tape`, `set_tape_drive_params`, `set_tape_mount_params`, `set_tape_file_params`, `verify_save`, and `write_tape`. For information about the `save` and `restore` commands, see *OpenVOS System Administration: Backing Up and Restoring Data* (R285).
save_object

Purpose
This command saves a directory, file, or link to a tape or to a disk.

Display Form

```
--------------------------------- save_object -----------------------------------
tape_device_or_port_name:  [tape_device_or_port_name]
path_name:  [path_name]
-volume_id:  [volume_id]
-number_of_volumes:  [number_of_volumes]
-log:  [log]
-tape_log:  [tape_log]
-compression:  if_available
-unattended:  no
```

Command Line Form

```
save_object tape_device_or_port_name path_name
    [-volume_id volume_id]  
    [-number_of_volumes number]  
    [-log log_name]  
    [-tape_log tape_log_name]  
    [-compression]  
    [-unattended]
```

Arguments

- `tape_device_or_port_name` Required
  The name of a tape drive, or the name of a port attached to a tape drive, holding the tape to be saved, or the name of a port attached to a disk file.

- `path_name` Required
  The relative or full path name of an object to be saved.

- `-volume_id volume_id`
  Specifies an ANSI tape volume ID. By default, if you are saving an object to a tape, `save_object` asks for a tape volume ID when it is ready to write the tape.

  When the tape is already mounted, `save_object` checks the specified volume ID against the volume ID of the tape. If the volume IDs differ, the command asks if you want to use the volume ID of the tape. If so, it replaces the volume ID specified with the volume ID of the tape; otherwise, the command aborts.
If you are saving an object to a disk, this argument is ignored.

- **-number_of_volumes**  \textit{number}

  Specifies the number of tape volumes to be saved. The \textit{number} must be a value between 2 and 255. If you specify this argument, you must also specify the \textit{-volume_id} argument.

- **-log**  \textit{log_name}

  Creates a log file in which \textit{save_object} logs information about every object it saves. The logged information is the full path name of the saved object and the volume ID of the tape on which the object is saved. The command appends the logged information to the end of the log file, so it does not overwrite any data in the file. By default, \textit{save_object} does not create a log file.

- **-tape_log**  \textit{tape_log_name}

  Creates a file \textit{tape_log_name} where the volume IDs of the save tapes are logged. The command appends each volume ID to the end of the tape log file, so it does not overwrite any data in the file.

  If you are saving an object to a disk, this argument is ignored.

- **-compression**

  Enables you to select data compression if you have a tape drive that supports data compression. This argument has no effect on tape drives for ftServer modules.

  The \textit{-compression} argument has a default value of \textit{if_available}. This default value enables data compression when the tape drive supports it but generates no error if the tape drive does not. You can use tape drives with the data compression capability without changing your existing macros.

  \textbf{Note:} Compressibility of data may vary widely.

- **-unattended**

  Causes tape drives with automatic loaders to switch from one tape to the next, without user intervention. This argument has no effect on tape drives for ftServer modules.

\textbf{Explanation}

The \textit{save_object} command saves an object to a tape or to a disk. The command writes the saved object so that you can restore it with the \textit{restore_object} command.

An \textit{object} is a directory, file, or link. If you specify the name of a directory as the object to be saved, \textit{save_object} saves the directory and all directories, files, and links in it.

If you are saving an object to a tape and have not yet attached a port with the \textit{attach_port} command, the \textit{save_object} command implicitly attaches a port. If you have not yet mounted a tape with the \textit{mount_tape} command, the \textit{save_object} command automatically mounts a tape before executing. When execution is completed, if \textit{save_object} implicitly attached a port, it implicitly detaches the port, forcing the tape to be unloaded. For more information, see the \textbf{Explanation} section in the \textit{mount_tape} command description.
If you are saving an object to a disk file, you must first attach a port with the `attach_port` command. Then specify the port name for the `tape_device_or_port_name` argument of the `save_object` command. When execution is completed, you must detach the port with the `detach_port` command.

You must switch tapes manually when saving objects that require more than one tape.

The `save_object` command, like `restore_object`, provides a means of backing up and restoring single objects of any type.

In contrast to the `save` command, `save_object` is provided so you can back up any objects to which you have appropriate access. It is not used for backing up the whole system or even a whole module. The `save` command is described in *OpenVOS System Administration: Backing Up and Restoring Data* (R285).

During execution of `save_object`, if any media or drive errors occur, the system displays a message to describe the situation and then prompts for a course of action.

The path name of each object is saved with the object, thereby retaining a record of its name and its original location in the hierarchy. Thus, when restoring the object with `restore_object`, you can restore it to its original location in the hierarchy. You can also restore the object to another location by explicitly specifying a destination other than the object’s original location.

**Note:** The `display_file_status` command shows index names in order of the index address inside the file. This order may change if the file is specified as the subject of the `copy_file`, `move_file`, `restore_object`, or `save_object` command.

### Access Requirements

By default, you have write access to a tape device. If your system administrator restricts access to the tape device, you need write access to use `save_object`.

You need status access to the directory containing the object you intend to save.

### Example

The following example illustrates a command line that saves a directory, and the directories, files, and links contained in it.

```
save_object #tape.1.0 >Sales>Smith>weekly_reports -volume_id REPORT
```

This command saves Smith’s directory `weekly_reports` to a tape on the tape device `#tape.1.0`, on the current module. The tape volumes are named `REPORT`.

### Related Information

See also the command descriptions of `copy_tape`, `create_tape_volumes`, `dismount_tape`, `display_tape_params`, `dump_tape`, `list_save_tape`, `mount_tape`, `position_tape`, `read_tape`, `restore_object`, `set_second_tape`, `set_tape_drive_params`, `set_tape_mount_params`, `set_tape_file_params`, `verify_save`, and `write_tape`. For information about the `save` and `restore` commands, see *OpenVOS System Administration: Backing Up and Restoring Data* (R285).
**send_message**

**Purpose**
This command enables a user or some other process to send a message to users who are logged in.

**Display Form**

```
----------------------------- send_message -----------------------------
person:                       [text]
  -module:        *  
  -beep:         yes  
  -system:       no   
  -check_receivers: yes
```

**Command Line Form**

```
send_message person
  [text]
    -module module_name
    -no_beep
    -system
    -no_check_receivers
```

**Arguments**

- **person**  
  Required  
  The user name or star name of the person to whom the message is directed.

- **text**  
  The text of the message. It can be up to 60 characters long if you use the display form and 256 characters long if you use the command line form. If you enter the `person` argument and the `text` argument on the command line, `text` must be enclosed by apostrophes if it contains any spaces or any command-line characters such as (, ), !, or ;. By default, `send_message` enters a dialogue mode where it prompts you for messages with an arrow. Each line you type while in this mode is sent as a separate message to `person`. To exit this mode, type a line with a single period.

- **-module module_name**  
  Specifies a module name or star name. By default, `send_message` sends the message to all modules on the current system on which the specified user is logged in.
**send_message**

- **-no_beep**
  Suppresses sending an audible beep to the user’s terminal with the message. By default, `send_message` sends a beep. In dialogue mode, the terminal beeps only once, when the first line of the message is sent.

- **-system**
  Sends the message to the message line of the terminal. If the message does not fit on the message line of the terminal, `-system` is ignored. By default, `send_message` displays the message on the next line of the screen as normal output to the terminal.

  Window terminal devices display the message in the status area, regardless of whether you specify `-system` or not. For a message that exceeds one line, only the first line is initially displayed in the status area. You can view subsequent lines by issuing the `UPDATE_STATUS` request (see the Window Terminal Programmer’s Guide for Asynchronous Communications (R194) for more information about this request).

- **-no_check_receivers**
  Adds your message to the message queue without waiting for the operating system to verify that the specified `person` is logged in. By default, `send_message` waits for this verification before adding your message to the message queue, and informs you if `person` is not logged in.

**Explanation**

The `send_message` command enables a user or some other process to send a message to users’ terminals. Only users who are logged in receive the message.

When you specify `-no_check_receivers`, `send_message` operates much more efficiently. This is most useful in a batch process, when you want a message sent to a person if that person is logged in, and not sent otherwise.

If you specify a period (.) for `person`, the message is sent to `user_name`. (``user_name`` being you). This is useful if you include `send_message` in a command macro. For example, if you use `send_message` to notify yourself that a process is complete, that message is sent to you regardless of your current `group_name`.

You can use the `send_message` command to send messages over the Remote Service Network.

**Examples**

**Example 1.**
To send a relatively short message to user Smith, include your text in the command line.

```
send_message Smith 'Our meeting with Bob and Susan is at 9.'
```

By including `-system` in the command line above, the message appears on the message line of Smith’s terminal rather than on the next line of the screen.
**Example 2.**
If you have a longer message, you can send it line by line by omitting the text in the command line and entering dialogue mode.

```plaintext
send_message Smith
   
```

As each line is sent, you are prompted for the next line with an arrow. To exit the dialogue, type a single period on a line. You do not enter dialogue mode if you specify `-system`

**Related Information**
For additional information about sending messages over the Remote Service Network, see the description of the `send_message` request of the `maint_request` command in the *OpenVOS System Administration: Administering and Customizing a System* (R281).
Purpose

The `set` command assigns, clears, or displays environment variables.

Display Form

None.

Command-Line Form

```
set [var [=value]]
```

Arguments

- `var`
  
  Deletes the environment variable named `var`; does nothing if `var` does not exist.

- `var=value`
  
  The string specified by `value` is assigned to the environment variable named `var`.

Explanation

The `set` command sets a single environment variable. If you do not specify an environment variable, the command displays the names and values of all environment variables.

Some system environment variables are read-only and cannot be modified or removed.

This command ignores extra or invalid arguments.

If you do not specify `=value`, the command deletes the environment variable.
set_cpu_time_limit

Purpose
This command sets the upper bound on the amount of CPU time a process can consume before it is stopped.

Display Form

```
--------------------------- set_cpu_time_limit ---------------------------
cpu_limit: ____________
process_name: __________
-user: ____________ current_user
-module: ____________
```

Command Line Form

```
set_cpu_time_limit cpu_limit
[process_name]
[-user current_user]
[-module module_name]
```

Arguments

- **cpu_limit** Required
  The amount of CPU time, in seconds, that the designated process is allowed to consume before being stopped.

- **process_name**
  The name or star name of the processes for which the CPU time limit is to be set. By default, `set_cpu_time_limit` sets the time limit on the process issuing the command.

- **-user current_user**
  Specifies one or more user names or star names. The command sets the CPU time limit on those processes designated by `process_name` that belong to the specified users. By default, `set_cpu_time_limit` uses your user name. To specify another user name, your process must be privileged.

- **-module module_name**
  Specifies the module on which the processes are running. By default, `set_cpu_time_limit` uses the current module.
set_cpu_time_limit

Explanation

The `set_cpu_time_limit` command sets a CPU time limit for one or more processes. The process is stopped when the accumulated CPU time of the process exceeds `cpu_limit`. If you set `cpu_limit` to 0, the process can consume an unlimited amount of CPU time.
set_default_open_options

Purpose
This command sets the default open options for one or more directories.

Display Form

set_default_open_options directory_names ...
    [-cache_mode default_cache_mode]
    [-brief]

Command Line Form

Arguments

- `directory_names` Required
  One or more names or star names of directories for which default open options are to be set. You must specify at least one directory name before you can bring up the display form for this command.

- `cache_mode default_cache_mode` Cycle
  Specifies the default cache mode. This argument has the following values:

  - None—If you do not specify one of the other values for -cache_mode, or if you do not specify -cache_mode on the command line, leaving the argument blank leaves the cache mode unchanged.
  - normal—Specifies that the cache mode is normal (that is, neither memory resident nor transient).
  - memory_resident—Specifies that files with this value stay in cache until the portion of the cache that is reserved for memory-resident files is filled.
  - transient—Specifies that files with this value are evicted quickly from cache.

- `no_brief` Cycle
  By default (the value yes), the command displays the path name for only those directories that this command modifies. For example, it does not modify directories that
set_default_open_options

currently have their default open options set as requested. If you specify the value no, the command displays all of the path names that are examined.

Explanation

This command sets the default open options for all directories that match the specified star names. Files that are located in the directory or that are created in that directory in the future will inherit the directory’s default open options unless you explicitly set open options for the file or its indexes. A newly created directory inherits its default open options from its parent directory.

If you set the -brief argument to no, the command displays each of the path names that the command examines. When a change is made, the command also displays the before and after values of the default open options. These values are displayed as canonical strings representing the option values (these values are also used in the display_dir_status command).

If the command cannot process one of the files, it displays an error message and continues on to the next file to match the specified directory names.

For detailed information about the open options, see the manual OpenVOS System Administration: Administering and Customizing a System (R281). See also the description of the s$set_default_open_options subroutine in the OpenVOS Subroutines manuals.

Access Requirements

You must have modify access to the directory.

Related Information

See also the descriptions of the display_default_open_options, display_open_options, and set_open_options commands.
set_dir_limits

Purpose
This command sets limits on directory growth.

Display Form

----------------------------- set_dir_limits -----------------------------
directory_names: directory_names
-max_entries: max_entries
-dft_max_entries: dft_max_entries
-max_blocks: max_blocks
-dft_max_blocks: dft_max_blocks
-ask: ask
-brief: brief

Command Line Form

set_dir_limits directory_names
[-max_entries entries]
[-dft_max_entries entries]
[-max_blocks blocks]
[-dft_max_blocks blocks]
[-ask]
[-brief]

Arguments

- directory_names Required
  The names of directories whose limits are to be changed. You can specify star names.
  You must specify at least one name before you can show the display form for this command.
  You cannot change the name in the display form. If you specify a single name that does not contain a star,
  the display form shows any limit values for that directory that you have not explicitly specified on the command line.

- -max_entries entries
  Specifies the maximum number of entries (that is, a file, subdirectory, or link) that can be created in the specified directory.
  Values can range from 1 to 32,700. Since the operating system does not enforce entry limits for standard directories,
  you can specify a value only when directory_names does not reference a specific standard directory,
  in which case an error is issued. When you specify multiple names or a star name, the command ignores this value for standard directories.
If there are already more entries than the number specified, the operating system does not allow more entries to be added until sufficient existing entries have been deleted. If you do not specify a value, the directory’s current value is unchanged.

- **-dft_max_entries entries**
  Specifies the default number of -max_entries values for subdirectories created in the applicable directory; this does not affect any existing subdirectories. Values range from 1 to 32,700. You can specify a value only when directory_names does not reference a specific standard directory, in which case any value is diagnosed. When you specify multiple names or a star name, the command ignores this value for standard directories. If you do not specify a value, the directory’s current value is unchanged.

- **-max_blocks blocks**
  Specifies the maximum number of blocks allowed for a directory. Values range from 1 to 8720. However, if directory_names represents a specific standard directory, the maximum value is 527. Any attempt to grow a directory past the specified number of blocks results in an error. For expandable directories, the smaller of the block limit and entry limit is enforced. When you specify multiple names or a star name, and the value is greater than 527, the command issues a warning for standard directories but does not change them. If you do not specify a value, the directory’s current value is unchanged.

Typically, directories do not shrink in size, so setting -max_blocks to less than the current size prevents further growth but does not reduce the directory size. Instead, use the consolidate_dir command for this purpose.

- **-dft_max_blocks blocks**
  Specifies the default block growth of subdirectories created in the specified directories. Values range from 1 to 8720. For standard directories in which this value is greater than 527 or for any expandable directory, newly created subdirectories are expandable; otherwise, they are standard. This argument allows a standard directory to be set so that all new subdirectories are expandable; such directories have an entry limit of 32,700 by default. If you do not specify a value, the directory’s current value is unchanged.

- **-ask**
  Asks for confirmation that the command should change the directory. If more than one directory name is affected, -ask defaults to yes. Otherwise, -ask defaults to no. However, any explicit designation overrides the default.

- **-brief**
  By default (no), the command issues a message showing the changes made to each directory whose limits were modified because of this command. Specifying yes suppresses these messages. If a directory required no changes, the command does not issue a message unless you have specified a specific directory. If the -ask argument has resulted in a query indicating confirmation of the change to be made, the command issues no further message involving the changing of that limit.
Explanation

The `set_dir_limits` command sets the following types of limits on directory growth:

- Number of blocks. The operating system enforces a limit on block growth. A standard directory can grow to 527 blocks, while an expandable directory can grow to 8720 blocks. Block usage of expandable directories can be limited to lower than this, and for directories on a module running OpenVOS Release 18.x or later, block usage of standard directories can be limited to lower than 527 blocks.

- Number of entries. The operating system enforces limits on entries only for expandable directories. The most entries that can exist in any kind of directory is 32,700. Because of the block limit of 527 on standard directories, the number of entries can never exceed 32,700 and thus enforcement is not necessary.

You cannot set directory limits on a directory located on a disk for which restricted expand mode is set. Any attempt to do so results in an error message. You can use the `display_disk_label` command to see if a disk has restricted expand mode set. For information about setting and displaying information about restricted expand mode, see the descriptions of the `set_dir_expand_mode` and `display_disk_label` commands, respectively, in *OpenVOS System Administration: Disk and Tape Administration* (R284).

Examples

The following example sets the maximum number of entries for the directory `d1` to 1000.

```
set_dir_limits d1 -max_entries 1000
Resetting entry limits of %s1#m1>Sales>d1 from 32700 to 1000.
```

The following example sets the maximum number of blocks for the directory `d2` to 200.

```
set_dir_limits d2 -max_blocks 200
Resetting block limits of %s1#m1>Sales>d2 from 100 to 200.
```

In the following example, the command sets all directories starting with `d` to have 2000 as their maximum number of entries and 500 as their maximum number of blocks.

```
set_dir_limits d* -max_entries 2000 -max_blocks 500
OK to reset limits of directory %s1#m1>Sales>d1?
(yes, no) yes
Resetting entry limits of %s1#m1>d1 from 1000 to 2000.
Resetting block limits of %s1#m1>d1 from 8720 to 500.
OK to reset block limits of %s1#m1>Sales>d2 from 527 to 500?
(yes, no) yes
```

Related Commands

See also the command descriptions of `consolidate_dir`, `display_dir_status`, and `set_dir_type`. 
set_dir_type

Purpose
This command converts a directory’s type from standard to expandable, and vice versa.

Display Form

```
set_dir_type directory_names
  -type: expandable
  -ask: yes
  -brief: no
  -reset_limits: no
```

Command Line Form

```
set_dir_type directory_names
  [-type]
  [-ask]
  [-brief]
  [-reset_limits]
```

Arguments

- **directory_names** Required
  The names of directories whose type is to be changed. You can specify star names. You must specify at least one name.

- **-type** [CYCLE]
  Determines the type of the directory: expandable (the default) or standard. The growth limits set on the directory are the maximum supported by that directory type: 8720 blocks for expandable directories and 527 blocks for standard directories. You can set growth limits independently with the set_dir_limits command.

- **-ask** [CYCLE]
  Asks for confirmation that the command should change the directory. If a directory name is specified as a star name, -ask defaults to yes. Otherwise, -ask defaults to no. However, any explicit designation overrides the default.

- **-brief** [CYCLE]
  By default (no), the command issues a message showing the changes made to each directory whose type was modified because of this command. Specifying yes suppresses these messages. If a directory required no changes, the command does not
issue a message. If the -ask argument has resulted in a query indicating confirmation of the change to be made, the command issues no further message.

-reset_limits
Resets a directory’s limits to the default values for the specified type, without changing the type. When a directory’s type is changed, limits are always set to the defaults for the new type. By default (no), no change occurs for directories of the type specified in the -type argument.

**Explanation**

The `set_dir_type` command converts standard directories to expandable, and expandable to standard, as long as the directory has not exceeded allowable standard directory growth restrictions. You can also use the command to reset the limit values of a directory to the defaults without changing the type.

You can always convert standard directories to expandable, but if an expandable directory has grown beyond standard limits, you cannot convert it back to standard. If you attempt to do so, you may see one of the following error messages:

```
Cannot revert directory of this size to standard.
```

or

```
Cannot revert directory with author entries to standard.
```

If the disk’s directory expand mode is restricted and you attempt to set a directory to expandable, the `set_dir_type` command displays the following error message:

```
set_dir_type mydir -type expandable
set_dir_type: Expandable directories are prohibited on this disk.
```

You can use the `display_disk_label` command to see if a disk has restricted expand mode set. For information about setting and displaying information about restricted expand mode, see the descriptions of the `set_dir_expand_mode` and `display_disk_label` commands, respectively, in *OpenVOS System Administration: Disk and Tape Administration* (R284).
set_dir_type

Examples
In the following example, assume that d1 is expandable and d2 is standard, and that both have default limits.

```
set_dir_type d*
OK to change %s1#m1>Sales>d2 to an expandable directory? (yes, no) y
ready 15:36:28

set_dir_type d1
%s1#m1>Sales>d1 is already an expandable directory.
ready 15:37:20

set_dir_type d1 -type standard
ready 15:46:21

set_dir_type d* -type expandable -no_ask
Changing %s1#m1>Sales>d1 to an expandable directory.
ready 15:47:16
```

Related Commands
See also the command descriptions of consolidate_dir, display_dir_status, and set_dir_limits.
set_expiration_date

Purpose
This command sets an expiration date on a file.

Display Form

```
set_expiration_date
```

Command Line Form

```
set_expiration_date file_name
    [date/time]
```

Arguments

- **file_name**  
  The name or star name of a file or set of files. The command sets the expiration date on all files with matching names.

- **date/time**  
  A date and time. The `date/time` value can be a character string in the standard form:

  \[ yy-mm-dd\_hh:mm:ss \]

  It can also be a character string in any form accepted by the `(date_time)` command function. In this case, the string must be enclosed in apostrophes.

  By default, `set_expiration_date` removes the expiration date from the selected files.

Explanation

The `set_expiration_date` command gives the file an expiration date attribute to protect the file against premature deletion. You cannot delete a file that has an expiration date later than the present time.

Use `set_expiration_date` with `date/time` to add or change a file’s expiration date; omit `date/time` to remove the expiration date if there is one. Use the `display_file_status` command to view a file’s expiration date if one has been set.

See Chapter 1, “OpenVOS Command Functions,” for examples of acceptable date/time input strings.
If you use the `copy_file` or `copy_dir` commands to copy a file or files for which you have set the expiration date, even if you specify the `-keep_dates` argument, these commands **remove** the expiration date from the copied files. If you use the `move_dir` command to move a file or files for which you have set the expiration date, even if you specify the `-keep_dates` argument, this command **removes** the expiration date from the moved files.

If you use the `move_file` command to move a file or files for which you have set the expiration date, this command **preserves** the expiration date in the moved files.

**Related Information**

For a complete list of file attributes, see the description of the `display_file_status` command.
set_external_variable

Purpose

This command sets an external static variable within a program module to a specified value. For tasking programs, the value is only changed in task number one.

Display Form

```
--------- set_external_variable ---------
external_variable_name:
  -in:                   [file_name]
  -to:                   [string]
  -type:                 [char]
```

Command Line Form

```
set_external_variable external_variable_name
  -in file_name
  -to string
  [-type string]
```

Arguments

- `external_variable_name`  
  Required
  The name of the external variable whose value is to be set.

- `-in file_name`  
  Required
  The path name of the program module containing the external variable. The program module can have either a fixed or stream file organization.

- `-to string`  
  Required
  The value to which `external_variable_name` will be set.

- `-type string`  
  [CYCLE]
  Specifies the data type of the variable. The possible values are `integer`, `char`, or `char_varying`. The default is `char`. The length of `external_variable_name` is defined in the program module. Integers must be signed and either 16 or 32 bits.
Explanation

The `set_external_variable` command sets an external static variable within a program module to a specified value. The variable must be defined as static external within the program. You must specify the variable name as it is defined in the program. You must also specify the program and the new value of the variable. You can specify a data type that is the same as or different from the definition within the program, as long as it is compatible with the variable’s function within the program. That is, if you assign a numeric value and specify a type of `char_varying`, you cannot perform arithmetic operations on that variable.

Examples

The following command sets the value of the variable `record_size`, in the program `monthly_sales`, to an integer value of 60.

```
set_external_variable record_size -in monthly_sales -to 60 -type integer
```

Related Information

To determine the current value of the variable, use the `get_external_variable` command.
set_file_allocation

Purpose
This command sets the number of disk blocks that the operating system allocates for a particular file each time that file needs more disk space.

Display Form

Command Line Form

set_file_allocation file_names allocation

Arguments

- **file_names**  
  The name or star name of the file or files whose allocation size you want to set.  

- **allocation**  
  The number of disk blocks to be allocated each time a file needs more disk space. The value can be from 0 to 2000. The value 0 or 1 means to use the default allocation size. The operating system allocates allocation disk blocks if enough free blocks exist on a given volume member. Typically, the blocks are allocated adjacently, but this is not guaranteed. Choosing an allocation size that corresponds to a valid extent size for DAE files (for example, 8, 32, 64, and so on) guarantees that the blocks are adjacent if possible (that is, if that number of unused adjacent blocks exist on the disk). Unlike with DAE files, allocation succeeds without error using noncontiguous blocks if sufficient adjacent blocks are unavailable.

Explanation

The set_file_allocation command sets the number of disk blocks that the operating system allocates for a file specified by file_names each time the file needs more disk space.

If you do not set the allocation size of a file, the operating system uses a default value to determine how many blocks to allocate when disk space is needed. Disks configured for fast-access allocation have a default value of 8; for all other disks, the default value is 1. You can choose to allocate from 1 to 2000 disk blocks. Specifying the value 0 or 1 tells the operating system to use the default value (either 1 or 8). The number of disk blocks allocated and their location depend upon available disk space, fragmentation, the structure of file maps, and the structure of the free space table.
If the file is extent-based, the new allocation size takes effect after all of the extents are used. New blocks are then allocated \( n \) blocks at a time, whenever the file needs new disk space. The `set_file_allocation` command has no effect on dynamically-allocated extent-based files; allocation for such files is always based on the size of the extent.

**Access Requirements**

You need write access to a file to set its allocation size.

**Examples**

The following command sets the allocation size of the file `current_reservations` in the current directory to 32 disk blocks.

```
set_file_allocation current_reservations 32
```

**Related Information**

See also the command descriptions of `compare_files`, `copy_file`, `create_file`, `delete_file`, `display_file_status`, `dump_file`, `locate_files`, `move_file`, and `truncate_file`. 
set_implicit_locking

Purpose

This command turns implicit locking on or off for a file or set of files.

Display Form

```
----------------------------- set_implicit_locking -----------------------------
file_name:                set_implicit_locking
state:                   on
```

Command Line Form

```
set_implicit_locking file_name
[state]
```

Arguments

- **file_name**
  - Required
  - The name or star name of a file or set of files. The command sets the implicit locking switch for the specified file(s).

- **state**
  - [CYCLE]
  - The state of the implicit locking switch to set for the specified file or files. The possible values are on and off. By default, implicit locking is set on for the specified files.

Explanation

The `set_implicit_locking` command turns implicit locking on or off for the file or files whose names match `file_name`.

When implicit locking is on for a file, the file can be opened only for implicit locking. For any attempt to open the file with a different locking specification, the file system overrides that locking specification and sends no error message. See the explanation of file locking in the OpenVOS Subroutines manuals.

Examples

The following command turns on implicit locking for the file `make_report.out`.

```
set_implicit_locking make_report.out on
```

In this case, one process, such as a batch process running a compiler, can append records to the file while another process, such as your login process, reads records in the file.
set_implicit_locking

Related Information

For a complete list of file attributes, see the description of the `display_file_status` command.
set_index_flags

Purpose
This command enables or disables an automatic update on indexes to files.

Display Form

```
set_index_flags path_name
    index_name
    [-no_automatic_update]
```

Arguments

- `path_name` Required
  The path name of a file with an associated index.

- `index_name` Required
  The index associated with the file specified in `path_name`.

- `-no_automatic_update` [CYCLE]
  Disables automatic update of indexes when writing or rewriting a record to a file with embedded keys or embedded-separate keys. By default, the operating system adds new keys to indexes.

Explanation
The `set_index_flags` command enables or disables an automatic update on indexes to files. Use this command with files that have embedded-separate-key indexes.
set_language

Purpose

This command changes the language of the current process.

Display Form

```
--------------------------------- set_language ---------------------------------
language_name:  module_default_language
```

Command Line Form

```
set_language  language_name
```

Arguments

- `language_name` (CYCLE)
  The language to use for the current process. By default, the process uses the default
  language as defined in the current module’s `languages.tin` file.

Explanation

The `set_language` command sets the language of the current process. The language name
you select is the name or synonym of one of the primary languages configured on your
system.

The languages and acceptable synonyms are listed in the
>system>configuration>languages.tin file. If the synonym is not in this file, once
you add it, you must use the `create_table` command to produce an updated
languages.table file, and then move that file to the >system directory. For more
information on this procedure, refer to the OpenVOS System Administration: Configuring a
System (R287).

The language that you select for your process influences things such as your date/time
formats, system and program module message-file selection, and word-search and
case-mapping functions.

Since language is a process attribute, different processes can use different languages. You can
create a subprocess and set its language to one different from that being used in another of
your processes. In a recursive login or a process started through the `start_process`
command, the initial language is that of the invoking process.
**Examples**

The following command sets the language of the process to Italian, provided that Italian has been defined and configured with a name or synonym of `italiano`.

```
set_language italiano
```

**Related Information**

For more information about language use, see the descriptions of the `(language_name)`, `(referencing_dir)`, and the date/time command functions, and the command descriptions of `add_library_path`, `list_library_paths`, and `set_library_paths`. See also the description of the `configure_languages` command in the *OpenVOS System Administration: Configuring a System* (R287). For general information about national language support, see the *National Language Support User’s Guide* (R212).
set_library_paths

Purpose

This command sets the path names of the directories defined as libraries for the current process.

Display Form

--- set_library_paths ---
library_name: include
library_path_names:
- check: yes
- ask: yes

Command Line Form

set_library_paths [library_name]

[library_path_names]
[ -no_check]
[ -no_ask]

Arguments

- **library_name**
  The library whose list of directories is to be changed. Possible values for library_name are include, object, command, and message. By default, set_library_paths changes the directories of the include library.

- **library_path_names**
  One or more directory path names of a library. The names can include the command functions (current_dir) or (home_dir); if library_name is message, the names can also include (referencing_dir) and (language_name). If enclosed in apostrophes, the command functions are evaluated when the path names are used. By default, set_library_paths uses the directories of the libraries defined for the current module. If you do use the argument, but specify a null library_path_names (""), all existing library path names are deleted. By default, the command exits without changing any library paths.

- **-no_check**
  Omit checking the validity of each name specified for library_path_names. By default, set_library_paths checks that each name is a valid path name, although it
**set_library_paths**

does not allow device names even though they are path names, and it allows file names, which cannot be library paths.

- **-no_ask**
  
  Suppresses the prompt when you specify a null string ("") for library_path_names, asking whether to clear the paths. By default, when you specify a null string, set_library_paths queries you before deleting all current library path names. The -no_ask argument is meaningful only when you supply the null string ("") for library_path_names.

**Explanation**

The set_library_paths command sets the list of directories that define the libraries for the current process.

A library is one or more directories that the operating system searches for objects of a particular type. Each module has the following libraries.

- include library
- object library
- command library
- message library

The compilers search the include library for include files; the binder searches the object library for object modules; the command processor searches the command library for commands and the message library for messages associated with individual commands.

A library library_name is defined by an ordered sequence of path names. The order of the list reflects the order in which the operating system searches the directories of a library.

When the operating system looks for an object, it searches for the object in the first directory on the library list. If the object is not in that directory, the search proceeds to the second directory on the list, to the third, and so on. Ordinarily, the module’s default list of libraries serves as a guide to where to find objects.

When you log in, the operating system sets the directories that define each library for your process to the default directories that define libraries for the entire module. You can use set_library_paths to override the default list of libraries, or to change the order in which directories are searched. Your new list replaces the module’s list of default libraries (or any others you have previously set). However, set_library_paths overrides the default list of libraries for your process only, and only for the duration of your process. When you next log in, the module’s default list of libraries applies for your process. If you want to use your own list in preference to the default lists routinely, use set_library_paths in your start_up.cm file.

If you specify a (current_dir), (home_dir), (referencing_dir), or (language_name) command function in library_path_name, you can enclose the command function in apostrophes to prevent its evaluation by the command processor when set_library_paths executes. For example, if you specify ‘(current_dir)’ as library_path_name and include as library_name, the command function is not evaluated until you compile a file containing include files. You can also enclose an entire path
set_library_paths

name containing the (language_name) command function in apostrophes:
‘(language_name)>abc’ or ‘(referencing_dir)>(language_name)>abc’.

If you enter the null string for library_path_names, set_library_paths deletes all of
the existing library path names. The system first prompts you to verify whether you intend to
clear them. The -no_ask argument suppresses this prompt.

Examples

The following command sets the referencing directory of a program module as the message
library to be searched when your process uses that program module.

set_library_paths message ‘(referencing_dir)’

Related Information

For information about other commands that affect the libraries for your process, see the
descriptions of the add_library_path, delete_library_path, and
list_library_paths commands. See also the descriptions of the
list_default_library_paths and set_default_library_paths commands in the
OpenVOS System Administration: Administering and Customizing a System (R281). For
more information about command functions, see Chapter 1, “OpenVOS Command
Functions.”
set_line_wrap_width

Purpose
This command sets the line width to the specified value.

Display Form

```
set_line_wrap_width

width_source:
-line_width:
```

Command Line Form

```
set_line_wrap_width
    [ width_source
    [-line_width width ]
```

Arguments

- `width_source`
  Specifies the source from which the command obtains the width value. Select either `system_default` or `from_driver`. The `system_default` value is 79. The `from_driver` value specifies that the command obtains the width from the driver. (Note that the value of `from_driver` is equivalent to the value returned by the `GET_LINE_LENGTH_OPCODE` opcode of the `s$control` subroutine.) The `width_source` argument and the `-line_width` argument are mutually exclusive.

- `-line_width width`
  Specifies the desired line width. The following table lists possible values for `width`.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-2</td>
<td>Use the line width specified in <code>from_driver</code>.</td>
</tr>
<tr>
<td>-1</td>
<td>Use the line width specified in <code>system_default</code>.</td>
</tr>
<tr>
<td>1 through the largest value that works with the driver</td>
<td>Use this value for the line width.</td>
</tr>
</tbody>
</table>

The `width_source` argument and the `-line_width` argument are mutually exclusive.
set_line_wrap_width

Explanation

The `set_line_wrap_width` command sets the line width to the specified value.

Related Information

For more information about the value of `from_driver`, see the description of the `GET_LINE_LENGTH_OPCODE` opcode of the `s$control` subroutine in the OpenVOS Subroutines manuals.
set_open_options

Purpose
This command sets the open options for one or more files or indexes.

Display Form

<table>
<thead>
<tr>
<th>file_names:</th>
<th>-index:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>-cache_mode:</td>
</tr>
<tr>
<td></td>
<td>-read_ahead:</td>
</tr>
<tr>
<td></td>
<td>-preread_extents:</td>
</tr>
<tr>
<td></td>
<td>-active: no</td>
</tr>
<tr>
<td></td>
<td>-brief: yes</td>
</tr>
</tbody>
</table>

Command Line Form

```
set_open_options file_names ...
  [-index index_name]
  [-cache_mode cache_mode]
  [-read_ahead number_of_blocks]
  [-preread_extents]
  [-active]
  [-no_brief]
```

Arguments

- **file_names** Required
  One or more names or star names of files for which open options are to be set. You must specify at least one name before you can bring up the display form for this command.

- **-index_name** index_name
  Specifies the index or indexes to change. The value of index_name is a string. By default, there is no value for index_name. You can specify an asterisk (*) to set all indexes of the file, including system-defined indexes (for example, _deleted_record_index). If you do not want to change system-defined indexes, specify a star name that precludes them (for example, system-defined indexes always start with an underline character (_)), or name the specific indexes you want to change.
**set_open_options**

- **-cache_mode cache_mode**
  Specifies the cache mode. This argument has the following values:
  - **default**—Specifies that the value should be taken from the containing directory's cache mode default. This is the default value.
  - **None**—If you do not specify one of the other values for `-cache_mode`, or if you do not specify `-cache_mode` on the command line, leaving the argument blank leaves the cache mode unchanged.
  - **normal**—Specifies that the cache mode is normal (that is, neither memory resident nor transient).
  - **memory_resident**—Specifies that files with this value stay in cache until the portion of the cache that is reserved for memory-resident files is filled.
  - **transient**—Specifies that files with this value are evicted quickly from cache.

- **-read_ahead number_of_blocks**
  Specifies the read-ahead value for the specified file. By default, `number_of_blocks` is a number or is empty. This argument has no effect on indexes. If you do not specify a value, the read-ahead value is unchanged.

- **-preread_extents**
  Specifies the preread policy for a dynamically allocated extent (DAE) file or index. Possible values follow:
  - **None**—If you do not specify one of the other values for `-preread_extents`, or if you do not specify `-preread_extents` on the command line, leaving the argument blank leaves the preread policy unchanged.
  - **normal**—Specifies that remaining blocks in the extent (which are always contiguous) are pre-read.
  - **never**—Suppresses any additional blocks from being read from an extent.
  - **seq_only**—Specifies that pre-reading blocks in an extent occurs only when a file is being accessed sequentially.
  - **full**—Specifies that all blocks in an extent, rather than just those ahead, are pre-read.

If you specify this argument for a file or index that is not a DAE file or index, the argument has no effect.

- **-active**
  Sets the open options associated with the current activation of the file. This argument applies only to opened files and has no effect on the permanent open options of those files. The open options set using this argument persist for only as long as the file(s) are activated. By default (the value `no`), the command sets the open options that are permanently associated with the file(s).
**Explaination**

This command sets the open options for all files or indexes that match the given star names.

The default values for the `-cache_mode`, `-read_ahead`, and `-preread_extents` arguments are blank in the following situations:

- If `file_name` is a star name
- If you specify more than one `file_name`
- If `index_name` is a star name

Otherwise, if you specify a specific file name or index, the default values shown are the current values associated with that file or index.

If you issue this command with either a specific name or a star name but do not change the display form, the command does not change any files or indexes. To replicate the value of one file to others, name the file and bring up the display form, and then change the name on the display form to another name (possibly a star name), thereby applying the initial open options to the other files.

If you set the `-brief` argument to `no`, the command displays each of the path names that the command examines. When a change is made, the command also displays the before and after values of the open options. These values are displayed as canonical strings representing the option values.

If the command cannot process one of the files, it displays an error message and continues on to the next specified file.

For detailed information about the open options, see the manual *OpenVOS System Administration: Administering and Customizing a System* (R281). See also the description of the `s$set_open_options` subroutine in the OpenVOS Subroutines manuals.

**Access Requirements**

You must have write access to the file and modify access to the containing directory.

**Related Information**

See also the descriptions of the `display_default_open_options`, `display_open_options`, and `set_default_open_options` commands.
set_owner_access

Purpose
This command sets the owner access switches of a program module.

Display Form

```
---------------------------------------------------------- set_owner_access ----------------------------------------
file_names: ...
access: none
```

Command Line Form

```
set_owner_access file_names ...
[access]
```

Arguments

- **file_names** Required
  One or more names or star names of program modules whose owner-access switches are to be set. You can omit the .pm suffix when you specify the program module names.

- **access**
  The access class to be set. There are four possible values.
  - person
  - group
  - person_and_group
  - none

  The set_owner_access command sets the owner-access switches using this value. By default, the command uses none.

Explanation

The set_owner_access command sets the owner-access switches of your program module. This allows you to give users the ability to execute a program that manipulates files not otherwise accessible to them, without giving them independent access to the files themselves. This is particularly useful for database application programs, where some data must be available to a wide group, but the availability of other data must be restricted.
The two owner-access switches are the person switch and the group switch. The command sets the values of the switches according to the value of the access class `access`. The switches are set as follows:

<table>
<thead>
<tr>
<th>Access Class</th>
<th>Person Switch</th>
<th>Group Switch</th>
</tr>
</thead>
<tbody>
<tr>
<td>person_and_group</td>
<td>True</td>
<td>True</td>
</tr>
<tr>
<td>person</td>
<td>True</td>
<td>False</td>
</tr>
<tr>
<td>group</td>
<td>False</td>
<td>True</td>
</tr>
<tr>
<td>none</td>
<td>False</td>
<td>False</td>
</tr>
</tbody>
</table>

When a user executes a program, the operating system uses the owner-access switches of the program module to assign a temporary user name to the process. When the person switch is true, the person component of the temporary user name is the program module owner’s person name instead of the user’s; when the group switch is true, the group component is the owner’s group instead of the user’s. (The owner of the program module is the same as the author of the program module.) During program execution, the operating system uses this temporary name to determine the process’s access rights, where necessary. (The temporary name may be used for other things as well, such as queue message headers, while in effect.) When execution is complete, the process’s user name reverts to the user’s user name.

When you set the owner-access switches for a program, specify a user who will not be deleted from the registration database. After a user is deleted from the registration database, the system will refuse to run any owner-access programs set to use that person name. You can restore the use of these programs by assigning them a different owner or by re-registering the user. You can prevent the user from logging in, while retaining the ability to run owner-access programs, by setting the account-terminated flag with the `registration_admin` command. For more information about `registration_admin`, see the manual *OpenVOS System Administration: Registration and Security* (R283).

You can see the current values of the person and group switches of a program module using the `display_file_status` command.

**Access Requirements**

To set the owner-access switches of a program module, you need modify access to the containing directory.

**Examples**

Assume that Clark.Accounting is the owner of the file `make_report.pm` and that the user Smith.Sales runs the program.
The following table shows the temporary user name of Smith’s process during the execution of the program for all possible settings of the person and group switches of the program module.

<table>
<thead>
<tr>
<th>Temporary Name</th>
<th>Person Switch</th>
<th>Group Switch</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clark.Accounting</td>
<td>True</td>
<td>True</td>
</tr>
<tr>
<td>Clark.Sales</td>
<td>True</td>
<td>False</td>
</tr>
<tr>
<td>Smith.Accounting</td>
<td>False</td>
<td>True</td>
</tr>
<tr>
<td>Smith.Sales</td>
<td>False</td>
<td>False</td>
</tr>
</tbody>
</table>

**Related Information**

See also the description of the `display_file_status` command.
**set_pipe_file**

**Purpose**
This command defines a file as a pipe file.

**Display Form**

```
set_pipe_file [file_name]
```

**Command Line Form**
```
set_pipe_file [file_name]
```

**Arguments**
- `file_name` **Required**
The name or star name of a file or set of files to be established as pipe files.

**Explanation**
The `set_pipe_file` command permanently defines a file as a pipe file. You can set the pipe file attribute on any empty file of the type fixed, relative, sequential, or stream. However, the file cannot be an indexed file or a 64-bit stream file. This attribute cannot be turned off once it has been turned on.

A pipe file is a special file, similar to a queue, that serves as a means to pass data between programs, input and output ports, tasks, and processes. A pipe file contains no data when closed. Records can be read from a pipe file only after a writer opens it and begins writing records to it. When a record in a pipe file is read, it is consumed, or removed from the file. Data can be written to a pipe file until it is full. (A pipe file is full when its contents exceed $2^{16}$, or 64K, bytes.) When a pipe file is full, a writer cannot write additional records to the file until a reader removes data by reading it. (The file must be reduced to less than 32K before additional records can be written.) Data can be read from the file until it is empty, that is, all the data is consumed. When the pipe file is empty, a reader must wait until a writer adds more data to continue reading.

When the last writer closes the pipe file, reading continues only to the last record in the file, and a reader gets the message e$end_of_file (1025).
**set_pipe_file**

A pipe file can be used in wait mode (the call will not return until the data is read or written) or in no-wait mode (the call returns the message e$caller_must_wait (1277) if the data cannot be read or written yet). A pipe file, therefore, behaves like a device with regard to wait and no-wait modes.

A pipe file uses at most 17 disk blocks for buffering data. When no user has the pipe open, any unconsumed data is no longer retrievable, and the pipe is empty.

**Related Information**

See the description of the `display_file_status` command for an example of setting the pipe file attribute.
**set_priority**

**Purpose**
This command sets the execution priority for a process or set of processes.

**Display Form**
```
-------------------------- set_priority --------------------------
priority: [process_name]
-user: [user_name]
-module: [module_name]
-ask: [ask]
```

**Command Line Form**
```
set_priority priority [process_name] [-user user_name] [-module module_name] [-ask]
```

**Arguments**

- **priority**
  Required
  The priority level to be assigned to the designated process. The minimum priority is 0 and the maximum is 9. The maximum priority for a process is set in the registration file. Use the (process_info) command function to determine the priority of your current process.

- **process_name**
  The name or star name of processes for which the execution priority is to be set. By default, the command sets the priority of the process issuing the command.

- **-user user_name**
  Specifies the name or star name of the users whose processes receive the designated priority. By default, set_priority uses your user name. Your process must be privileged to set the priority of another user’s process.

- **-module module_name**
  Specifies the module on which the processes are running. By default, set_priority uses the current module.
set_priority

- ask

Enables set_priority to ask whether to set priority for a process with a matching name. By default, set_priority does not prompt you to verify the priority setting of each process.

Explanation

The set_priority command sets the execution priority for one or more processes.

Each process has a priority that defines when that process can run relative to all of the other active processes. If your process is privileged, you can set the level of priority to any level on any user process, including your own. If your process is not privileged, you can set the level of priority no higher than the maximum execution priority for which you are registered and only for your own process.

The process name identifies the process (among several processes running with a given user name) for which the priority is to be set.

Examples

If you have more than one process and you issue the set_priority command using a star name for process_name, the system issues the following prompt.

Verify processes to change priority.
Smith.Sales(login)? (yes,no,info)

If you type yes at the prompt, the execution priority is changed; if you type no, the priority remains the same. If you specify info, the system displays information about the subprocess level, program name, and login time of the process. If the process is interactive, the system returns the terminal name from which the process was started. The system does not return a terminal name if the process is not interactive or if the process is logged in remotely from a module that is not running a current version of the operating system.

The system then issues the prompt again.

Logged in at 90-01-19 07:33:26 EDT, sub-process level 0.
Running emacs.pm on %s1#t1.6
Smith.Sales(login)? (yes,no,info)

If your process is at command level, the system returns the following information.

Logged in at 90-01-19 07:33:26 EDT, sub-process level 0.
Running %s1#t1.6
Smith.Sales(login)? (yes,no,info)

Related Information

See OpenVOS System Administration: Administering and Customizing a System (R281) for a discussion on setting and defining priority levels.
**set_ram_file**

**Purpose**

This command sets RAM usage for a specified file or set of files.

**Display Form**

```
--------------------------------- set_ram_file ---------------------------------
file_name: ...
state: on
```

**Command Line Form**

```
set_ram_file file_name
      [state]
```

**Arguments**

- `file_name`  
  The name or star name of a file or set of files.

- `state`  
  Sets the state of the RAM usage for the specified file(s). Values are on (the default) and off.

**Explanation**

The `set_ram_file` command turns RAM usage on or off for the file(s) specified in `file_name`. This command also affects any indexes associated with `file_name`.

When RAM usage is turned on for a file, the data in the file (now called a RAM file) is non-persistent and the file is truncated the next time it is deactivated. This allows the file to occupy cache without any restrictions on the number of blocks that are concurrently modified. This feature is particularly useful for temporary work files.

You can specify `set_ram_file` for a file with the fixed, relative, sequential, or stream organization, and also with one- and two-way server queues. The file cannot be transaction protected.

When you create a one- or two-way server queue, the operating system assigns RAM file usage to the queue by default. One- or two-way server queues, by definition, are non-persistent and are always truncated when deactivated.

If `file_name` has pre-allocated disk storage, all disk addresses in use when the file is deactivated are retained. Otherwise, `file_name` occupies no disk space after it is truncated.
You cannot set RAM file usage on SAE files, because truncating an SAE file reverts it to a normal file. Similarly, truncating a RAM file with an SAE index reverts the index to a normal index without extents.

A file cannot have both RAM usage and its safety switch set.

If a non-empty RAM file is copied using `copy_file`, `move_file`, or `s$copy_file`, the newly created file does not have RAM file usage. This exception is made because a RAM file can be non-empty only when it is currently open, and if the target file had RAM usage, the data copied to it would be lost following completion of the operation. Note that if an attempt is made to move any such open file, an error is reported (after the copy occurs) and the old file is not deleted.

Server queues are assigned RAM file usage by default, and when copied or moved, always retain RAM file usage. The contents of a server queue with RAM usage are never copied; the new server queue is always empty.

If the containing directory of a RAM file is being copied or moved using `copy_dir`, `move_dir`, or `s$copy_dir`, the file in the newly created directory retains its RAM file usage and thus is always empty, regardless of the contents of the original file.

You cannot copy or move a file to an existing RAM file with the `-truncate` or `-pack` command arguments or the corresponding subroutine flags. The result of such an operation would always result in an empty file.

To use this command, you need write access on the file. The file must be empty and not currently in use.

**Examples**

The following command turns on RAM usage for the file `ram_example`.

```sh
set_ram_file ram_example on
```

After specifying the command, `ram_example` has the characteristics of a RAM file. That is, any blocks that are modified, and those of any current or future indexes, remain in cache unwritten until disk activity is very low or until they must be evicted due to other demands on the cache. The number of unwritten, modified blocks for `ram_example` is unlimited; therefore, processes accessing the file never suffer delays due to excessive modified blocks.

If you also specify the `set_open_options` command with the `-cache_mode memory_resident` argument, `ram_example` remains in memory without being evicted if the size of all memory-resident files is less than or equal to the percentage of cache reserved for them.

**Related Information**

See also the descriptions of the `copy_file`, `display_file_status`, and `move_file` commands.
set_ready

Purpose
This command sets the form of the prompting message displayed when your process returns to command level.

Display Form

```
---------------------------------- set_ready -----------------------------------
-format: medium
```

Command Line Form

```
set_ready [-format format_code]
```

Arguments

- `-format format_code`
  Selects the format of the message. There are four permitted values.
  
  - off
  - brief
  - medium
  - long

  By default, the message is in the `medium` format.

Explanation

The `set_ready` command sets the form of the prompting message the operating system displays on your terminal each time it returns your process to command level.

The `off` format code suppresses any prompting message.

The `brief` format of the message is as follows:

```
ready:
```

The `medium` format of the message is as follows:

```
ready  time_of_day
```
The long format of the message is as follows:

    ready time_of_day cpu_time page_faults

The time_of_day value has the form hh:mm:ss, where hh is the hour, mm is the minute of the hour, and ss is the second of the minute. The cpu_time value is the amount of CPU time in seconds that your process used since the previous ready message. The page_faults value is the number of page faults your process took since the previous ready message.

Related Information

See also the description of the ready command.
set_safety_switch

Purpose
This command sets the safety switch for one or more files.

Display Form

```
--------------------- set_safety_switch ---------------------
file_name: ...
state:    on
```

Command Line Form

```
set_safety_switch file_name [state]
```

Arguments

- **file_name** Required
  The name or star name of a file or set of files. The command sets the safety switch for the specified file(s).

- **state** [CYCLE]
  The state of the switch, which can be either on or off. By default, `set_safety_switch` sets the switches for the specified files to on.

Explanation

The `set_safety_switch` command sets the values of the safety switches of the files identified by `file_name`.

When a file's safety switch is on, the file is protected against operations that destroy or damage it. When a file is protected, you cannot do the following:

- delete the file
- truncate the file
- rename the file
- move the file
- edit the file using the `edit` or `line_edit` command
set_safety_switch

However, when a file is protected, you can do the following:

- display a file
- copy a file
- append a record to a file or delete a record from a file
- edit the file using the emacs command

Although setting the safety switch prevents the file from being truncated using the truncate_file command, the copy_file command, and so forth, it does not prevent some or all of its contents from being deleted by a program that opens the file for output or updating. For example, if you open the file in emacs, you can effectively truncate it (even with the safety switch on), by deleting all of its contents and then saving the result.

A file cannot have both RAM usage and its safety switch set.

Access Requirements

You must have modify access to the directory containing the files on which you are setting the safety switch.

Examples

The following command turns on the safety switch of the file make_report.out.

```bash
set_safety_switch make_report.out
```

Related Information

The display_file_status command shows you all of a file’s attributes, including the state of the safety switch.
set_second_tape

Purpose
This command specifies a magnetic tape drive that the tape facility is to use as a second tape drive when processing multivolume tape files.

Display Form

```
--------------------------- set_second_tape ---------------------------
tape_device_or_port_name:  
second_tape_path_name:  
```

Command Line Form

```
set_second_tape tape_device_or_port_name
    second_tape_path_name
```

Arguments

- **tape_device_or_port_name**
  A port attached to a tape drive.

- **second_tape_path_name**
  The device name of the second tape drive.

Explanation

The `set_second_tape` command tells the tape facility the name of a second tape drive to attach to a port that is already attached to a tape drive. In this way, you can use the same port to process different reels of a multivolume tape file on alternating tape drives.

Load the second tape (and fourth tape, sixth tape, and so on) on the second tape drive. (The first drive is used for the first tape, third tape, fifth tape, and so on.) You need to mount only the first tape reel on the first tape drive connected to `tape_device_or_port_name`. The tape facility mounts the subsequent reels, and prompts you with messages to physically load the reels. The tape facility dismounts and unloads tapes on both tape drives when you issue a `dismount_tape` command with `tape_device_or_port_name` as the port.

**Note:** The `set_second_tape` command cannot be used with the `dump_disk` and `reload_disk` commands.

Note: The `set_second_tape` command cannot be used with the `dump_disk` and `reload_disk` commands.
Access Requirements
By default, you have write access to a tape device. If your system administrator restricts access to the tape device, you need read access to read from tapes, or write access to read from and write to tapes.

Related Information
For information about processing tapes, see the command descriptions of attach_port, mount_tape, and dismount_tape. See also the command descriptions of dump_tape, list_save_tape, position_tape, read_tape, restore_object, verify_save, and write_tape.
set_tape_drive_params

Purpose
This command sets the user tape drive parameters that override, for the duration of the port attachment, the default drive attributes of the tape drive attached to a given port.

Display Form
--- set_tape_drive_params ---

Command Line Form

```bash
set_tape_drive_params tape_device_or_port_name
    [-disposition disposition]
    [-reel_retention reel_retention]
    [-multivolume_default value]
    [-message message]
    [-compression]
    [-reset_to_defaults]
```

Arguments

- **tape_device_or_port_name**  
  Required  
  The name of the tape device, or the name of the port attached to the tape drive, whose parameters are to be set.

- **-disposition disposition**  
  Position a tape volume mounted on the tape drive after the operating system closes a tape file. The possible values are rewind, reread, and next. To rewind the tape to the beginning-of-tape mark after the operating system closes the file, use the value rewind. For tape files closed by a program, the default is rewind.

Notes:

1. The `read_tape` and `write_tape` commands position the tape to the next tape file (or to the end-of-volume) instead of rewinding.
2. The `rewind` value does not affect the disposition of multivolume tapes.
To position the tape at the beginning of the tape file following the tape file that was last read or written (or to end-of-volume if there are no more files in the volume), use the value next. To position the tape at the beginning of the tape file that was last read or written, use the value reread.

Note: You cannot reread a 1/4-inch cartridge tape.

- **-reel_retention reel_retention**
  Specifies whether the tape reel remains physically loaded when the port is detached. The possible values are dismount and retain. To prevent the operating system from unloading a tape from a device that you have detached, use the value retain.

  The default value, dismount, causes the tape to be unloaded, unless you enable reel retention with the mount_tape command. Regardless of the value of -reel_retention, the dismount_tape command dismounts the tape and unloads it, unless you use the -no_unload argument of dismount_tape.

- **-multivolume_default value**
  Specifies whether the operating system is to rewind and dismount a volume automatically during a multivolume tape operation on labeled tapes. If you set the argument to yes, the operating system automatically rewinds and dismounts the volume. It also issues a message asking you to mount another volume. If you set the argument to no, the operating system returns an error message when the end of the volume is reached. To set the argument in the command line form, type the argument and the value yes or no. To see the default value, use the display_tape_params command.

- **-message message**
  Specifies messages that you send to the operator each time a user or the operating system mounts or dismounts the tape. After you set this value, the operating system displays your message at the operator’s terminal. If you specify an empty string as the value, the operating system does not display a message.

  When a port is attached to a tape drive, the operating system sets the default value for the -message argument to an empty string.

- **-compression**
  Enables you to select data compression if you have a tape drive that supports data compression. This argument has no effect on tape drives for ftServer modules.

- **-reset_to_defaults**
  Specifies that all the user tape drive parameter values, except those that you are specifying, are to revert to the default values. The default setting of this argument is no. When you cycle this argument to no and you specify any new user tape drive parameters with this command, the new user values are overlaid on the existing user and default values. The new values remain in effect until the port is detached, at which time the new values revert to the default values.
**Explanation**

The `set_tape_drive_params` command lets you set several attributes of the tape drive attached to a given port. Your settings override the default tape parameter values. If you then update the user values, the new user values are overlaid on the existing user values.

The `set_tape_drive_params` command does not implicitly attach a port or mount a tape. You must attach a port explicitly with `attach_port` before you can use `set_tape_drive_params`. Once the port is attached, you can specify either the tape device or port name for `tape_device_or_port_name`, as convenient.

In most cases, you will not need to change the default tape drive parameters. To examine the current user values, use the `display_tape_params` command.

For more information about default, user, and actual values, see the Appendix A, “Setting and Displaying Tape Parameter Values.” For information about tape mounting, see Explanation in the `mount_tape` command description.

**Access Requirements**

By default, you have write access to the tape device, with which you can read from and write to a tape. If your system administrator restricts access to the tape device, you need read access to use `set_tape_drive_params`.

**Examples**

**Example 1.**
The following command rewinds the tape to the beginning-of-tape mark after the operating system closes the file.

```
set_tape_drive_params t_port -disposition rewind
```

**Example 2.**
To position the tape at the beginning of the tape file following the tape file that was last read or written (or to end-of-volume if there are no more files in the volume), use this command.

```
set_tape_drive_params t_port -disposition next
```

**Example 3.**
To prevent the operating system from unloading a tape from a device when you detach the port, use this command.

```
set_tape_drive_params t_port -reel_retention retain
```

**Related Information**

See also the command descriptions of `copy_tape`, `create_tape_volumes`, `dismount_tape`, `display_tape_params`, `dump_tape`, `list_save_tape`, `mount_tape`, `position_tape`, `read_tape`, `restore_object`, `save_object`, `set_second_tape`, `set_tape_mount_params`, `set_tape_file_params`, `verify_save`, and `write_tape`.
**set_tape_file_params**

**Purpose**
This command sets the user tape file parameters that override the default tape parameters while a tape file is open.

**Display Form**

```
set_tape_file_params
```

```
tape_device_or_port_name:  
-file_id:  
-file_id:  
-file_number:  
-expiration_date:  
-translation:  
-file_format:  
-block_length:  
-record_length:  
-blocking_factor:  
-reset_to_defaults:  no
```

**Command Line Form**

```
set_tape_file_params tape_device_or_port_name
    -file_id file_id
    -file_number file_number
    -expiration_date expiration_date
    -translation translation
    -file_format file_format
    -block_length block_length
    -record_length record_length
    -blocking_factor blocking_factor
    [reset_to_defaults]
```

**Arguments**

- **tape_device_or_port_name**  
  Required
  The name of the tape device, or the name of the port attached to the tape drive, on which a tape is mounted.

- **-file_id file_id**
  Specifies the user file ID of a file on a labeled tape to be written. If you specify this argument, the operating system writes the next tape file with `file_id` as its name. The operating system disregards this argument if the tape is unlabeled.
For a labeled tape to be read, if you specify this argument, the operating system checks that one of the file IDs of the tape files matches the file ID that you specify. If the IDs do not match, the operating system tells you it could not find the file you specified and does not read a file. By default, the operating system does not check file IDs.

- **-file_number file_number**
  Specifies the default file number of a file on a labeled tape to be read. When you attach a port to a tape drive, the operating system sets the default value of this parameter to 1. If you do not reset this value, the operating system does not check the file number of the tape file processed through the port. If you specify a number of 1 or higher, the operating system checks that the tape file has the specified file number. If you set the value to 0, the operating system does not check the tape file numbers.

- **-expiration_date expiration_date**
  Specifies the date after which you can delete the contents of a file. You can read and write an expiration date, but the tape processing commands are not affected by the expiration date. To enter the expiration_date, use the ANSI date format (which is the same as the IBM date format for tape labels). This six-byte field has the format cyyddd, where:
  
  - **c** = century (blank = 1900, 0 = 2000, 1 = 2100)
  - **yy** = year (0-99)
  - **ddd** = day (001-366)

- **-translation translation**
  Specifies the character set that the operating system is to use. The possible values are ascii, ebcDIC, and binary. When you attach a port to a tape drive, the operating system sets the default value of this parameter to ascii.

- **-file_format file_format**
  Before you read or write a tape, specifies the format of the files to be read or written. This argument is not necessary when reading labeled tapes. The possible values are summarized in Table 2-30.
The operating system lets you create variable length unix, tar, cpio, and cpioc tape records, but the underlying format is fixed length blocks of 512 bytes.

**Note:** The tar, cpio, and cpioc archive formats are the only valid file formats on tapes with a `tape_format` of `unix`.

- **-block_length block_length**
  Specifies the block length used when reading or writing a tape. When reading a tape, the block length must match that of the data on the tape. In most cases, the operating system default is sufficient, but occasionally it is necessary to alter it. Block length must always be an integral multiple of the record length, and in some file formats, must equal the record length. Alternatively, you can specify the block length, or both the block length and the record length, or else both the record length and the blocking factor. The record length multiplied by the blocking factor equals the block length.

- **-record_length record_length**
  Specifies the record length used when reading or writing a tape. When reading a tape, the record length must match that of the data on the tape. The operating system default is sufficient in most cases, but occasionally it is necessary to alter it. Block length must always be an integral multiple of the record length, and in some file formats, must equal the record length. Alternatively, you can specify the block length, or both the block length and the record length, or else both the record length and the blocking factor. The record length multiplied by the blocking factor equals the block length.

- **-blocking_factor blocking_factor**
  Specifies the number of records per block. Use this argument to determine the maximum number of records that can be packed per block on a write. When you specify a blocking factor, the operating system calculates the block length by multiplying the record length and the blocking factor. The `-file_format` argument must be set to `fb`, `vb`, or `vbs` for packing to occur. You cannot choose this argument and the `-block_length` argument at the same time.

### Table 2-30. File Formats

<table>
<thead>
<tr>
<th>File Format</th>
<th>Record Organization</th>
<th>Blocked</th>
<th>Spanned</th>
</tr>
</thead>
<tbody>
<tr>
<td>f</td>
<td>fixed length</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>fb</td>
<td>fixed length</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>v</td>
<td>variable length</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>vb</td>
<td>variable length</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>vs</td>
<td>variable length</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>vbs</td>
<td>variable length</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>u</td>
<td>variable length</td>
<td>no</td>
<td>no</td>
</tr>
</tbody>
</table>

*f* = fixed length  
*fb* = fixed length, blocked  
*vs* = variable length, spanned  
*u* = undefined  
*v* = variable length  
*vb* = variable length, blocked  
*vbs* = variable length, blocked spanned
-reset_to_defaults

Specifies that all the user tape file parameter values except those that you are specifying are to revert to the default values. The default of this argument is no. When you set this argument to no and you specify any new user tape file parameters with this command, the new user values are overlaid on the existing user and default values. The new values are in effect until the file is closed, at which time the new values revert to the default values.

Explanation

The set_tape_file_params command sets the user tape parameters of tape files that you plan to read or write. The user tape parameters override the default tape parameters while a tape file is open.

The set_tape_file_params command does not implicitly attach a port or mount a tape. You must attach a port with attach_port before you can use the set_tape_file_params command. You must also mount a tape with mount_tape before you can use set_tape_file_params. Once the port is attached, you can specify either the tape device or port name for tape_device_or_port_name, as convenient.

In most cases, you will not need to change the default tape file parameters. To examine the current user values, use the display_tape_params command.

For more information about default, user, and actual values, see the Appendix A, “Setting and Displaying Tape Parameter Values.” For information about tape mounting, see Explanation in the mount_tape command description.

For the -file_format argument, labeled tapes can contain files with any of the formats given in Table 2-30. Unlabeled tapes can contain files having either undefined-length or fixed-length record formats. Undefined-length records are the same on both labeled and unlabeled tapes. On unlabeled tapes, fixed-length records have one or more records packed into a single tape block. Make the block length an exact multiple of the record length. If you do not, and the operating system writes a partial record to end the block, the operating system reports the remaining part of the record as an error. You cannot use variable-length record formats with unlabeled tapes. If you try to specify a variable-length record format, the operating system defines undefined-length records instead.

The operating system can process files containing records with the formats defined by ANSI and IBM for tape files. ANSI allows files with fixed-length or variable-length records. IBM allows files with fixed-length, variable-length, and undefined-length records. The operating system can also process undefined-length records on ANSI-labeled tapes.

Note: ANSI and IBM label tapes have the default expiration date value of none ("00000"—note the leading space).

The file format names shown in Table 2-30 are the names for use with OpenVOS; they are not the names specified on non-OpenVOS tapes. The names translate into record formats on both ANSI-format tapes and unlabeled tapes, and into record formats and block attributes on IBM-format tapes. Table 2-31 shows OpenVOS names and equivalent names in ANSI, IBM, and unlabeled formats.
Table 2-31. Tape File Format Names

<table>
<thead>
<tr>
<th>File Format</th>
<th>ANSI-Record Format</th>
<th>IBM-Record Format</th>
<th>IBM-Block Attribute</th>
<th>Unlabeled Record Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>f</td>
<td>F</td>
<td>F</td>
<td>blank</td>
<td>F</td>
</tr>
<tr>
<td>fb</td>
<td>F</td>
<td>F</td>
<td>B</td>
<td>F</td>
</tr>
<tr>
<td>v</td>
<td>D</td>
<td>V or D</td>
<td>blank</td>
<td>U</td>
</tr>
<tr>
<td>vb</td>
<td>D</td>
<td>V or D</td>
<td>B</td>
<td>U</td>
</tr>
<tr>
<td>vs</td>
<td>S</td>
<td>V or D</td>
<td>S</td>
<td>U</td>
</tr>
<tr>
<td>vbs</td>
<td>S</td>
<td>V or D</td>
<td>R</td>
<td>U</td>
</tr>
<tr>
<td>u</td>
<td>U</td>
<td>U</td>
<td>blank</td>
<td>U</td>
</tr>
</tbody>
</table>

Related Information

See also the command descriptions of copy_tape, create_tape_volumes, dismount_tape, display_tape_params, dump_tape, list_save_tape, mount_tape, position_tape, read_tape, restore_object, save_object, set_second_tape, set_tape_drive_params, set_tape_mount_params, verify_save, and write_tape.
set_tape_mount_params

Purpose

This command determines the user tape mount parameters that override, for the duration of
the mount, the default mount attributes of the mount_tape command.

Display Form

```
--------------------------- set_tape_mount_params ---------------------------
tape_device_or_port_name:
-volume_id:
-owner_id:
-tape_format:
-cartridge_no:
-access_rights:
-reset_to_defaults: no
```

Command Line Form

```
set_tape_mount_params tape_device_or_port_name
 [-volume_id volume_id]
 [-owner_id owner_id]
 [-tape_format tape_format]
 [-cartridge_no cartridge_no]
 [-access_rights access_rights]
 [-reset_to_defaults]
```

Arguments

- **tape_device_or_port_name** Required
  The name of the tape device, or the name of the port attached to the tape drive, on which
  a tape is mounted.

- **-volume_id volume_id**
  Identifies the volume with a character string. If you specify a volume ID, then mount a
  tape, the operating system checks the specified volume ID against the volume ID on the
  tape label.

- **-owner_id owner_id**
  Identifies the owner with a character string. When you attach a port to a tape drive, the
  operating system sets the default value of this parameter to your person name. If you
  specify an owner ID, then mount a tape, the operating system checks the specified
  owner against the owner ID on the tape label.
*set_tape_mount_params*

- **-tape_format tape_format**
  Specifies the format of the tape, including the format of the label at the beginning of the tape and the markers that begin and end each tape file.

  The possible values for `tape_format` are: `ansi` for ANSI-labeled tapes, `ibm` for IBM OS/VS-labeled tapes, `ibm_mvs` for tapes to be used on MVS/RACF systems, `unix` for tapes that have UNIX tar, cpio, or cpio formats, and `unlabeled` for unlabeled tapes.

  The operating system also sets the default translation mode according to the tape format you specify. When you choose the format `ansi` or `unix`, the default translation mode is `ascii`; when you choose the format `ibm` or `ibm_mvs`, the default translation mode is `ebcdic`; and when you choose the format `unlabeled`, the default translation mode is `binary`. You can set the default translation mode by giving the `-translation` option of the `set_tape_file_params` command. In this case, be careful to choose a translation mode that is consistent with the specified tape format. The `-translation` option overrides any default.

- **-cartridge_no cartridge_no**
  Determines the tape that is mounted next. Only 0 (or blank) is allowed.

- **-access_rights access_rights**
  Specifies the access to set on the tape to be mounted. The possible values are `read_write` and `readonly`. When you attach a port to a tape drive, the operating system sets the default value of this parameter to `read_write`. You can use the `mount_tape` command to override the default setting of the `-access_rights` argument. If you plan only to read tapes, it is safer to specify `readonly`. If you specify `readonly`, the tape facility accepts a tape with or without a write ring, but does not allow you to write to the tape.

- **-reset_to_defaults**
  Specifies that all the user tape mount parameter values, except those that you are specifying, are to revert to the default values. The default of this argument is `no`. When you set this argument to `no` and you specify any new user tape mount parameters with this command, the new user values are overlaid on the existing user and default values. The new values are in effect until the tape is dismounted, at which time the new values revert to the default values.

**Explanation**

The `set_tape_mount_params` command determines the user tape mount parameters that override, for the duration of the mount, the default mount attributes of the `mount_tape` command.

This command does not implicitly attach a port or mount a tape. You must attach a port explicitly with `attach_port` before you can use `set_tape_mount_params`. Once the port is attached, you can specify either the tape device or port name for `tape_device_or_port_name`, as convenient.

In most cases, you will not need to change the default tape mount parameters. You can examine the default tape mount parameters by using the `display_tape_params` command.
If you do choose to change the default mount parameters, issue the `set_tape_mount_params` command before mounting the tape with `mount_tape` or with a command that mounts the tape implicitly.

For more information about default, user, and actual values, see the Appendix A, “Setting and Displaying Tape Parameter Values.” For information about tape mounting, see the Explanation section in the `mount_tape` command description.

The ANSI tape format, the IBM OS/VS and MVS label formats, and the UNIX `tar` (TARchive), `cpio` (CoPy I/O), and `cpioc` (`cpio -c`) header formats are some of the most commonly used formats in the computer industry. Unlabeled tapes are those that do not have ANSI labels, IBM OS/VS or MVS labels, or UNIX `tar`, `cpio`, or `cpioc` headers. You can process a labeled tape with a nonstandard label by treating it as an unlabeled tape. A nonstandard label is one that does not correspond to the ANSI, IBM OS/VS, IBM MVS, UNIX `tar`, UNIX `cpio`, or UNIX `cpioc` standards.

**Note:** The difference between the `ibm` OS/VS and `ibm_mvs` formats is important only when you are writing a tape that is to be read by one of these systems. OpenVOS ignores the difference between the two formats when reading a tape. For example, you can specify the `ibm` OS/VS format and still read a tape written with an IBM System MVS/370 tape label, or vice versa.

When the operating system processes an unlabeled tape, all records are treated as data records. A tape mark, which is a zero-length block, is treated as an end-of-file mark. Two consecutive tape marks define an end-of-volume mark. Any label records encountered on an unlabeled tape are returned as data.

Because OpenVOS uses ANSI tape labels by default, ANSI terms are used for tape label fields. Table 2-32 lists the names of the fields and the equivalent IBM OS/VS and MVS label names.

**Table 2-32. ANSI and IBM Tape Label Fields**

<table>
<thead>
<tr>
<th>ANSI</th>
<th>IBM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Volume ID</td>
<td>Volume Serial Number</td>
</tr>
<tr>
<td>Owner ID</td>
<td>Owner Name and Address Code</td>
</tr>
<tr>
<td>File ID</td>
<td>Data Set Identifier</td>
</tr>
<tr>
<td>Block Length</td>
<td>Block Length</td>
</tr>
<tr>
<td>Record Length</td>
<td>Record Length</td>
</tr>
</tbody>
</table>

**Access Requirements**

By default, you have write access to the tape device, with which you can read from and write to a tape. If your system administrator restricts access to the tape device, you need read access to use `set_tape_mount_params`.

**Related Information**

See also the command descriptions of `copy_tape`, `create_tape_volumes`, `dismount_tape`, `display_tape_params`, `dump_tape`, `list_save_tape`.
set_tape_mount_params

mount_tape, position_tape, read_tape, restore_object, save_object,
set_second_tape, set_tape_drive_params, set_tape_file_params,
verify_save, and write_tape. See also OpenVOS System Administration: Disk and Tape
Administration (R284).
set_terminal_parameters

Purpose

This command sets the terminal parameters for your terminal or window terminal device.

The command’s display and command line forms differ, depending on your OpenVOS terminal’s device type. Therefore, this command description contains the display form and command line form for terminal devices and for window terminal devices. The argument descriptions are presented in order, with the unique window-terminal arguments interspersed with the non-window terminal arguments, and the additional window-terminal arguments at the end of the argument descriptions.

To determine your terminal’s device type, issue the following command at OpenVOS command level.

display_device_info (terminal_name)

The value of the type field shown in the output of this command is your terminal type.

Display Form for Terminal Device

```plaintext
----------------------------- set_terminal_parameters -----------------------------
-line_length: 0
-terminal_type: v102
-setup:
-escape:
-system_message:
-prompt_message:
-continue_message: +
-pause_message: --PAUSE--
-pause_lines: 23
-tabs1: 6,11,16,21,26,31,36,41,46,51,56,61,66,71,76,81
-tabs2: 86,91,96,101,106,111,116,121,126
-break_enabled: yes
-black_on_white: no
-cursor_format: steady_block
-key_click_on: no
-smooth_scroll: no
-interrupt_key_enabled: yes
```
**set_terminal_parameters**

**Command Line Form for Terminal Device**

```
set_terminal_parameters [-line_length line_length]

[-terminal_type terminal_type]
[-setup setup_name]
[-escape escape_char]
[-system_message system_message]
[-prompt_message prompt_message]
[-continue_message continue_message]
[-pause_message pause_message]
[-pause_lines pause_lines]
[-tabs1 tabs1]
[-tabs2 tabs2]
[-no_break_enabled]
[-black_on_white]
[-cursor_format cursor_format]
[-key_click_on]
[-smooth_scroll]
[-no_interrupt_key_enabled]
```

**Display Form for Window Terminal Device**

```
--------------------------- set_terminal_parameters --------------------------
-line_length:           80
-terminal_type:         cac_xterm
-setup:                
-system_message:       
-prompt_message:       
-continue_message:     +
-pause_message:        --PAUSE--
-pause_lines:          24
-max_typeahead_lines:  10
-tabs1:         6,11,16,21,26,31,36,41,46,51,56,61,66,71,76,81
-tabs2:             86,91,96,101,106,111,116,121,126
-break_action:       signal_and_discard
-cursor_format:       blinking_block
-wait_cursor_format:  off
-forms_style:         overlay
-key_timeout:         5
-escape:              ` -break_to_wmgr: yes
-black_on_white:      no -initial_overlay_on: no
-key_click_on:        no -cancel_doesnt_abort: no
-smooth_scroll:       no -return_doesnt_unpause: no
-interrupt_key_enabled: yes -return_doesnt_tab: no
-clear_reused_screen_pages: no -allow_multiple_users: no
```
Command Line Form for Window Terminal Device

```
set_terminal_parameters [-line_length line_length_number]
```

```
[-terminal_type terminal_type]
[-setup setup_name]
[-system_message system_message]
[-prompt_message prompt_message]
[-continue_message continue_message]
[-pause_message pause_message]
[-pause_lines pause_lines_number]
[-max_typeahead_lines max_typeahead_lines]
[-tabs1 tabs1]
[-tabs2 tabs2]
[-break_action break_action]
[-cursor_format cursor_format]
[-wait_cursor_format cursor_format]
[-forms_style forms_style]
[-escape escape_char]
[-no_break_to_wmgr]
[-black_on_white]
[-initial_overlay_on]
[-key_click_on]
[-cancel_doesnt_abort]
[-smooth_scroll]
[-return_doesnt_unpause]
[-no_interrupt_key_enabled]
[-return_doesnt_tab]
[-clear_reused_screen_pages]
[-allow_multiple_users]
[-key_timeout number]
```

**Arguments**

- `-line_length line_length_number`
  Sets the terminal’s line length. Setting a line length that is too small may affect the display form function; the effect depends on the command being displayed. The operating system displays up to one less character on a line than `line_length_number`. If a line's length equals or exceeds `line_length_number`, the operating system uses as many additional terminal lines as necessary to display all the characters. The value of `line_length_number` can range from 16 to 160, and includes the length of the prompt message. By default, the operating system uses the current value set for your terminal. If you specify both `line_length_number` and a new `terminal_type` on the same command line, the operating system ignores `line_length_number` and uses the value implicit in the new terminal type. To set the line length for a new terminal type, you must reinvoke the command.

  **Note:** If you are logged into a window terminal, you **cannot** change the `-line_length` value.
set_terminal_parameters

- **-terminal_type terminal_type**
  
  Sets the terminal type. Use the list_terminal_types command to list all possible values of terminal_type. Cycling through this field displays the same values as the list_terminal_types command. By default, the operating system uses the current value set for your terminal. When you set terminal_type to a new value, the operating system uses the values for line_length_number, escape_char, and pause_lines_number that are implicit in the new terminal type, and ignores any new values given for these arguments on the command line. To change the values implicit in the new terminal type, you must reinvoke the command.

- **-setup setup_name**
  
  Sets the dimensions of the screen. The permissible values are determined by the setup configuration section of the .ttp file for the particular terminal type. All terminal types supported by Stratus use setup names of the form HxV, where H is the number of lines and V is the number of columns. By default, the operating system uses the default value for the specified terminal type.

- **-escape escape_char**
  
  Sets the escape character. The default value for the escape character is the grave accent (’), the ASCII character with the hexadecimal representation 60h. By default, the operating system uses the current value set for your terminal. If you specify both escape_char and a new terminal_type on the same command line, the operating system ignores escape_char and uses the value implicit in the new terminal type. To set the escape character for a new terminal type, you must reinvoke the command.

- **-system_message system_message**
  
  Specifies the character string that the operating system displays on line 25 of your terminal, if the terminal has a 25th line. The system_message can be up to 53 characters long. This message is displayed once, when the command is executed. By default, the operating system uses the current value set for your terminal.

- **-prompt_message prompt_message**
  
  Specifies the character string that the operating system displays when it is ready to accept a command or request. For window terminals, the prompt_message can be up to 32 characters long; otherwise, the prompt_message can be up to 8 characters long. By default, the operating system uses the current value set for your terminal.

- **-continue_message continue_message**
  
  Specifies the character string that the operating system displays at the beginning of a continued line. For window terminals, the continue_message can be up to 32 characters long; otherwise, the continue_message can be up to 8 characters long. By default, the operating system uses the current value set for your terminal.

- **-pause_message pause_message**
  
  Specifies the character string that the operating system displays after it has displayed the number of terminal lines specified by pause_lines. For window terminals, the pause_message can be up to 32 characters long; otherwise, the pause_message can be up to 20 characters long. By default, the operating system uses the current value set for your terminal.
-pause_lines pause_lines_number
Specifies the number of terminal lines that the operating system displays on your screen before pausing. If you set pause_lines_number to zero, no pauses occur in output to the terminal. If you set pause_lines_number to a value greater than zero, the operating system displays pause_message after pausing and waits for you to press the RETURN key, the ENTER key, the key that invokes the CANCEL function, or the key that invokes the NO PAUSE function. If you press RETURN or ENTER, the operating system displays the next number of terminal lines (determined by pause_lines_number) of output from the current command. If you press the key that invokes the CANCEL function, the operating system aborts the rest of the output from the current command and returns you to command level. If you press the key that invokes the NO PAUSE function, the operating system scrolls any remaining output from the current command onto the screen without further pauses. By default, or if you specify a negative number, the operating system uses the current value set for your terminal. If you specify both pause_lines_number and terminal_type on the same command line, the operating system ignores pause_lines_number and uses the value implicit in the new terminal type. To set pause_lines_number for a new terminal type, you must reinvoke the command.

If the value of pause_lines_number is a negative number, if the value is greater than the screen size of the specified terminal type, or if the value is the same as the argument’s current setting, OpenVOS sets -pause_lines to the value implicit in the new terminal type.

-tabs1 tabs1
Sets a group of tab stops for your terminal. The argument tabs1 must be a sequence of column numbers in increasing order. The numbers must be separated by commas. If there are tab stops set for -tabs2, then the largest column number specified in tabs1 must be less than the smallest column number specified in tabs2. By default, the operating system uses the current tabs1 values set for your terminal.

-tabs2 tabs2
Sets a second group of tab stops for your terminal. The argument tabs2 must be a sequence of column numbers in increasing order. The numbers must be separated by commas. If there are tab stops set for -tabs1, then the smallest column number specified in tabs2 must be greater than the largest column number specified in tabs1. If you specify -tabs2 and there are no tab stops set for -tabs1, the operating system assigns the values you specify for tabs2 to tabs1 instead. By default, the operating system uses the current tabs2 values set for your terminal.

-no_break_enabled
Disables the function of the [CTRL]BREAK key sequence. For more information on specifying the break level, see the OpenVOS Commands User’s Guide (R089). By default, the operating system uses the current value set for your terminal.

-black_on_white
Displays dark characters on a light background (inverse video). By default, the operating system uses the current value set for your terminal.
set_terminal_parameters

- `cursor_format cursor_format`
  Sets the form of the cursor. There are five possible values.
  - blinking_block
  - blinking_underline
  - steady_block
  - steady_underline
  - off

  By default, the operating system uses the current value set for your terminal.

- `wait_cursor_format cursor_format`
  Sets the format of the wait cursor that is displayed in the lower left corner of the window while a window terminal is waiting for a response. There are five values.
  - blinking_block
  - blinking_underline
  - steady_block
  - steady_underline
  - off

  By default, the operating system uses the current value set for your terminal.

- `forms_style forms_style`
  Determines how display forms are displayed. There are three possible values.
  - overlay
  - insert
  - scroll

  If you specify the `overlay` style and later press the key that invokes the DISPLAY FORM function for a command, the operating system causes the command’s display form to overlay the primary window. When you press the [ENTER] key or the key that invokes the CANCEL function, the display form disappears and command line prompt appears on the next line.

  If you specify the `insert` style and later press the key that invokes the DISPLAY FORM function, the operating system causes the command’s display form to scroll up the primary window as is done by non-window terminal devices. When you press the [ENTER] key or the key that invokes the CANCEL function, the command line prompt appears following the display form.

  If you specify the `scroll` style and later press the key that invokes the DISPLAY FORM function for a command, the operating system causes the command’s display form to scroll up the primary window. When you press the [ENTER] key or the key that invokes the CANCEL function, the operating system scrolls the primary window down so that the display form disappears. It then displays the command line prompt on the next line.

  By default, the operating system uses the current value set for your terminal.
- **key_click_on**
  Enables key clicks. By default, the operating system uses the current value set for your terminal.

- **smooth_scroll**
  Enables smooth scrolling on your terminal. By default, the operating system uses the current value set for your terminal.

- **no_interrupt_key_enabled**
  Disables the INTERRUPT key function when you are executing edit, edit_form, nls_edit_form, or a program using the forms processor. By default, the INTERRUPT key function is enabled.

- **max_typeahead_lines number**
  Specifies the maximum number of lines you can type ahead before the operating system reads the lines.

  **Note:** This argument is only available to window terminal users.

- **break_action break_action**
  Determines the function of the [CTRL BREAK] key sequence and replaces the -break_enabled argument. You can specify one of the following values.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>signal_and_discard</td>
<td>Signal a break and discard input (equivalent to -break_enabled).</td>
</tr>
<tr>
<td>signal_only</td>
<td>Signal a break, but do not discard input.</td>
</tr>
<tr>
<td>return_nul_char</td>
<td>When calling set_terminal_parameters from an application, break returns a null character instead of signalling. Stratus does not recommend the use of this value from the command line.</td>
</tr>
<tr>
<td>return_error</td>
<td>When calling set_terminal_parameters from an application, break returns an error code instead of signalling. Stratus does not recommend the use of this value from the command line.</td>
</tr>
<tr>
<td>ignore</td>
<td>Ignore breaks (equivalent to -no_break_enabled).</td>
</tr>
</tbody>
</table>

  For more information on specifying the break level, see the OpenVOS Commands User’s Guide (R089).

  **Note:** This argument is only available to window terminal users.

- **clear_reused_screen_pages**
  Specifies that the window manager clear a new page of terminal memory before displaying a subsequent application screen. By default, the window manager uses the existing application screen as a basis for displaying a subsequent application screen. This argument affects how the window manager animates the display of application screens; it does not affect the final appearance of application screens.
Usually, specifying a value of no improves display time because many application screens have a similar appearance. The disadvantage is that some information from an old screen appears momentarily on a subsequent screen when the window manager first displays the subsequent screen. You may notice this problem if you have a slow terminal or are connected to the Stratus module over a slow communications line. Specifying a value of yes may help reduce this problem, depending on the terminal type.

**Note:** This argument is only available to window terminal users that have terminals with multiple pages of screen memory.

**-break_to_wmgr**

Specifies that the window manager is invoked when you press the [BREAK] key. To get to break level, press the [BREAK] key while in window manager mode. In other words, you must press the [BREAK] key twice to get to break level. If you specify a value of no, pressing the [BREAK] key executes the break action. The default value is to invoke the window manager.

If you are running a command or application which does not use terminal independent keystroke translation, specifying the default value lets you invoke the window manager. If you specify a value of no, you cannot access the window manager from a command or application which does not use terminal independent keystroke translation.

**Note:** This argument is only available to window terminal users.

**-initial_overlay_on**

Specifies the value of the insert/overlay mode switch. By default, the overlay switch is on when a window is first opened. On subsequent opens of a window, you can set the insert/overlay mode with the insert/overlay mode function key.

**Note:** This argument is only available to window terminal users.

**-cancel_doesnt_abort**

Prevents the CANCEL key function from aborting the display of output. When you give this argument, pressing the key that invokes the CANCEL function causes the display of a command-level message that explains the proper way to abort a pause. If you do not give this argument, pressing the key that invokes the CANCEL function aborts the display of output.

**Note:** This argument is only available to window terminal users.

**-return_doesnt_unpause**

Prevents the [RETURN] key from releasing a pause when there is no current input record at command level. When you give this argument, pressing the [RETURN] key when there is no current input record causes the display of a command-level message that explains the proper way to release a pause. If you do not give this argument, the [RETURN] key releases the pause when there is no current input record. In either case, the [RETURN] key enters the input record when one exists.

**Note:** This argument is only available to window terminal users.
**set_terminal_parameters**

- **-return_doesnt_tab**  
  In a display form, prevents the \[RETURN\] key from tabbing in every direction. When you give this argument, pressing the \[RETURN\] key moves the cursor to the first field on the next line instead of performing a tab, and you must press the \[TAB\] key to tab horizontally to a field in the display form. If you do not give this argument, the \[RETURN\] key enables the cursor to tab in every direction in a display form.

  **Note:** This argument is only available to window terminal users.

- **-allow_multiple_users**  
  Allows window terminal devices to be shared on an individual basis. This argument takes effect immediately for the specified device.

  **Note:** This argument is only available to window terminal users.

- **-key_timeout** \( \text{number} \)  
  Specifies the number of seconds that the buffer retains a partially typed generic input request before returning it to the application. This argument allows the \[ESC\] key to be returned to those applications that are looking for it on terminals where the \[ESC\] key begins function-key sequences. You can specify a value of 0 through 25 seconds for \( \text{number} \). The value 0 disables the timeout. The default value is 5 seconds.

  **Note:** This argument is available only to window terminal users.

**Explanation**

The `set_terminal_parameters` command sets the parameters for the terminal attached to your process’s `default_input` port. The parameter settings take effect immediately for the current terminal, but only for the duration of the current process. When the port closes, the channel or subchannel reverts to the default parameters specified in the device table or specified by the `update_channel_info` command.

The `set_terminal_parameters` command processes arguments in the following order: `-terminal_type` is first, `-setup` is next, and the other arguments follow. You can specify the `-terminal_type`, `-setup`, and other arguments on the same command line.

When you invoke the display form of this command, the operating system displays some of the current values of the parameters.

Pressing the key that invokes the `INTERRUPT` function when you are editing or using the forms processor is equivalent to logging in when your process is at command level. See the Preface of this manual for information about how to invoke the `INTERRUPT` function for your terminal.

If you are a window terminal user, you can specify additional arguments in the display form of the `set_terminal_parameters` command. These arguments are `-max_typeahead_lines`, `-break_action`, `-wait_cursor_format`, `-clear_reused_screen_pages`, `-break_to_wmgr`, `-initial_overlay_on`, `-cancel_doesnt_abort`, `-return doesnt_unpause`, and `-return_doesnt_tab`.

  **Note:** You cannot specify the window-terminal specific arguments in command line form or in a macro such as `start_up.cm`. 
You can invoke `set_terminal_parameters` when you use the debugger, but the results are unpredictable.

**Examples**

The following command switches a V105 terminal into 132-column mode.

```
set_terminal_parameters -setup 132x24
```

**Related Information**

For more information about the characteristics of your terminal device, see the description of the `display_device_info` and `display_terminal_parameters` commands in this manual. For more information on window terminals, see the *Window Terminal User’s Guide* (R256).
set_text_file

Purpose
This command sets the default character set and shift mode of an existing file.

Display Form

<table>
<thead>
<tr>
<th>Command Line Form</th>
</tr>
</thead>
</table>
| set_text_file file_names
  [-character_set character_set]
  [-shift_mode shift_mode]
  [-force] |

Arguments
- **file_names** Required
  The name of a file.
- **-character_set character_set** [CYCLE]
  Assigns one of the following default character sets to the file.
  - none
  - ascii
  - latin_1
  - latin_9
  - kanji
  - katakana
  - hangul
  - simplified_chinese
  - chinesel
  - chinese2
  - user_dbcs

By default, a value of none is assigned to the file. Specify a character set only for a fixed, relative, or sequential file.
**set_text_file**

- **-shift_mode** *shift_mode*  
  Specifies the shift combinations allowed in the file. The values for *shift_mode* are single, locking, all, and none. By default, both single- and locking-shift combinations (all) are allowed. The -shift_mode argument is ignored if the value of *character_set* is none.

- **-force**  
  Changes the shift mode and default character set of a non-empty file. This allows you to set file attributes on files predating this release of the operating system, to support multiple character sets in such files. The *set_text_file* command performs no validation or conversion of the file.

Set the character set attribute to the character set of the file. For best performance, set the shift mode to single. For most efficient use of disk space, set the shift mode to all (both single and locking shifts allowed). If embedded single or locking shifts are present in the file, data may not be interpreted correctly. In such cases, follow the conversion steps described in the Explanation section to guarantee the data’s correctness.

By default, you can set these attributes only for an empty file.

**Explanation**

The *set_text_file* command sets the default character set and shift mode of an existing file.

Character sets that are supported for fixed, relative, or sequential files include ASCII, Latin alphabet No. 1, Latin alphabet No. 9, kanji, katakana, hangul, simplified Chinese, Chinese1, Chinese2, and a user-defined double-byte character set. Indexes on files having one of these default character sets are allowed only if the file’s shift mode allows no shifts; indexes are not allowed for files with a multiple-byte default character set.

The default character set and shift mode of a file are attributes used by file and I/O services to store and present text file data in a compatible format. Changing these attributes changes the way data is interpreted, so using this command for non-empty files, with the -force argument, may cause unexpected results. Use -force only if you have specific knowledge of the actual contents of a file and can ensure valid results. Otherwise, a file whose contents disagree with its new character set and shift mode settings is misinterpreted when accessed; you may be unable to print it or perform other operations on it. If this situation occurs, immediately reset the attributes to the original values and convert the file using the *convert_text_file* command. You cannot return a file whose attributes have been erroneously set with this command to its previous state once you have written to it.
You can safely set any file to binary by specifying `-character_set none` and `-shift_mode none`. When you set a non-empty file to something other than binary, however, set the character set to the character set representing the majority of the file’s data. This minimizes translation during file operations. The `set_text_file` command itself performs no actual translation or conversion. However, changing the character set of a text file containing unshifted right graphic set characters may alter the semantics of those characters, since they are interpreted as characters according to the new default character set and shift mode. If the semantics of the file could change as a result of changing the file’s attributes, use `convert_text_file` to convert the data to conform with the file’s new text attributes.

If the shift mode is `locking` or `all`, file data is stored as compactly as possible at the expense of I/O execution speed. The `-shift_mode` argument is ignored if a file’s `character_set` value is `none`. Changing a file’s default character set to `none` from one of the other values effectively converts it back to binary format.

### Access Requirements
To set a file’s default character set and shift mode, you need modify access to its containing directory.

### Examples
The following command sets the text file attributes of a relative file named `European_report`.

```
set_text_file European_report -character_set latin_1
   -shift_mode single
```

The default character set becomes Latin alphabet No. 1, and the shift mode allows single shifts.

If no shifts are allowed, the file can only contain one character set as specified with the character set attribute.

### Related Information
See the descriptions of the `create_file` command for information on creating a text file, and the `convert_text_file` command for information on converting the contents of an existing text file to conform to a new default character set or shift mode.
**set_time_zone**

**Purpose**
This command sets the time zone for your current process. The operating system changes the time of day in your process to correspond to the new time zone.

**Display Form**

```
set_time_zone
```

**Command Line Form**

```
set_time_zone [time_zone] [difference] [-daylight]
```

**Arguments**

- **time_zone**
  A code for the new time zone. **Required**

- **difference**
  The difference in minutes between the new time and Greenwich Mean Time (GMT). If you set the zone to one of the predefined codes, you do not need to supply this value. If the new `time_zone` is not one of the predefined time zones, you must provide a `difference` value.

- **-daylight**
  Sets the daylight time zone flag. By default, the daylight time zone flag is set to off. **[CYCLE]**

**Explanation**

The `set_time_zone` command sets the one, two-, or three-letter code for the time zone of your current process and adjusts the time in your process to the new time zone.

Table 2-33 lists the time zones that OpenVOS supports, as well as the corresponding differences from GMT.
Table 2-33. Time Zones Supported in OpenVOS

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
<th>Difference from GMT (in Minutes)</th>
<th>Implements Daylight Savings</th>
</tr>
</thead>
<tbody>
<tr>
<td>adt</td>
<td>Atlantic Daylight</td>
<td>-180</td>
<td>yes</td>
</tr>
<tr>
<td>ast</td>
<td>Atlantic Standard</td>
<td>-240</td>
<td>no</td>
</tr>
<tr>
<td>at</td>
<td>Azores</td>
<td>-120</td>
<td>no</td>
</tr>
<tr>
<td>bst</td>
<td>British Summer</td>
<td>60</td>
<td>yes</td>
</tr>
<tr>
<td>bt</td>
<td>Baghdad</td>
<td>180</td>
<td>no</td>
</tr>
<tr>
<td>cad</td>
<td>Central Australia Daylight</td>
<td>630</td>
<td>yes</td>
</tr>
<tr>
<td>cas</td>
<td>Central Australia Standard</td>
<td>570</td>
<td>no</td>
</tr>
<tr>
<td>cdt</td>
<td>Central Daylight</td>
<td>-300</td>
<td>yes</td>
</tr>
<tr>
<td>cet</td>
<td>Central European</td>
<td>60</td>
<td>no</td>
</tr>
<tr>
<td>chd</td>
<td>China Daylight</td>
<td>540</td>
<td>yes</td>
</tr>
<tr>
<td>cht</td>
<td>China Time</td>
<td>480</td>
<td>no</td>
</tr>
<tr>
<td>cst</td>
<td>Central Standard</td>
<td>-360</td>
<td>no</td>
</tr>
<tr>
<td>ead</td>
<td>East Australia Daylight</td>
<td>660</td>
<td>yes</td>
</tr>
<tr>
<td>eas</td>
<td>East Australia Standard</td>
<td>600</td>
<td>no</td>
</tr>
<tr>
<td>edt</td>
<td>Eastern Daylight</td>
<td>-240</td>
<td>yes</td>
</tr>
<tr>
<td>eet</td>
<td>Eastern Europe</td>
<td>120</td>
<td>no</td>
</tr>
<tr>
<td>est</td>
<td>Eastern Standard</td>
<td>-300</td>
<td>no</td>
</tr>
<tr>
<td>fst</td>
<td>French Summer</td>
<td>120</td>
<td>yes</td>
</tr>
<tr>
<td>fwt</td>
<td>French Winter</td>
<td>60</td>
<td>no</td>
</tr>
<tr>
<td>gmt</td>
<td>Greenwich Mean</td>
<td>0</td>
<td>no</td>
</tr>
<tr>
<td>gst</td>
<td>Greenland Standard</td>
<td>-180</td>
<td>no</td>
</tr>
<tr>
<td>hdt</td>
<td>Hawaii-Aleutian Daylight</td>
<td>-540</td>
<td>yes</td>
</tr>
<tr>
<td>hfe</td>
<td>Heure Francaise d’Ete</td>
<td>120</td>
<td>yes</td>
</tr>
<tr>
<td>hfh</td>
<td>Heure Francaise d’Hiver</td>
<td>60</td>
<td>no</td>
</tr>
<tr>
<td>hkt</td>
<td>Hong Kong</td>
<td>480</td>
<td>no</td>
</tr>
<tr>
<td>hst</td>
<td>Hawaii-Aleutian Standard</td>
<td>-600</td>
<td>no</td>
</tr>
<tr>
<td>ist</td>
<td>Indian Standard</td>
<td>330</td>
<td>no</td>
</tr>
</tbody>
</table>
Table 2-33. Time Zones Supported in OpenVOS (Continued)

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
<th>Difference from GMT (in Minutes)</th>
<th>Implements Daylight Savings</th>
</tr>
</thead>
<tbody>
<tr>
<td>jst</td>
<td>Japan Standard</td>
<td>540</td>
<td>no</td>
</tr>
<tr>
<td>jt</td>
<td>Java</td>
<td>450</td>
<td>no</td>
</tr>
<tr>
<td>kdt</td>
<td>Alaska Daylight</td>
<td>-540</td>
<td>yes</td>
</tr>
<tr>
<td>kst</td>
<td>Alaska Standard</td>
<td>-600</td>
<td>no</td>
</tr>
<tr>
<td>mas</td>
<td>Malaysia Standard</td>
<td>480</td>
<td>no</td>
</tr>
<tr>
<td>mdt</td>
<td>Mountain Daylight</td>
<td>-360</td>
<td>yes</td>
</tr>
<tr>
<td>mes</td>
<td>Middle Europe Summer</td>
<td>120</td>
<td>yes</td>
</tr>
<tr>
<td>met</td>
<td>Middle Europe</td>
<td>60</td>
<td>no</td>
</tr>
<tr>
<td>mew</td>
<td>Middle Europe Winter</td>
<td>60</td>
<td>no</td>
</tr>
<tr>
<td>mst</td>
<td>Mountain Standard</td>
<td>-420</td>
<td>no</td>
</tr>
<tr>
<td>ndt</td>
<td>Newfoundland Daylight</td>
<td>-150</td>
<td>yes</td>
</tr>
<tr>
<td>nst</td>
<td>Newfoundland Standard</td>
<td>-210</td>
<td>no</td>
</tr>
<tr>
<td>nt</td>
<td>Nome</td>
<td>-660</td>
<td>no</td>
</tr>
<tr>
<td>nzd</td>
<td>New Zealand Daylight</td>
<td>780</td>
<td>yes</td>
</tr>
<tr>
<td>nzs</td>
<td>New Zealand Standard</td>
<td>720</td>
<td>no</td>
</tr>
<tr>
<td>pdt</td>
<td>Pacific Daylight</td>
<td>-420</td>
<td>yes</td>
</tr>
<tr>
<td>phs</td>
<td>Philippine Standard</td>
<td>480</td>
<td>no</td>
</tr>
<tr>
<td>pst</td>
<td>Pacific Standard</td>
<td>-480</td>
<td>no</td>
</tr>
<tr>
<td>sdt</td>
<td>Samoa Daylight</td>
<td>-660</td>
<td>yes</td>
</tr>
<tr>
<td>sis</td>
<td>Singapore Standard</td>
<td>480</td>
<td>no</td>
</tr>
<tr>
<td>sst</td>
<td>Samoa Standard</td>
<td>-600</td>
<td>no</td>
</tr>
<tr>
<td>tpe</td>
<td>Taiwan Standard</td>
<td>480</td>
<td>no</td>
</tr>
<tr>
<td>tst</td>
<td>Thailand Standard</td>
<td>420</td>
<td>no</td>
</tr>
<tr>
<td>ut</td>
<td>Universal</td>
<td>0</td>
<td>no</td>
</tr>
<tr>
<td>wad</td>
<td>West Australia Daylight</td>
<td>540</td>
<td>yes</td>
</tr>
<tr>
<td>was</td>
<td>West Australia Standard</td>
<td>480</td>
<td>no</td>
</tr>
<tr>
<td>wat</td>
<td>West Africa</td>
<td>-60</td>
<td>no</td>
</tr>
</tbody>
</table>
The column labeled **Difference** is the difference between the listed time and Greenwich Mean Time, in minutes. For example, a clock set to Eastern Standard Time reads 300 minutes earlier than a clock set to Greenwich Mean Time.

If the time difference is for a daylight-savings time zone, then the value of the **difference** argument is the time difference with Greenwich Mean Time (GMT).

You can give a predefined time zone code in either uppercase or lowercase letters.

When a process is created, its time zone is set to the time zone of the current module. If you are in a subprocess, and you issue the `set_time_zone` command, the time zone for both processes is reset.

### Examples

**Example 1.**
The following command sets the time zone to Eastern Standard Time.

```
set_time_zone est
```

The operating system automatically sets the difference for this zone, -300 minutes.

**Example 2.**
Assume that you want to set the time zone to 11 hours before Greenwich Mean Time, and that you want to represent the new time zone by the code `jst`. Use the following command to make this change.

```
set_time_zone jst -660
```

Note that the time difference is given in minutes, not hours.

Setting the time zone for your process affects only how times are displayed in your process. It has no effect on how the operating system internally stores times, such as the time-modified attribute of a file.

### Table 2-33. Time Zones Supported in OpenVOS (Continued)

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
<th>Difference from GMT (in Minutes)</th>
<th>Implements Daylight Savings</th>
</tr>
</thead>
<tbody>
<tr>
<td>wib</td>
<td>Indonesia Standard</td>
<td>420</td>
<td>no</td>
</tr>
<tr>
<td>ydt</td>
<td>Yukon Daylight</td>
<td>-480</td>
<td>yes</td>
</tr>
<tr>
<td>yst</td>
<td>Yukon Standard</td>
<td>-540</td>
<td>no</td>
</tr>
<tr>
<td>z</td>
<td>Zulu (Universal)</td>
<td>0</td>
<td>no</td>
</tr>
</tbody>
</table>

The column labeled **Difference** is the difference between the listed time and Greenwich Mean Time, in minutes. For example, a clock set to Eastern Standard Time reads 300 minutes earlier than a clock set to Greenwich Mean Time.
**Purpose**
This command puts the process that issues the command to sleep for a specified period, after which the operating system reactivates the process.

**Note:** The OpenVOS GNU Tools layered product also supplies a command named `sleep`. When the GNU Tools product is installed and used as directed, the GNU Tools version of this command is found before the OpenVOS version, depending upon command library paths. Since the OpenVOS and GNU Tools commands behave differently, you can use the alternate name (`vsleep`) to ensure that you invoke the OpenVOS version of the command.

**Display Form**
```
------------------------------------ sleep -------------------------------------
-hours: [ ]
-minutes: [ ]
-seconds: [ ]
-until: [ ]
-forever: no
```

**Command Line Form**
```
sleep [-hours hours]

   [-minutes minutes]
   [-seconds seconds]
   [-until date_time]
   [-forever]
```

**Arguments**
- `-hours hours`
  Puts the issuing process to sleep for the specified number of hours. By default, the value of `hours` is 0.

- `-minutes minutes`
  Puts the issuing process to sleep for the specified number of minutes. By default, the value of `minutes` is 0.
In this context, the string must be enclosed in apostrophes. See Chapter 1, “OpenVOS Command Functions” for examples of acceptable date/time input strings.

-forever

Puts the issuing process to sleep for an indefinite period of time.

**Explanation**

The `sleep` command suspends a process for a period of time. The suspended process is the process that issues the command; if you invoke the `sleep` command from command level at your terminal, your login process sleeps. A process is normally suspended to wait for an event to occur.

If you specify one or more of `-hours`, `-minutes`, and `-seconds`, the issuing process sleeps for the period of time given by the values you specify for `hours`, `minutes`, and `seconds`.

If you specify `-until`, the issuing process sleeps until the specified date and time. If the date and time are in the past, the operating system reactivates the process immediately.

If you specify `-forever`, the process is suspended until the break condition is signaled in the process.

Unless you specify a value for one or more arguments, the `sleep` command does not suspend the process.

**Related Information**

For information on reactivating a sleeping process, see the description of the `break_process` command.
sort, vsort

Purpose

This command sorts the records in one or more ASCII text files and merges the sorted files with a set of pre-sorted files, using collating sequences and sort keys specified either in the command or in a sort control file.

Note: The OpenVOS GNU Tools layered product also supplies a command named sort. When the GNU Tools product is installed and used as directed, the GNU Tools version of this command is found before the OpenVOS version, depending upon command library paths. Since the OpenVOS and GNU Tools commands behave differently, you can use the alternate name (vsort) to ensure that you invoke the OpenVOS version of the command.

Display Form

```
------------------------------------- sort -------------------------------------
sort_path_names:  
-merge_path:  
-output_path:  
-exceptions_path:  
-duplicates_path:  
-control:  
-statistics:  no  
-position_1: 1  
-length_1: 32767  
-collation_1: ascending_alphabetical  
-position_2:  
-length_2:  
-collation_2: ascending_alphabetical  
-position_3:  
-length_3:  
-collation_3: ascending_alphabetical  
-position_4:  
-length_4:  
-collation_4: ascending_alphabetical  
-sort_in_memory: 1000  
-work_dir:  
```
Command Line Form

sort { 
  sort_path_names
  -merge_path merge_path_names
  [-output_path output_path_name]
  [-exceptions_path exceptions_path_name]
  [-duplicates_path duplicates_path_name]
  [-control control_path_name]
  [-statistics]
  [-position_1 position_1]
  [-length_1 length_1]
  [-collation_1 collation_1]
  [-position_2 position_2]
  [-length_2 length_2]
  [-collation_2 collation_2]
  [-position_3 position_3]
  [-length_3 length_3]
  [-collation_3 collation_3]
  [-position_4 position_4]
  [-length_4 length_4]
  [-collation_4 collation_4]
  [-sort_in_memory number_of_records]
  [-work_dir [work_directory]]
}

Arguments

- **sort_path_names**
  One or more names or star names of files to be sorted. If you specify more than one file by giving either a star name or multiple file names, you must also specify an output path. In this case, sort generates a sorted file that contains the records of all of the files. You must specify either sort_path_names or merge_path_names.

- **-merge_path merge_path_names**
  Specifies one or more names or star names of files to be merged with any files specified in sort_path_names. If you do not supply a value for sort_path_names, you must also specify a value for output_path_name. For multiple files, sort generates a merged file that contains the records of all of the files. You must specify either sort_path_names or merge_path_names.

- **-output_path output_path_name**
  Specifies the path name of the file that is to contain the sorted and merged records. You can omit this argument only when sort_path_names matches a single file name; in this case, sort replaces the single input file with the sorted file.

- **-exceptions_path exceptions_path_name**
  Specifies the file to which sort is to write all records that do not have the form required for the sort. The command appends the extension .sort_exc to the path name that you specify. If you omit this argument and output_path_name is the same as sort_path_names, the command creates a default exceptions file when sort_path_names matches a single file and gives it the name output_path_name.
with the extension .sort_exc appended. This reduces the chance of accidental loss of
records. If you do not use -exceptions_path, and either specify a star name that
matches more than a single file for sort_path_names, or output_path_name does
not match the first sort path name, the command does not create a default exceptions
file. In this case, exceptional records are discarded.

- duplicates_path duplicates_path_name

  Specifies the file to which sort is to write records whose sort key value or values
  match the values of another record. The command writes to the output file only the first
  record found in the set of records having a single sort key value. If you do not give the
  suffix .sort_dup in duplicates_path_name, the operating system adds it to the
  name of the file when creating it. By default, sort writes all of the records in the set
to the output file in the order that it finds them.

- control control_path_name

  Specifies the sort control file. The sort control file defines the keys to be used in the
  sort. See the explanation of the sort control file later in this section. If you do not supply
  it in control_path_name, the operating system adds the suffix .dd to the name of
  the file when searching for it.

- statistics

  Displays the sort’s statistics on your terminal as the sort proceeds. By default, the
  operating system does not display the statistics.

- position_1 position_1

  Specifies the position in a record of the substring to be used as the primary sort key. By
default, the first position in the record is the primary sort key.

- length_1 length_1

  Specifies the length of the substring to be used as the primary sort key. By default, the
  value is 32,767.

- collation_1 collation_1

  Specifies the collating sequence for the primary sort key that the command is to use.
  There are eight possible values.

  ascending_alphabetical  descending_alphabetical
  ascending_ascii         descending_ascii
  ascending_numerical     descending_numerical
  ascending_ebcdic        descending_ebcdic

  By default, the collating code is ascending_alphabetical.

- position_2 position_2 -length_2 length_2
- position_3 position_3 -length_3 length_3
- position_4 position_4 -length_4 length_4

  Specify the positions in a record and the lengths of the secondary, tertiary, and
  quaternary sort keys.
**specify the collating sequences the command is to use with the secondary, tertiary, and quaternary sort keys. By default, the collating sequence is ascending alphabetical.**

- **-sort_in_memory number_of_records**
  Specifies the number of records to be sorted in memory at one time. The command sorts faster for a higher number. You must specify at least 100 records and at most 10,000 records. By default, the command sorts 1,000 records in memory at one time.

- **-work_dir [work_directory]**
  Specifies that the temporary work files created when sorting the designated files reside in a specified directory. The directory can be on the same or another disk of the current module. If you specify -work_dir but do not specify work_directory, sort uses the current directory. By default, the command uses the process directory of the current module.

**Explanation**

The **sort** command sorts the records in a file or set of files and writes the sorted records to an output file.

Specify the file(s) to be sorted in the argument **sort_path_names**. The arguments can be devices. The **sort** command also merges the records in a set of files; specify the files with -merge_path.

The **sort** command writes the sorted records to the output file specified in -output_path. You can omit this only when **sort_path_names** matches a single file name; in this case, the operating system overwrites the input file with the sorted file.

If you do not specify either -output_path or a control file (that is, a single file for **sort_path_names**), the command creates a new file with the same organization and characteristics as the specified file. This new file replaces the sorted file. See the Examples section for examples.

You can use the control file to specify many characteristics of the output file; if you use the control file, the command does not use attributes of the sorted file.

As a result of the sort, separate-key indexes are truncated, and embedded key indexes are modified and will be correct.

**Note:** The **sort** command may not create 64-bit stream files as expected.

**Specifying Sort Keys**

The **sort** command sorts records using keys specified either in the command or in a sort control file. You can specify up to four keys in the command, or up to 32 keys in a sort control file. In the command or control file, specify the positions, the lengths, and the collating sequences of the keys.
In a sort command, you must specify a primary key if you specify a secondary key, both of these if you specify a tertiary key, and all keys if you specify a quaternary key.

The primary key determines the sorted order of two records if their primary key values differ; otherwise, the secondary key determines the sorted order when the primary key values of the records are equal and the secondary key values differ; and so on.

If any record to be sorted is shorter than the minimum record size implied by a key specification, the sort command rejects the record and writes it to the exceptions file or device, if one was specified. In particular, if the last byte of the record plus 1 is less than the starting point of the key, the record is placed in the exceptions file. For example, assume you have four records.

```
Red
Yellow
Green
Blue
```

Sorting with -position_1 equal to 4 does not place Red in the exceptions file. Sorting with -position_1 equal to 5 places Red in the exceptions file. Blue is not placed in the exceptions file.

To remove blank lines from a file, you must specify a value of 2 for -position_1.

**Specifying Collation Codes**

The ascending collation codes sort records into ascending order, and the descending codes sort records into descending order.

When you select either of the two ASCII collation codes, the sort command interprets the data in the keys as ASCII characters and sorts the records according to the ASCII collating sequence.

When you select either of the two alphabetical collation codes, the sort maps all of the alphabetical characters in a key to a common case before sorting them, using the actual values to resolve ties. The sort command trims leading and trailing spaces from keys and reduces multiple spaces to one space in key values so that an alphabetical sort puts records into alphabetical order.

When you select either of the two numerical collation codes, the key fields can contain only characters that represent valid optionally signed constants and leading and trailing spaces. The sort command trims the spaces from the key values. The resulting values are sorted as numbers so the records are put into ordinary numerical order.

When you select either of the two EBCDIC collation codes, the sort command sorts the keys according to the EBCDIC collating sequence. The records, nonetheless, must be ASCII records.

**Specifying a Working Directory**

If you specify -work_dir, the command creates the temporary work files it needs in a specified directory. If you want to sort a large file, and the current disk is relatively full, you can use work space on an emptier disk in the current module. (Use the display_disk_info
command to determine how much disk space is in use.) If you omit -work_dir, the command creates the work files in the process directory of the current module. In either case, if the directory in which the work files are to be created is located on a disk with inadequate space, the command aborts the sort, displays an error message, and returns you to command level.

Sort Control Files

Instead of supplying the sorting parameters as arguments in a command, you can specify them in a sort control file. A sort control file names and describes the fields of a record in the file or files to be sorted and merged. Next, it specifies the key or keys, which are simply the fields in the records, used to sort the records. Finally, it can specify indexes to be created for the sorted output file.

All data in a sort control file must be word-aligned, except pictured data, unaligned non-varying character-string data, and unaligned bit-string data.

The following is an example of a sort control file.

```plaintext
field:
    name char(32) varying,
    age decimal(2),
    height decimal(2),
    weight decimal(3),
    modes bit(8),
    attributes char(64) varying;
key:
    age descending,
    name space_suppress,
    attributes order(user_collation) substr(5,10);  
user_collation:
    'aAbBcCdCdEfFfGgHHiIjJkJkKlLmMnNoOpPqQrRsStTuUvVwWxXyYzZ';
index:
    name no_duplicates order(alphabetical);  
organization:
    relative;
end;
```

In the preceding example, the fields supply the following information:

- **field** defines the records in the file to be sorted
- **key** specifies the fields of the records to be used as keys for sorting
- **user_collation** specifies the sequence in which the records are to be collated
- **index** designates an index or indexes that the operating system is to create when it makes the output file
organization specifies the organization of the sorted output file. Extended sequential files are denoted by \texttt{sequential} \([N]\), where \(N\) represents one of the various types of extended sequential files. See the description of the \texttt{create_file} command for more information about extended sequential files.

- \texttt{end} is the last statement in this sort control file

If \texttt{sort\_path\_names} is an OpenVOS COBOL file, you must specify \texttt{cobol} using the \texttt{language} statement. The following is an example of an OpenVOS COBOL sort control file.

```cobol
language: cobol;
field:
  name pic 'x(32)',
  age pic '9(02)',
  height pic '9(02)',
  weight pic '9(03)',
  modes comp-4,
  attributes pic 'x(64)';
key:
  age descending,
  name space_suppress,
  attributes order(user_collation) substr(5,10);
user_collation:
  'aAbBcCdDeEfGhHiIjJkKlLmMnNoOpPqRsStTuVvWxXyYzZ';
index:
  name no_duplicates order(alphabetical);
organization:
  relative;
end;
```

**Using the field Statement**

Use the \texttt{field} statement to declare the fields in a record in the file to be sorted. The declaration must name the field and declare its data type. All bytes from the beginning of the record to a position that is guaranteed to contain all of the keys used in the sort must be declared; you may need to add fields as fillers between the fields you want to use as keys.

A field statement has the following general form.

```cobol
field:field_descriptor...;
```

The \texttt{field\_descriptor} is the name of the field followed by the data type of the field, that is, \texttt{field\_name\ data\_type}. When you give several \texttt{field\_descriptor} terms in one \texttt{field} statement, you must separate them with commas. A field name can contain up to 32 characters, letters, and numbers; the first character must be a letter.
The following table shows the possible OpenVOS PL/I field data types.

<table>
<thead>
<tr>
<th>OpenVOS PL/I Data Type</th>
<th>Defaults</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>bit [ (N) ]</td>
<td>N = 1</td>
<td>N bits</td>
</tr>
<tr>
<td>bit [ (N) ] aligned</td>
<td>N = 1</td>
<td>2*ceiling(N/16) bytes</td>
</tr>
<tr>
<td>char [ (N) ]</td>
<td>N = 1</td>
<td>N bytes</td>
</tr>
<tr>
<td>char [ (N) ] varying</td>
<td>N = 1</td>
<td>N+2 bytes</td>
</tr>
<tr>
<td>char [ (N) ] aligned</td>
<td>N = 1</td>
<td>2*ceiling(N/2) bytes</td>
</tr>
<tr>
<td>char [ (N) ] varying aligned</td>
<td>N = 1</td>
<td>2*ceiling(N/2)+2 bytes</td>
</tr>
<tr>
<td>fixed binary [ (P) ]</td>
<td>P = 15</td>
<td>2*ceiling(P/15) bytes</td>
</tr>
<tr>
<td>float binary [ (P) ]</td>
<td>P = 24</td>
<td>4*ceiling(P/24) bytes</td>
</tr>
<tr>
<td>fixed decimal [ (P, Q) ]</td>
<td>P = 9, Q = 0</td>
<td>4*ceiling(P/9) bytes</td>
</tr>
<tr>
<td>float decimal [ (P) ]</td>
<td>P = 7</td>
<td>4*ceiling(P/7) bytes</td>
</tr>
<tr>
<td>picture 'picture-string'</td>
<td></td>
<td>Length of picture-string bytes</td>
</tr>
</tbody>
</table>

The variables \( N, R \) and \( Q \) are unsigned integers. The operating system uses the default values shown in the table if you do not specify a value for these variables. The size of the corresponding fields is also shown in the table.

The following table shows the possible OpenVOS COBOL field data types.

<table>
<thead>
<tr>
<th>OpenVOS COBOL Data Type</th>
<th>Defaults</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>display pic 'x(N)'</td>
<td>N = 1</td>
<td>N bytes</td>
</tr>
<tr>
<td>display-2 pic 'x(N)'</td>
<td>N = 1</td>
<td>2*ceil(N/2)+2 bytes</td>
</tr>
<tr>
<td>display sync left pic 'x(N)'</td>
<td>N = 1</td>
<td>2*ceil(N/2)</td>
</tr>
<tr>
<td>comp-1</td>
<td>P = 24</td>
<td>4 bytes</td>
</tr>
<tr>
<td>comp-2</td>
<td>P = 53</td>
<td>8 bytes</td>
</tr>
<tr>
<td>comp-3 pic 's9(i)V9(f)'</td>
<td>None</td>
<td>(i+f)/2 + 1</td>
</tr>
<tr>
<td>comp-4</td>
<td>P = 15</td>
<td>2 bytes</td>
</tr>
<tr>
<td>comp-5</td>
<td>P = 31</td>
<td>4 bytes</td>
</tr>
<tr>
<td>comp-6 pic 's9(i)V9(f)'</td>
<td>None</td>
<td>4 bytes if i+f &lt;= 9, 8 bytes otherwise</td>
</tr>
</tbody>
</table>

The variable \( i \) represents the number of integral digits; the variable \( f \) represents the number of fractional digits for \texttt{comp-6} fields.

At least one \texttt{field} statement is required in a sort control file, but you can give more than one. The sequence of fields within a \texttt{field} statement and the sequence of \texttt{field} statements define the sequence of fields in a record. For example, the following \texttt{field} statements define...
the same sequence of record fields as the single field statement shown in the earlier example.

```plaintext
field: name char(32) varying;
field: age decimal(2);
field: height decimal(2);
field: weight decimal(3);
field: modes bit(8);
field: attributes char(64) varying;
```

The order of the field statements defines the order of the fields in the records.

The keywords field and fields are interchangeable.

**Using the key Statement**

A key statement tells sort which collating sequence and field to use when sorting. At least one of each is required. A key statement contains a field name, declared in a field statement, a definition of a substring of the field to be used as the key, an indication of how to treat space characters in the key, and the collating sequence to use when sorting the keys. A key statement has the following general form.

```plaintext
key: key_descriptor ...
```

A key_descriptor consists of the name of a field declared in a field statement and one or more options. If you specify more than one key_descriptor, separate them with a comma.

A key_descriptor has the following general form.

```
[space_suppress
no_space_suppress]
[substr(P[,L])]
[ascending
descending]
[order(ascii)
order(alphabetical)
order(alphabetical)
order(numerical)
order(ebcdic)
order(user_collation)]
```

The options space_suppress and no_space_suppress tell the sort command whether to trim leading and trailing spaces and reduce multiple spaces inside a field to one space. You can specify only one of these options in a key descriptor. If you omit both in a key descriptor, the value is space_suppress when the collating sequence is alphabetical or numerical, and no_space_suppress otherwise.

The option substr(P[,L]) defines a substring of the field as the key. The integer P is the position of the first character or bit in the substring, and the optional integer L is the length of the substring. When you omit L, the last character or bit of the field is the last character or bit of the substring. The sort command extracts a substring from a record before suppressing spaces. If you omit a substr option from a key descriptor, the entire field is used as the key.
The options ascending and descending specify the direction of the sort order. If you omit both of these options from a key descriptor, the value is ascending.

The options order(ascii), order(alphabetical), order(numerical), order(ebcdic), and order(user_collation) specify the collating sequence used to sort the keys. The collating sequence codes have the same meaning as the previously described codes you can give in a sort command. You can specify one of these collating sequences only when the key is a character string. The code order(user_collation) tells the sort command to use the collating sequence you give in a user_collation statement. If you omit an order option, the value is alphabetical.

At least one key statement is required in a sort control file, but you can give more than one. The sequence of fields within a key statement and the sequence of key statements define the sequence of keys used to sort the records. For example, the following key statements define the same sequence of keys as the single key statement shown in the earlier example.

```
key:       age  descending;
key:       name  space_suppress;
key:       attributes order(user_collation) substr(5,10);
```

The keywords key and keys are interchangeable.

If you specify an order(user_collation) option in a key descriptor, you must give one and only one user_collation statement in the sort control file. The user_collation statement has the following general form.

```
user_collation: 'character_string';
```

*character_string* is the collating sequence.

**Using the index Statement**

An index statement tells the operating system to create an index or indexes for the output file. It contains the name of the index, which must be the name of a field, an option specifying whether duplicate keys are allowed in the index, and the collating sequence used to sort the index. The index statement has the following general form.

```
index: index_descriptor ...;
```

An index_descriptor consists of the name of a field declared in a field statement and one or more options. The field must be declared as a character string (char for OpenVOS PL/I, display for OpenVOS COBOL). If you specify more than one index_descriptor, separate them with a comma.

The index_descriptor has the following general form.

```
field_name [ duplicates | no_duplicates ] [ order(ascii) | order(alphabetical) | order(numerical) ]
```
The options duplicates and no_duplicates determine whether duplicate keys are allowed in the index. You can give only one; if you omit both, the value is no_duplicates.

The options order(ascii), order(alphabetical), and order(numerical) define the collating sequence used to sort the keys in the index. You can give only one; if you omit all three, the value is order(ascii).

An index statement is not required in a sort control file. You can give more than one. Also, you can use the keyword indexes for the keyword index.

Using the organization Statement
An organization statement specifies the organization of the output file. If you omit this statement, the organization of the output file is the same as the organization of the input file.

Using the language Statement
You must use the language statement to specify OpenVOS COBOL as the language of the field descriptions in the sort control file. If you omit this statement, the sort command defaults to OpenVOS PL/I.

Examples
In the following example, assume that seq_file is a sequential file with dynamic extents of 8 and implicit locking set.

```
sort seq_file
```

After the sort, seq_file has the same file characteristics as it did before the sort.

In the following example, assume that rel_file is a relative file with a maximum record size of 10 and an embedded key index.

```
sort rel_file file_a -output_path new_file
```

After you specify the preceding command, new_file has the same characteristics as rel_file, including its index. If new_file had existed before the sort, the command would delete it and replace it with a new file with rel_file's characteristics.

Related Information
For more information on creating indexes, see the description of the create_index command. For more information on OpenVOS COBOL data types, see the VOS COBOL Language Manual (R010). For more information on PL/I data types, see the OpenVOS PL/I Language Manual (R009).
**start_logging**

**Purpose**

This command starts logging the I/O traffic through a port to a log file or I/O device.

**Display Form**

```
path_name: [port_name]
-append: no
-file: no
-record: no
-reads: yes
-writes: yes
-opcode: no
```

**Command Line Form**

```
start_logging path_name
  [port_name]
  -append
  -file
  -record
  -no_reads
  -no_records
  -opcode
```

**Arguments**

- **path_name**
  
  The log file or I/O device.

- **port_name**
  
  The port to be logged. By default, the start_logging command logs the first five ports. See the Explanation section for more information.

- **-append**
  
  Appends the logged records to the end of the log file. By default, the operating system truncates the log file before writing to it.

- **-file**
  
  Includes with a logged record the full path name of the file to which the logged record was written, or from which the logged record was read. By default, the operating system does not log the name of the file.
The `start_logging` command begins copying into a log file or to an I/O device the records that pass through a port. Note that forms-related input is terminal specific and therefore not logged.

The `path_name` argument specifies the log file or I/O device to which the logged records are written. If you specify a file that does not exist, the operating system creates a sequential file with the name `path_name`. If `path_name` is an I/O device that pauses after a given number of lines of display, your own process pauses after that number of lines has been displayed on the device, and you must respond to the device’s pause message to continue logging.

The `port_name` argument specifies the port that is to be logged. If you omit this argument, the operating system logs the first five ports. However, if you specify the `attach_default_output` command before you specify the `start_logging` command with the default port name value of `default_output`, the operating system logs only the `default_output` port.

If you specify the `attach_default_output` and `detach_default_output` commands within a logging session, the operating system creates a zero-length file. For example:

```
start_logging log1.log
attach_default_output log.ado
list_port_attachments
detach_default_output
stop_logging
```

To create the expected log results, specify `attach_default_output` and `detach_default_output` outside the logging session. For example:

```
attach_default_output log.ado
start_logging log1.log
list_port_attachments
stop_logging
detach_default_output
```
Specifying `-append` appends logged records to the log file; otherwise, the file is truncated first.

Specifying `-file` includes with the logged copy the path name of the source file or destination file of a record passing through the port.

Specifying `-record` logs the record number and, when the access mode is indexed, the key used to access each record.

Specifying `-opcode` includes a code with each logged record to indicate the action performed on the record.

The `-no_reads` and `-no_writes` arguments allow you to select the type of I/O access through the port that is logged.

A file can simultaneously be the log file of more than one port in your process.

**Examples**

**Example 1.**
Use the following command to start logging your `default_output` port to the file `log_file.90-04-10` in the current directory.

```
start_logging log_file.90-04-10
```

If the file `log_file.90-04-10` does not exist, the operating system creates it. The log file will contain copies of the records written through your `default_output` port.

**Example 2.**
Suppose that the current directory is `%s1#d02>Sales>Smith`, and that the only file it contains is a small sample file named `file_A`. The following sequence of commands logs a list of the files in the current directory, and produces a log file named `log_file`.

```
start_logging log_file
list -all
stop_logging
```
The contents of log_file might look like this.

list

Files: 2, Blocks: 1
w 1 file_A
w 0 log_file

Directories: 1
m 1 directory_A

Links: 0

stop_logging

Example 3.
If you specify -opcode in the command line, the log file contains opcodes for each record logged. If you issue the list command for files only, the contents of log_file might look now like this:

%$1#d02>Sales>Smith>log_file 90-10-12 12:21:30 EST

/****************** s$start_logging ******************/
call s$start_logging(terminal, '%$1#d02>Sales>Smith>log_file', 25, 0);
/****************** s$start_logging ******************/

/***************** s$control ******************/
call s$control(terminal, 224, ***, 0);
/***************** s$control ******************/

call s$control(terminal, 230, ***, 0);

call s$control(terminal, 226, ***, 0);

call s$control(terminal, 226, ***, 0);

call s$control(terminal, 300, 4, '...', 0);

/***************** '...' ******************/
list

/********************* s$control *********************/
call s$control(terminal, 1, ***, 0);

(Continued on next page)
(Continued)

```c
/**************** s$seq_write ******************
call s$seq_write(terminal, 0, '...', 0);
/**************** '...' ******************/

/**************************** s$seq_write ****************************/
call s$seq_write(terminal, 19, '...', 0);
/**************************** '...' ****************************/
Files: 2, Blocks: 1
/************************************************************************/

/**************************** s$seq_write ****************************/
call s$seq_write(terminal, 0, '...', 0);
/**************************** '...' ****************************/

/**************************** s$seq_write ****************************/
call s$seq_write(terminal, 15, '...', 0);
/**************************** '...' ****************************/
w      1 file_A
/************************************************************************/

/**************************** s$seq_write ****************************/
call s$seq_write(terminal, 17, '...', 0);
/**************************** '...' ****************************/
w      0 log_file
/************************************************************************/

/**************************** s$seq_write ****************************/
call s$seq_write(terminal, 0, '...', 0);
/**************************** '...' ****************************/
```

(Continued on next page)
start_logging

(Continued)

/**************************** s$control ******************/
call s$control(terminal, 224, ***, 0);
/****************************

/**************************** s$control ******************/
call s$control(terminal, 230, ***, 0);
/****************************

/**************************** s$control ******************/
call s$control(terminal, 226, ***, 0);
/****************************

/******************** s$seq_read ******************/
call s$seq_read(terminal, 300, 12, '...', 0);
/******************** '...' ******************/
stop_logging
/********************

/**************************** s$control ******************/
call s$control(terminal, 1, ***, 0);
/****************************

/**************************** s$stop_logging ******************/
call s$stop_logging(terminal, 0);
/****************************

Related Information

See also the descriptions of the attach_default_output, detach_default_output, and stop_logging commands.
**start_process**

**Purpose**
This command creates and starts a noninteractive process.

**Display Form**

```
command_line: start_process
-process_name: 
-output_path: 
-priority: 
-privileged: no 
-module: 
-current_dir: current_dir 
-wait: no 
-root: no 
-cpu_limit: 
-memory_pool: default 
```

**Command Line Form**

```
start_process command_line
    [-process_name process_name]
    [-output_path output_path_name]
    [-priority priority_number]
    [-privileged]
    [-module module_name]
    [-current_dir directory_name]
    [-wait]
    [-root]
    [-cpu_limit cpu_time]
    [-memory_pool memory_pool_number]
```

**Arguments**

- **command_line**  
  The command line that the started process is to execute.

- **-process_name process_name**  
  Names the process. By default, `start_process` gives the process a name derived from the first word in `command_line`. If it is a valid path name, the command uses the file-name portion with any suffixes removed. If the resulting name is longer than 32 characters, the command uses the first 32 characters. If the resulting file name contains an apostrophe or is invalid, the command uses the name `start_process`. 
- `output_path output_path_name`
  Specifies an output file or device to attach to the default_output port of the started process. If you specify #null for output_path_name, the output is discarded. By default, start_process attaches the port to a file in your current directory. The command derives the name of the file from the first word in the command_line argument. If it is a valid path name, the command uses the file-name portion with any suffixes removed and append the suffix .out. If the resulting file name is too long, the command uses the first 28 characters and append the .out suffix. If the resulting file name contains an apostrophe (‘) or is invalid, the command uses the name start_process.out.

- `priority priority_number`
  Specifies the priority of the started process. The range of process priorities is from 0 to 9. Only a privileged user can start a process with a priority higher than the maximum priority of the current process. By default, start_process gives the new process the same priority as your current process. You can use the (process_info) command function to determine the priority of your current process.

- `privileged`
  Starts the process as a privileged process. By default, the process is not privileged. Only a privileged process can start another privileged process.

- `module module_name`
  Specifies the module on which the process is started. By default, start_process starts the process on your module.

- `current_dir directory_name`
  Sets the current directory of the started process to directory_name. By default, start_process uses your current directory. Specifying a new current directory does not change the default location of the output path.

- `wait`
  Finishes executing the commands in command_line before returning your process to command level. By default, start_process immediately returns your process to command level after starting the process.

- `root`
  Starts the process with the user name root.root. By default, this command uses the user name of the current process to start the process. Only a process with a user name of Overseer.System or root.root can specify this argument. This argument is intended for use within the module_start_up.cm file.

- `cpu_limit cpu_time`
  Specifies the upper boundary on the amount of CPU time, in seconds, that the started process can consume. By default, start_process puts no limit on the amount of CPU time the started process can use.

- `memory_pool memory_pool_number`
  When starting a process, this argument specifies the CPU memory pool with which the process will be associated. Allowed values are default and 0 for ftServer modules.
you do not specify a value, the default value is default, which indicates that OpenVOS is using memory pool 0.

**Explanation**

The `start_process` command creates and starts a new process.

The home directory of a started process is the same as the home directory of the starting process. If the home directory contains the startup command macro, the operating system executes it before executing the commands in `command_line`. Abbreviations in the command line `command_line` are expanded using the abbreviations file of the started process, not the starting process.

If you specify `-wait`, the command processor waits until the command or commands in `command_line` finish before starting the next command. You can use `-wait` when the command is in a command macro, and the rest of the macro depends on the results of the started process.

The `-memory_pool` argument specifies the memory pool with which a process will be associated. All ftServer modules have only one memory pool.

All messages sent by the `start_process` command to the system error log contain the date and time on which the process was started.

**Examples**

**Example 1.**
The following command starts a process to compile the OpenVOS COBOL source module `make_reports.cobol` in the current directory.

```
start_process 'cobol make_reports -list'
```

The operating system names the process `cobol`. It creates the default output file `cobol.out` in the current directory.

**Example 2.**
This command starts a process that executes the program `program1.pm`.

```
start_process program1.pm -output_path results
```

The operating system names this process `program1` and creates an output file `results.out` in the current directory for output messages.

**Related Information**

For information on creating an interactive subprocess after your initial login, see the description of the `login` command. For information on how to stop a started process, see the description of the `stop_process` command.
stop_logging

Purpose
This command stops logging a port.

Display Form

```
--------------------------------- stop_logging ---------------------------------
port_name: default_output
```

Command Line Form

```
stop_logging [port_name]
```

Arguments

- **port_name**
  
The port to stop logging. By default, *stop_logging* uses your *default_output* port.

Explanation

The *stop_logging* command tells the operating system to stop logging the port
*port_name* or your *default_output* port.

If the log file or device is not logging any other port, the operating system closes the file or clears the port.

Examples

The following command stops logging the port *accounts_recv* in the current directory.

```
stop_logging accounts_recv
```

Related Information

See also the descriptions of the *attach_default_output*, *detach_default_output*, and *start_logging* commands.
stop_process

Purpose
This command stops a process or set of processes.

Display Form

```
--------- stop_process ---------
process_name: *
-user: current_user
-module: *
-ask: yes
```

Command Line Form

```
stop_process [process_name]

[-user user_name]
[-module module_name]
[-no_ask]
```

Arguments

- **process_name**
The name or star name of a process or set of processes to be stopped. The command stops all of the processes whose names match `process_name`, except for the process issuing the command. If `process_name` is a star name and you do not specify `-no_ask`, the command asks you for confirmation before stopping each process. By default, the operating system stops all processes identified by `-user` or `-module`.

- **-user user_name**
  Specifies the name or star name of a user or set of users. This allows you to stop only the processes named `process_name` that were started by the specified users. By default, the operating system uses your user name. The command does not stop the process from which you issue the command. Your process must be privileged to stop another user's process.

- **-module module_name**
  Specifies the module executing the processes to be stopped. By default, the operating system uses the module executing your login process.
stop_process

- **-no_ask**
  Suppresses the prompt, when you specify a star name for `process_name`, asking whether to stop a process with a matching name. By default, the operating system asks before stopping each process.

**Explanation**

The `stop_process` command stops the processes specified by `process_name` belonging to the user `user_name` on the module `module_name`.

Use `stop_process` to stop a process started by mistake or to stop a process that is looping indefinitely or otherwise in error.

If your process is not privileged, you can only stop processes that have your person name. A privileged process can stop any process except the process issuing the command. (The `stop_process` command cannot stop the process issuing it. See the description of the `logout` command.)

The `stop_process` command terminates a process without creating a `keep` module; to suspend a noninteractive process for debugging purposes, use the `break_process` command.

When the operating system stops a process, it closes all files the process opened that can be closed, detaches all ports the process attached, detaches all events the process attached, and unlocks all locks the process locked.

If you are stopping certain system processes, you are prompted for confirmation before those processes are stopped, even if you specify `-no_ask`.

**Examples**

**Example 1.**

If you have more than one process and you issue the `stop_process` command using a star name for `process_name`, the system issues the following prompt.

```
Verify processes to be stopped.
Smith.Sales(login)? (yes,no,info)
```

If you type `yes` at the prompt, the process is stopped; if you type `no`, the process continues uninterrupted. If you specify `info`, the system displays information about the subprocess level, program name, PID (that is, the process identifier of the process), and login time of the process. If the process is interactive, the system returns the terminal name from which the process was started. The system does not return a terminal name if the process is not interactive or if the process is logged in remotely from a module that is not running a current version of the operating system.

The system then issues the prompt again.

```
Logged in at 90-02-19 07:33:26 EDT, sub-process level 0.
Running emacs.pm on %s1#t1.6, PID 011D88DDx.
Smith.Sales(login)? (yes,no,info)
```
If your process is at command level, the system returns the following information.

Logged in at 90-02-19 07:33:26 EDT, sub-process level 0.
Running on %s1#t1.6, PID 011D88DDx.
Smith.Sales(login)? (yes,no,info)

**Example 2.**

To stop the processes named `update_reports` and `update_accounts` that you started earlier on the current module, use this command.

```
stop_process update_*
```

**Example 3.**

The `stop_process` command can stop a process that has locked a file in order to unlock the file. Stopping a process releases all of the resources the process reserved, including locked files. Suppose a file, `receipts`, is locked but you do not know which process locked it. The `who_locked` command tells you. With that information, you can stop the process.

For example, the following command tells you the names of the processes that currently have `receipts` locked.

```
who_locked receipts
```

Suppose that only the process `update_receipts` has `receipts` locked, and that `update_receipts` is preventing you from using it. (Perhaps `update_receipts` is a long interactive session or is looping infinitely.) Issue this command.

```
stop_process update_receipts
```

The command stops the process `update_receipts` and consequently unlocks `receipts`.

**Related Information**

For information on starting a process, see the description of the `start_process` and `login` commands. For information on stopping a process and keeping a record for debugging, see the description of the `break_process` command. For information on stopping your login process, see the `logout` command. For information on sending signals to one or more processes, see the description of the `kill` command.
tail_file

Purpose

This command enables you to display the last part of a file or files.

Display Form

```
---------------------------------------tail_file -------------------------------
file_path_name:       [ ]
-records:         10
-match:            no
-follow:          yes
-check_lockers:   yes
```

Command Line Form

```
tail_file file_path_name
       [-records num_records]
       [-match character_string]
       [-follow]
       [-no_check_lockers]
```

Arguments

- **file_path_name** \[Required\]
The path name or star names of the file or files you want to display. The files may contain only ASCII text and standard control characters. They may not contain any word processing format information.

- **-records num_records**
  Begins displaying the file at the designated number of lines from the end of the file. If you omit this argument, the command displays the last ten lines of the file.

- **-match character_string**
  Displays only the lines in the files that contain the character string `character_string`. The command disregards the case of the alphabetical characters.

- **-follow** \[CYCLE\]
  If the file is implicitly locked by another program or other programs, causes the command to read from the end of the file until the file is unlocked or `BREAK` is signalled. If a star name matches more than one file, the `-follow` argument is ignored.

- **-no_check_lockers** \[CYCLE\]
  When used with the `-follow` argument, causes the command to display the file even when there are no other processes that have the file open. If you omit this argument, and
invoke `tail_file` with the `-follow` argument, the command stops displaying the file when no other processes have the file open or when `BREAK` is signalled.

**Explanation**

The `tail_file` command enables you to display the last part of an ASCII text file or files, depending upon the criteria you choose. If you give the command with only a file name or star name, and no arguments, the command displays the last 10 lines of the file or files.

The `-match` argument allows you to display only the lines containing the string `character_string`. If `character_string` contains spaces, you must enclose the string in apostrophes (`'`). This argument is convenient for displaying only the portions of a file that contain a particular string and for identifying all files that contain the string.

You can use the `-follow` argument to display output added to an implicitly locked file by a background process as the process executes. In order for the command to display this information, the file must be open for implicit locking by all processes accessing the file.

When invoked with the `-check_lockers` argument and the `-follow` argument, the `tail_file` command continues to display the file contents, including any output added by another process which has the file locked implicitly, until all processes have closed the file or `BREAK` is signalled.

You can use the `-no_check_lockers` argument to track a file even when no other processes have the file open. This is particularly useful for displaying system error log files which may receive error reports at irregular intervals, and therefore may not be open by a process at all times.

**Access Requirements**

You must have read access to the specified file or files in order to display them.

**Examples**

The following examples show several uses of this command.

**Example 1.**

Type this command to display the last ten lines of the file `reports`.

```
tail_file reports
```

**Example 2.**

Type this command to display the last 30 lines of the file `rpts.93` along with all other files which match the star name `rpts.*`

```
tail_file rpts.* -records 30
```
tail_file

Example 3.

Type this command to display the last 15 lines of the file rpts.93, followed by any lines that are appended to rpts.93 between the initiation and completion of the tail_file command.

    tail_file rpts.93 -records 15 -follow -no_check_lockers

Related Information

See also the description of the display_file command.
temacs

Purpose
This command provides an alternate version of the Emacs text editor for use within the bash shell. It creates stream files, uses the traditional key bindings, and accepts POSIX-style path names.

Display Form

<table>
<thead>
<tr>
<th>Option</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>file_names:</td>
<td></td>
</tr>
<tr>
<td>-start_up_path:</td>
<td>current_start_up_path_name</td>
</tr>
<tr>
<td>-num_windows:</td>
<td>1</td>
</tr>
<tr>
<td>-backup:</td>
<td>no</td>
</tr>
<tr>
<td>-keystrokes:</td>
<td>no</td>
</tr>
<tr>
<td>-keystrokes_dir:</td>
<td>current_directory</td>
</tr>
<tr>
<td>-flow_control:</td>
<td>no</td>
</tr>
<tr>
<td>-nls:</td>
<td>no</td>
</tr>
<tr>
<td>-dictionary:</td>
<td>current_dictionary_path_name</td>
</tr>
<tr>
<td>-organization:</td>
<td>stream</td>
</tr>
<tr>
<td>-record_size:</td>
<td></td>
</tr>
<tr>
<td>-character_set:</td>
<td>none</td>
</tr>
<tr>
<td>-shift_mode:</td>
<td>none</td>
</tr>
<tr>
<td>-pathname_style:</td>
<td>posix</td>
</tr>
<tr>
<td>-compatibility:</td>
<td>traditional</td>
</tr>
</tbody>
</table>

Command Line Form

`temacs [file_names...]`

[-start_up_path start_up_path_name]
[-num_windows number]
[-backup]
[-keystrokes]
[-keystrokes_dir keystrokes_path_name]
[-flow_control]
[-nls]
[-dictionary dictionary_path_name]
[-organization organization]
[-record_size record_size]
[-character_set character_set]
[-shift_mode shift_mode]
[-pathname_style style_name]
[-compatibility method_name]
Arguments

Note: Except for the -organization, -pathname_style, and -compatibility arguments, the arguments of the temacs command are identical to those shown in the description of the emacs command. See the description of the emacs command for more information about the arguments not shown here.

-organization organization
  Specifies one of the following types of file organization for the new file.
  - sequential
  - stream
  - relative
  - fixed

  By default, the temacs command creates a stream file. In contrast, the non-POSIX emacs command creates a sequential file by default.

-pathname_style style_name
  Determines whether Emacs interprets path names as POSIX-style (slash-separated or greater-than-separated) path names or as OpenVOS-style (greater-than-separated) path names. Possible values for style_name are posix or vos. By default, the temacs command interprets all path names as POSIX-style path names. In contrast, the non-POSIX emacs command interprets all path names as OpenVOS-style path names by default.

  This argument applies to all input path names, whether on the command line, given to prompts, or processed by the ESC<TAB> completion action. It also applies to the path names that are arguments to the -dictionary, -start_up_path, and -keystrokes_dir arguments. This argument has no effect on output path names; Emacs always displays OpenVOS-style path names.

-compatibility method_name
  Determines whether Emacs commands and mode settings are initialized to their “traditional” values (that is, GNU Emacs values) or to OpenVOS-specific values. Possible values are vos or traditional. By default, the temacs command initializes its commands and mode settings to traditional values. In contrast, the non-POSIX emacs command initializes its commands and mode settings to OpenVOS-specific values.

Description

The temacs command is a shell script that provides an alternate version of the Emacs text editor for use within the bash shell. When you are within the bash shell and specify the temacs -form command, the display form for the emacs command appears. However, the default values for the -organization, -pathname_style, and -compatibility arguments differ from those of the non-POSIX emacs command.

Access Requirements

You need read access to a file in order to read it into an Emacs buffer. To write the contents of an Emacs buffer to a file, you need modify access to the directory and write access to the
file (which can be specified in the default access list for the directory or the access list for the file).

**Examples**

In the following example, the `bash` command opens a bash shell. The `temacs` command then opens a file named `test` in Emacs.

```
ready  13:22:32
bash
bash-2.05$ temacs test
```

**Related Information**

See the *VOS Emacs User’s Guide* (R093) for a complete description of Emacs requests. See also the descriptions of the `emacs` and `vemacs` commands.
text_data_merge

Purpose

This command generates form letters by replacing identifiers in documents with character strings, using records read in from a value file.

Display Form

<table>
<thead>
<tr>
<th>Option</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>primary_file:</td>
<td></td>
</tr>
<tr>
<td>-value_file:</td>
<td></td>
</tr>
<tr>
<td>-data_definition_file:</td>
<td></td>
</tr>
<tr>
<td>-output_file:</td>
<td></td>
</tr>
<tr>
<td>-sample:</td>
<td>no</td>
</tr>
<tr>
<td>-notify:</td>
<td>no</td>
</tr>
</tbody>
</table>

Command Line Form

```
text_data_merge primary_file
   [-value_file value_file_name]
   [-data_definition_file data_definition_file_name]
   [-output_file output_file_name]
   [-sample]
   [-notify]
```

Arguments

- **primary_file** Required
  The name of an input form letter that contains text and instructions that the operating system uses when it generates output form letters through reformatting and text substitution. The primary file must be in one of four formats: edit, emacs, SOS-WP, or plain text.

- **-value_file value_file_name**
  Specifies the file whose records contain the values to be substituted for the identifiers in the primary file. The value file can be a mailing list, such as a purchased or internal list, or can be created with the create_table command or a text editor. In any case, you must describe the record structure in a data description file specified in the -data_definition_file argument. By default, the operating system looks for a file named primary_file.value in the current directory; if no such file exists, you receive an error message and return to command level.
text_data_merge

▶ -data_definition_file data_definition_file_name
   Specifies the file that describes the format of the data in the value file. By default, the operating system looks for a file named primary_file.dd in the current directory; if no such file exists, you receive an error message and return to command level.

▶ -output_file output_file_name
   Specifies the name of the file that contains the merged form letters after text_data_merge finishes generating them. By default, the command creates a file named primary_file.tdm_out in the current directory. In either case, the command opens the output file before attempting to open any input files. If it cannot open any input file, the command writes an error message to the output file.

▶ -sample (CYCLE)
   Performs a sample merge rather than a full merge. A sample merge merges only the first record in the value file with a copy of the primary file, so that you can verify the correctness of the merge before running a full merge. A full merge merges all records in the value file.

▶ -notify (CYCLE)
   Sends a message when all merges are completed.

Explanation
The text_data_merge command generates form letters by replacing identifiers within the text of a primary document with character strings from records read from a value file.

The command writes the results to an output file that contains one merged copy of the primary document for each record in the value file. The output form letters, separated by page breaks, are ready to print. The text_data_merge command can be called interactively or through the batch processing system.

The text_data_merge command uses four files (note that these files cannot have extended names):

1. a primary file containing the form letter and identifiers to be replaced
2. a value file containing the fields to replace the identifiers in the primary file
3. a data definition file defining the records in the value file
4. an output file containing the merged form letters after the program has executed.

You must provide the first three files before invoking text_data_merge. The fourth is created by the program. All four files are described in the following sections.

Primary Files
A primary file is a combination of the following:

- text
- text_data_merge functions
- embedded replacement references
The following example creates a primary file and names it form_letter.

```plaintext
&LET h1 = 'Acme Corporation'
&LET h2 = '100 Main Street'
&LET h3 = 'Boston, Ma. 02111'

&FORMAT OFF

&h1&
&h2&
&h3&

&Name&
&Street&
&City&

&FORMAT ON

Dear &Name&,

In reference to your account, number &AccountNo&, please note the status of your payment schedule.

&FILE 'document1'&

.
.
.

May we expect a prompt reply?

&FORMAT OFF

Sincerely,

Joyce Jones
Account Department

&FORMAT ON
```

The `text_data_merge` functions and embedded replacement references are described later in this section.

All `text_data_merge` functions and embedded replacement references are delimited by the `&` character, unless a different delimiter character is specified. (See the description of the `&SET_DELIMITER` function later in this command description.) Embedded references have the form `&identifier&`, with a delimiter on each side of the identifier name. These references are replaced by the corresponding value of that identifier from the current record in the value file. Functions have the form `&KEYWORD qualifiers`; they must begin a line, with the delimiter character in column one; any text on the line other than the function is disregarded.

The parsing of embedded references begins at the opening `&` and terminates at the next `&`, at which time `text_data_merge` attempts to make a replacement as described.

If the `text_data_merge` command finds a space within the identifier string, or if the end of the line is reached, the command views this potential identifier simply as text and includes it
in the form letter unchanged. It then continues to search for another text_data_merge code. This is not considered to be an error in the primary file.

**Embedded Replacement References**

Table 2-34 lists the replacement references that you can use to create primary files. These references are described in the text that follows.

**Table 2-34. Embedded Replacement References**

<table>
<thead>
<tr>
<th>Reference</th>
<th>Reference Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>&amp;identifier&amp;</td>
<td>Simple replacement reference</td>
</tr>
<tr>
<td>&amp;FILE 'string'&amp;</td>
<td>File reference</td>
</tr>
<tr>
<td>&amp;IF string1 = string2 THEN identifier ELSE identifier&amp;</td>
<td>Conditional replacement reference</td>
</tr>
<tr>
<td>&amp;LET identifier = 'string_constant'</td>
<td>Local variable</td>
</tr>
</tbody>
</table>

- **&identifier&**
  The identifier is the name of a field defined in the data definition file or the name of a local variable. There cannot be any spaces within identifier or between an ampersand (&) and identifier.

- **&FILE 'string'&**
  The #FILE reference reads the contents of the secondary file string in an output form letter. The value of string must be a string constant that names a file. It cannot be a local variable name or the name of an identifier from the current record in the value file. The secondary file is processed like the primary file, and all functions and references are interpreted as the text is included. Once the end of the file is reached, the text_data_merge command continues to process the primary file.

  **Note:** The #FILE reference can only appear within a primary file, not a secondary file. If the #FILE reference appears within a secondary file, text_data_merge aborts.

- **&IF string1=string2 THEN identifier ELSE identifier&**
  The strings string1 and string2 can be values of an identifier taken from the current record of the value file, the names of local variables, string constants of the form 'string_constant', or text_data_merge functions.

  The identifier in the THEN and ELSE clauses can be an identifier from the value file or the name of a local variable.

  If the ELSE clause is missing, the value of the second identifier is assumed to be the null character string.

- **&LET identifier = 'string_constant'**
  This function must start in column 1 and be the only text on the line. When the text_data_merge command processes the primary file, all occurrences of &identifier& are replaced with string_constant.
Text Data Merge (TDM) Functions and Keywords

- **&function&**
  
  A function can be one of the following keywords.

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DATE</td>
<td>The local date when the text_data_merge command began running</td>
</tr>
<tr>
<td>DAY</td>
<td>The local day of the week when the text_data_merge command began running</td>
</tr>
<tr>
<td>TIME</td>
<td>The local time when the text_data_merge command began running</td>
</tr>
</tbody>
</table>

The date, day, and time will be in this form.

January 1, 1990
Tuesday
10:00 am

- **&keyword value**
  
  These functions must begin in column 1 and be the only text on the line. The values for keyword and value are as follows:

  **&TDM ON/OFF**
  
  If you set TDM to OFF, the text_data_merge command does not attempt to find functions or references in any text that appears after the line on which the function appears. If you set TDM to ON, the command looks for functions and references in the text. The default value of TDM is ON.

  **&FORMAT ON/OFF**
  
  Unformatted text files can be formatted using the &FORMAT ON/OFF command. When you request formatting, the right margin is set to column 70. The first and second lines of each paragraph keep their current starting columns, and the remainder of the paragraph is indented to the same column as the second line. However, the left margin of every line in the output file must be at least 10 columns; if the left margin of any line is less than 10, it is set to exactly 10 columns.

  For plain, unformatted text, if you set FORMAT to OFF, the text_data_merge command does not format any text that appears after the line on which the function appears. The text appears precisely as entered in the primary file. The default value of FORMAT is ON.

  When the primary file is a formatted document created using the word processing editor, the reformatting of the output file is totally controlled by the format information specified in the primary file. In this case, it is unnecessary to use the #FORMAT control function.
&SET_DELIMITER to 'delimiter_char'

This function changes the text_data_merge delimiter character to the valid delimiter you specify. (The text_data_merge command will always recognize &SET_DELIMITER, even if you have set the delimiter to some character other than an ampersand (&).)

Default delimiter: &
Valid delimiter characters: & @ # $ % ^ * +
- | \ [ ] < > { }

Value File

You can create a value file using either the create_table command or a text editor. This file is usually named primary_file.value.

To create a value file using the create_table command, you must create a .tin (table input) file such as the following:

```
/ =Name        D. Clarke
  =Street      1 Shady Lane
  =City        Riverside, MA  01701
  =AccountNo   36225

/ =Name        J. Smith
  =Street      5 Pleasant Drive
  =City        Marlboro, MA  01752
  =AccountNo   22591
```

With this .tin file as input, the create_table command produces a table file that looks like this.

```
D. Clarke           1 Shady Lane             Riverside, MA  01701    36225
J. Smith            5 Pleasant Drive         Marlboro, MA  01752     22591
```

This table file can be a value file; for example, form_letter.value. Note that in the table file, all records must be the same length; if the length of the information contained in a field is less than the length of the field as defined in the data definition file, the field is padded on the right with spaces. (A description of data definition files appears in the “Data Definition File” section.)

See the OpenVOS System Administration: Configuring a System (R287) for a description of the create_table command and for information about .tin files and table files.

To create a value file with an editor, first create the file by invoking the create_file command. Set -organization to fixed and -record_size to the total length of the fields you want to enter. Then, using an editor, type records into the file, being sure to blank-fill any fields that are shorter than the maximum length specified in the file.
Data Definition File
The data definition file describes the format of the value file to enable the
`text_data_merge` command to process the records. The data definition file, usually a file
with the suffix `.dd`, must be in a form like this.

```plaintext
organization: fixed;

field: Name char (20);
field: Street char (25);
field: City char (25);
field: AccountNo char (5);
end;
```

The field specifications in this file, `form_letter.dd`, match the format of the data in the file
`form_letter.value`.

Output File
When you invoke the `text_data_merge` command interactively, the command writes all
error messages both to your terminal and to an error file with the name
`output_file.error`.

If no errors occur while the `text_data_merge` command processes, no
`text_data_merge` codes appear in the output file, since these codes are all interpreted by
the command. However, if the program cannot interpret any codes in the primary file, the
program stops after the first merge completes, producing only one copy of the form letter. If
this occurs, all codes that `text_data_merge` does not recognize are left in the output file
text.

After the merge finishes, look at the output file to verify that the resulting form letter is
correct, and that all codes were interpreted as expected.

Examples
Assume that the current directory contains the following files.

- `form_letter`, the sample primary file
- `form_letter.value`, the sample value file
- `form_letter.dd`, the sample data definition file
- `document1`, a text file referenced by `&FILE` that begins with 'The terms of our
agreement...'

The following command will produce one output form letter in the output file
`form_letter.tdm_output`.

```plaintext
text_data_merge form_letter -sample
```
The contents of the file will look like this.

D. Clarke
1 Shady Lane
Riverside, MA  01701

Dear D. Clarke,

In reference to your account, number 36225, please note the status of your payment schedule.

'The terms of our agreement...' 

May we expect a prompt reply?

Sincerely,
Joyce Jones
Account Department

'\0C

The '\0C characters that appear between documents in such an output file are a page-break instruction for a printer.

The command that generates the previous form letter also produces an empty error file named form_letter.error.
translate_links

Purpose
This command changes a system name, disk name, or directory name in link targets throughout a directory hierarchy. This command is useful when you move a disk from one system to another or when you change the name of a disk or a system.

Display Form

```
-ROOT_DIR: current_dir
-old_system: old_system_name
-new_system: new_system_name
-old_disk: old_disk_name
-new_disk: new_disk_name
-old_dir: old_dir_path_name
-new_dir: new_dir_path_name
-log: no
```

Command Line Form

```
translate_links [-ROOT_DIR root_dir_path_name]

[ -old_system old_system_name ]
[ -new_system new_system_name ]
[ -old_disk old_disk_name ]
[ -new_disk new_disk_name ]
[ -old_dir old_dir_path_name ]
[ -new_dir new_dir_path_name ]
[ -log ]
```

Arguments

- `-ROOT_DIR root_dir_path_name`
  Specifies the directory that is the root of the directory hierarchy throughout which the operating system is to change the disk or system name. The default value is the current directory.

- `-old_system old_system_name`
  Specifies the system name to be changed. This argument is required if `-new_system` is specified.
-new_system new_system_name
   Specifies the new system name. This argument is required if -old_system is specified.

-old_disk old_disk_name
   Specifies the disk name to be changed. This argument is required if -new_disk is specified.

-new_disk new_disk_name
   Specifies the new disk name. This argument is required if -old_disk is specified.

-old_dir old_dir_path_name
   Specifies the path name to be changed. This argument is required if -new_dir is specified. If -old_dir is specified, then -old_system and -old_disk must be omitted.

-new_dir new_dir_path_name
   Specifies the new path name. This argument is required if -old_dir is specified.

-log <CYCLE>
   Displays the names of the old and new targets of each link as the names are changed. By default, translate_links does not display the names.

Explanation

The translate_links command changes a system name, a disk name, and a directory name throughout the specified directory hierarchy.

The name of the command reflects the fact that disk names, system names, and directory names are stored in links throughout a directory hierarchy.

If you specify both -old_system and -old_disk, a link is translated only if its target is on %old_system_name# old_disk_name.

If you specify -old_disk but omit -old_system, a link is translated only if its target is on %current_system# old_disk_name.

This command checks for and deletes circular links. (A circular link is a link that points to itself.)

Access Requirements

You need modify access to a directory in order to translate its links.
translate_links

Examples
The following command changes the name of the disk disk_2 to the new name, d02, throughout the current directory hierarchy.

    translate_links -old_disk disk_2 -new_disk d02

Related Information
The translate_links command is one of the steps in the procedure for changing a disk name or a system name. For more information, see the OpenVOS System Administration: Configuring a System (R287).
**truncate_file**

**Purpose**

This command truncates a specified file and all of its indexes.

**Display Form**

```
file_name: ...
-retain: no
-brief: no
```

**Command Line Form**

```
truncate_file file_name...

[-retain]
[-brief]
```

**Arguments**

- `file_name`  
  Required  
  One or more names or star names of files to be truncated.

- `-retain`  
  CYCLE  
  Retains the allocated disk space for the files when they are truncated. You cannot specify this argument for statically-allocated extents (SAE) files. By default, the operating system frees the disk space.

- `-brief`  
  CYCLE  
  Suppresses the display of each file name that matches a star name before the file is truncated. By default, `truncate_file` displays the name(s).

**Explanation**

The `truncate_file` command truncates the specified file to zero length and truncates all of the file’s indexes. The command deletes all the records of the file, but neither the file nor any of its indexes is deleted.

**Note:** You cannot set RAM file usage on SAE files, because truncating an SAE file reverts it to a normal file. Similarly, truncating a RAM file with an SAE index reverts the index to a normal index without extents. See `set_ram_file` for more information about RAM files.
**truncate_file**

**Access Requirements**
You must have write access to a file in order to truncate it.

**Examples**
The following command truncates the file `receipts` in the current directory.

```
truncate_file receipts -retain
```

The command also retains the disk blocks already allocated to the file. When records are written to the file, the retained disk space will be reused before new disk blocks are allocated.

**Related Information**
See also the command descriptions of `compare_files`, `copy_file`, `create_file`, `display_file_status`, `dump_file`, `locate_files`, and `move_file`.
unbundle

Purpose
This command extracts files that were combined with the bundle command.

Display Form

```
unbundle

source_file: Required
destination_dir:
-extract_only:
-overwrite: yes
-backup: no
-keep_dates: no
-restore_acls: no
-list_only: no
-brief: no
```

Command Line Form

```
unbundle source_file
[destination_dir]
[-extract_only star_name...]
[-no_overwrite]
[-backup]
[-keep_dates]
[-restore_acls]
[-list_only]
[-brief]
```

Arguments

- **source_file**  
  Either the path name of a file that was created with the bundle command, or the path name of a file, received as email, that contains a uuencoded or MIME (base64) encoded attachment.

  If the file was created with the bundle command, you must specify all of the suffixes, just as they appear in the file's name. You cannot specify a star name.

  If the file was received as email, you can name it was any suffix or with no suffix.

  Table 2-35 shows the suffixes that the bundle command appends to destination files.
Some file-transfer programs (for example, Microsoft® Internet Explorer®) may translate all but the last period to underscores. Therefore, a name such as `sendfile.save.evf.gz` may be translated to `sendfile_save_evf.gz`. The `unbundle` command accepts either version of the name.

- **destination_dir**
  The path name of the directory where you want the extracted files restored. If the directory contains files of the same name as one of the extracted files, the old file is silently overwritten unless you specify `-no_overwrite`.

- **-extract_only star_name**
  Specify this argument only when you do not want to extract all of the files in the bundle source. Enter a list of star names that you want to extract from the bundle source. They must be object names, but you cannot specify a directory. If you specify more than one `star_name` on the command line, separate the names with spaces and enclose the whole list in quotes. (You do not need quotes when you enter the names into the display form.) For example:

  ```
  unbundle sendfile mydir -extract_only '*.doc *.help'
  ```

- **-no_overwrite**
  Specifies that the command does not silently overwrite any files. By default (the value `yes`), the command silently overwrites existing files that have the same names as those being extracted from the bundle.

Note that this argument applies to final extracted files only. The `unbundle` command always overwrites intermediate scratch files, which are created and deleted during the unbundling process, even if you specify `-no_overwrite.`

### Table 2-35. Suffixes Appended by the `bundle` Command

<table>
<thead>
<tr>
<th>Source File Suffix</th>
<th>Description</th>
<th><code>bundle</code> Command Arguments Specified</th>
</tr>
</thead>
<tbody>
<tr>
<td>.bz2</td>
<td>Binary stream</td>
<td><code>for_ftp -no_short_suffix -compress_using bzip2</code></td>
</tr>
<tr>
<td>.gz</td>
<td>Binary stream</td>
<td><code>for_ftp -no_short_suffix</code></td>
</tr>
<tr>
<td>.ftp</td>
<td>Binary stream</td>
<td><code>for_ftp -short_suffix</code></td>
</tr>
<tr>
<td>.evf</td>
<td>Binary stream</td>
<td><code>for_ftp -compress_using none</code></td>
</tr>
<tr>
<td>.rsn</td>
<td>Binary sequential</td>
<td><code>for_rsn</code></td>
</tr>
<tr>
<td>.uue</td>
<td>ASCII sequential</td>
<td><code>uuencode</code></td>
</tr>
<tr>
<td>.uu</td>
<td>ASCII stream</td>
<td><code>uuencode_stream</code></td>
</tr>
<tr>
<td>.b64</td>
<td>Alphanumeric sequential</td>
<td><code>base64</code></td>
</tr>
<tr>
<td>.b</td>
<td>Alphanumeric stream</td>
<td><code>base64_stream</code></td>
</tr>
</tbody>
</table>
unbundle

- **-backup** <CYCLE> (Privileged)
  Specifies whether to restore the original values of the date-time-used (DTU), date-time-modified (DTM), and date-time-saved (DTS) attributes of the files and directories in the bundled file. If it is set, the DTS value is set to the current time. By default (no), the value of the DTS attribute is 0, and the value of the DTM attribute is the time that the object was restored. Specify yes to restore the original values of these attributes.

  You can use this privileged argument to get temporary access to any object to overwrite it, and to restore the saved ACL of each object. For more information, see the description of the restore command in OpenVOS System Administration: Backing Up and Restoring Data (R285).

- **-keep_dates** <CYCLE>
  Specifies whether to restore the original values of the DTU, DTM, and DTS attributes of the files and directories in the bundled file. If it is set, the DTS value is set to the current time. By default (no), the value of the DTS attribute is 0, and the value of the DTM attribute is the time that the object was restored. Specify yes to restore the original values of these attributes.

- **-restore_acls** <CYCLE>
  Specifies whether to restore access control lists (ACLs) and default access control lists (DACLs). By default (no), ACLs are not restored.

- **-list_only** <CYCLE>
  Lists the path names of the files in the bundled file on the screen. By default (no), the files are extracted from the bundled file and placed into the destination directory, but path names are not listed on the screen. If you specify yes, you can also specify the -brief argument to suppress display of the intermediate file extraction steps and show only the file names.

- **-brief** <CYCLE>
  Announces each step in the unbundling process on the screen as it occurs. By default (no), all output to the screen is suppressed except for error messages.

**Explanation**

The unbundle command extracts files that were combined with the bundle command.

If the bundling operation resulted in a destination file larger than 2 GB, the bundle command generated multiple compressed save files as well as a .toc file. If this .toc file is not located with the save files, the unbundle command fails.

If you specify the -backup argument with the -no_restore_acls argument and/or the -no_keep_dates argument, unbundle restores ACLs and dates (that is, it ignores -no_restore_acls and -no_keep_dates).

**Related Information**

See the description of the bundle command.
unlink

Purpose
This command deletes one or more links.

Display Form

Command Line Form

unlink link_names...
   [-no_ask]
   [-brief]

Arguments

- link_names  Required
  One or more names or star names of links to be unlinked.

- no_ask  CYCLE
  Suppresses the prompt that asks you whether to unlink a link with a matching name,
  when a link_names term is a star name. By default, the command asks you before
  unlinking a link with a matching name.

- brief  CYCLE
  Suppresses the display of the name of each link that matches a star name before it
  unlinks the link.

Explanation

The unlink command deletes one or more links specified by link_names.

Unless you specify -no_ask, when you give a star name in the command, the unlink
command asks whether you want to unlink each link whose name matches the star name. You
can decide which links with matching names to unlink.

When you are answering a series of questions about a set of links whose names match a
link_names term, the unlink command does not unlink any link in the set until it has asked
you about all of them. If you abort the command before answering all the questions about the
set, none of the links in the set is unlinked. After you have answered the command’s questions
about all the names that match one `link_names` term, however, the `unlink` command unlinks the links in that set before asking you questions about the next set of names.

Specifying `-brief` suppresses messages telling you which links are being unlinked.

**Access Requirements**

You need modify access to the directory containing a link to unlink it.

**Examples**

To unlink all links in the current directory whose names begin with `make_reports`, use this command.

```
unlink make_reports.*
```

**Related Information**

See the description of the `link` command for information about how to create a link.
update_batch_requests

Purpose
This command modifies a pending batch request.

Display Form

```
+----------------------------- update_batch_requests -----------------------------+
| process_names:             |
| -user:                     |
|   -ask:                    |
|     yes                    |
| -command_line:             |
| -output_path:              |
| -process_name:             |
| -process_priority:         |
| -queue_priority:           |
| -privileged:               |
| -restart:                  |
| -queue:                    |
|     normal                 |
| -module:                   |
| -defer_until:              |
| -control:                  |
| -after:                    |
| -cpu_limit:                |
| -notify:                   |
|     no                     |
```

Command Line Form

```
update_batch_requests process_names ...
    [-user user_name]
    [-no_ask]
    [-command_line command_line]
    [-output_path output_path_name]
    [-process_name process_name]
    [-process_priority process_priority]
    [-queue_priority queue_priority]
    [-privileged]
    [-restart]
    [-queue queue_name]
    [-module module_name]
    [-defer_until date_time]
    [-control control_file_name]
    [-after process_names]
    [-cpu_limit cpu_time]
    [-notify]
```
Arguments

- **process_names**
  One or more names or star names of batch processes to be updated. Unless you specify `-no_ask`, when you give a process name that is a star name, the command asks you before updating any process with a matching name.

- **-user user_name**
  Specifies one or more user names or star names. This updates the batch requests matching `process_names` that the specified users submitted. By default, `update_batch_requests` uses your user name and allows you to update only your own batch requests.

- **-no_ask**
  Suppressed the prompt, when a `process_names` term is a star name, asking whether to update a batch process with a matching name. By default, the operating system asks before updating a process.

- **-command_line command_line**
  Modifies the command line of a batch request.

- **-output_path output_path_name**
  Changes the output path of the batch request.

- **-process_name process_name**
  Renames the batch request.

- **-process_priority process_priority**
  Resets the execution priority of the process to any level, from the minimum priority to the existing maximum priority.

- **-queue_priority queue_priority**
  Resets the priority of the process within the queue. The queue priority can be any level between 0 and 9, with 9 being the highest level.

- **-privileged**
  Makes the batch process a privileged process. A process must be privileged in order to request a privileged batch process. To update a deferred job to a privileged process, the original submitter of the deferred job must have been privileged.

- **-restart**
  Restarts the batch request if processing is interrupted, for example, by a system shutdown.

- **-queue queue_name**
  Specifies the queue to be searched for batch requests. By default, the `batch` command puts your batch request in the normal queue, either on the module specified by the `-module` argument or on the current module.

- **-module module_name**
  Identifies the module on which the queue is located.
update_batch_requests

- **-defer_until date_time**
  Changes the time at which the batch request is to be executed. The `date_time` value can be a character string in the standard form.

  
  \[ yy-mm-dd_hh:mm:ss \]

  It can also be a character string in any form accepted by the `(date_time)` command function. In this case, the string must be enclosed in apostrophes.

- **-control control_file_name**
  Changes the control file for the batch process.

- **-after process_names**
  Specifies the process name of a request currently in the queue. The newly issued batch request is not executed until the process identified by `process_names` has been executed. By default, the operating system executes the batch request after other requests with a higher priority in the queue have been executed. You can supply multiple names with this argument.

- **-cpu_limit cpu_time**
  Changes the upper bound on the amount of CPU time, in seconds, that the batch process can consume before it is stopped.

- **-notify**
  Displays a message on the status line of your terminal when the process finishes.

**Explanation**

The `update_batch_requests` command modifies the batch request in the batch queue. Using the `update_batch_requests` command, you can change the command line, the order of requests in the queue, or the time to which a job is deferred.

You can use this command only until the batch process starts to run. Once the process starts running, you can no longer update its parameters using `update_batch_requests`.

If you want to use the `update_batch_requests` command to hurry the batch request through the queue, use `-queue_priority`.

You must specify which batch requests to update by their process names.

**Access Requirements**

You must have write access to the queue file of the batch queue in order to update batch requests submitted by other users. In this case, you can also specify batch requests by user name.

**Related Information**

See the OpenVOS Commands User’s Guide (R089) for information about batch control files. To display the batch processes you have submitted, use the `list_batch_requests` command. The batch processor gives the batch request a queue sequence number, which the `list_batch_requests` command displays. See also the command descriptions of `batch`, `cancel_batch_requests`, `display_batch_status`, `reserve_device`. 

2-790
move_device_reservation, and cancel_device_reservation. See Chapter 1, “OpenVOS Command Functions,” for examples of acceptable date/time input strings.
update_print_requests

Purpose
This command modifies a pending print request.

Display Form

```
update_print_requests

file_names:  
-user: 
-queue: standard
-module: 
-title: 
-destination: 
-device: 
-header: 
-footer: 
-index: 
-exception_handling: ''
-interpret_tabs: 
-defer_until: 
-ask: yes 
-indentation: 
-top_margin: 
-line_length: 
-delete: 
-page_breaks: 
-wrap: 
-notify: 
-pass_thru: 
-first_page: 

-copies: 
-page_size: 
-bottom_margin: 
-line_numbers: 
-raw: 
-use_fortran_controls: 
-queue_priority: 
-pass_thru: 
-last_page: 
```
Command Line Form

```
update_print_requests file_names...
  -user user_name
  -queue queue_name
  -module module_name
  -title file_name
  -destination destination_address
  -device device_name
  -header header_string
  -footer footer_string
  -index index_name
  -exception_handling string
  -interpret_tabs start_column, spacing
  -defer_until date_time
  -no_ask
  -copies number
  -indentation [indentation]
  -page_size
  -top_margin [top_margin]
  -bottom_margin [bottom_margin]
  -line_length line_length
  -line_numbers
  -delete
  -no_raw
  -no_page_breaks
  -use_fortran_controls
  -wrap
  -queue_priority [queue_priority]
  -notify
  -pass_thru
  -first_page first_page
  -last_page [last_page]
```

Arguments

- **file_names**  
  Required  
  One or more explicit names or star names of print processes to be updated. Unless you specify `-no_ask`, when you give a file name that is a star name, the command asks you before updating any print request with a matching name.

- **-user user_name**  
  Specifies the name and group of the user originally submitting the print request. This name can be a star name. By default, the command uses your user name and allows you to update only your own print requests.

- **-queue queue_name**  
  Specifies the print queue that contains all the specified requests that are being processed. If you want to update print requests in more than one print queue, you must invoke `update_print_requests` for each print queue you want to specify.
-module module_name
   Specifies the module that contains the print queue in which requests are located.

-title file_name
   Specifies the name of the file queued for printing. By default, the command selects the
   next file owned by you in the queue.

-destination destination_address
   Specifies the destination of the hardcopy after the job is printed. By default, the
   command prints with the original destination for your user ID.

-device device_name
   Specifies the name of the printer to which the job is queued. By default, the command
   leaves the print request queued with the original printer setting.

-header header_string
   Specifies the information you want printed at the top of hardcopy output. By default,
   the command leaves the original header at the top of the hardcopy output.

-footer footer_string
   Specifies the information you want printed at the end of the hardcopy output. By
   default, the command leaves the original footer at the end of the hardcopy output.

-index index_name
   Specifies an index that controls the order in which the records in a specified file are
   printed. By default, the command leaves the original index unchanged.

-exception_handling string
   Specifies how to handle nonprinting characters in the text. The possible values are
   replace, ignore, abort and ' '. If you specify replace, the operating system
   prints the hexadecimal number representing the ASCII code for the nonprinting
   character. If you specify ignore, the operating system ignores nonprinting characters.
   If you specify abort, the operating system cancels this print request if a nonprinting
   character is encountered, and continues with the next request in the queue. If you
   specify '' or omit this argument, update_print_requests replaces the original
   setting. You cannot specify ignore and -raw in the same command.

-interpret_tabs start_column,spacing
   Interprets occurrences of the ASCII tab character. You must specify the column number
   start_column of the first tab stop and the number spacing of positions between tab
   stops. A comma must separate the two numbers. You cannot specify
   -interpret_tabs and -raw in the same command. By default, the command prints
   the request with its original setting.

-defer_until date_time
   Changes the time at which the print request is to be executed. The date_time value
   must be a character string in the standard form. By default, the command executes the
   print request at the original date and/or time.
-no_ask
Suppresses the prompt, when a file_names term is a star name, asking whether to update a print request with a matching name. By default, the operating system asks before updating a print request.

copies number
Prints multiple copies of each specified file. By default, the operating system prints the original number of copies specified.

-indentation indentation
Sets the left margin to the column designated by indentation. By default, the operating system sets the left margin to the original position on the line.

-page_size page_size
Sets the number of lines on a page. After printing page_size lines, including top and bottom margin lines, the printer skips to a new page. By default, the command uses the original page size.

top_margin top_margin
Sets the number of lines in the top margin of each printed page. By default, the value of top_margin is 3. The first line of the file that appears on each page is printed on the first line after the top margin (line 4). If you print a file that incorporates formatting features of the edit command, or if you specify -no_page_breaks, this argument is ignored.

-bottom_margin bottom_margin
Sets the number of lines in the bottom margin of each printed page. By default, the value of bottom_margin is 3. The last line of the file that appears on each page is printed on the line immediately before the bottom margin. If you print a file that incorporates formatting features of the edit command, or if you specify -no_page_breaks, this argument is ignored.

-line_length line_length
Specifies the number of character positions per line. The line length includes any indentation. By default, the command sets the line length to the original requested value.

-line_numbers
Prints the file with line numbers. By default, the command prints the request with the original value.

-delete
Deletes the file after it is printed. By default, the command does not delete the file after it is printed.

-raw
Prints the file literally; all character sequences that are normally control sequences for the printer (and not printed) are replaced with the ASCII digits representing the hexadecimal value of the bytes. You cannot specify -raw and either -exception_handling ignore or -interpret_tabs in the same command. By default, the command prints the file without any control sequences displayed.
**update_print_requests**

- **-no_page_breaks**
  Prints the file without page breaks and automatically sets the top and bottom margins to 0. By default, the command prints the file with its original value.

- **-use_fortran_controls**
  Interprets any of the following characters as a FORTRAN printing control character when it appears in column 1 of a file. The command treats all other characters as space characters.

<table>
<thead>
<tr>
<th>Character</th>
<th>Printing Instruction</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Skip to the next page</td>
</tr>
<tr>
<td>0</td>
<td>Double space</td>
</tr>
<tr>
<td>+</td>
<td>Overstrike</td>
</tr>
<tr>
<td>space character</td>
<td>Skip to the next line</td>
</tr>
</tbody>
</table>

The command disregards any generic/canonical control sequences it encounters in the first column. (Generic/canonical control sequences are those that a user has entered in the body of text of a file.)

- **-wrap**
  Continues the printing of overflow from a long line on a subsequent line or lines. *Overflow* consists of the character or characters at the end of a line that will not fit into the positions available within the specified margins. By default, the command truncates long lines.

- **-queue_priority queue_priority**
  Sets the print request’s priority in the print queue. The value of *queue_priority* can be from 0 to 9, with 9 representing the highest queue priority. If you assign a queue priority to a print request, update_print_requests inserts the request in the queue before all requests with lower queue priority. By default, the command, the queue priority of a print request with a size less than or equal to 20 disk blocks is 5, and that of a print request with a size greater than 20 disk blocks is 4.

- **-notify**
  Tells the print command to send you a message when the printing of your job is complete.

- **-pass_thru**
  Passes any control codes embedded in the file through to the printer. You can use -pass_thru with the -device argument to ensure that the file is printed on the device for which it was generated.

- **-first_page first_page**
  Specifies that the request should start printing at the given page. All pages prior to the specified page are ignored.

- **-last_page last_page**
  Specifies the last page of the file to be printed. All pages after the specified page are ignored.
Explanation

The `update_print_requests` command modifies your print requests that exist in the print queue. Using the `update_print_requests` command, you can alter many characteristics of the hardcopy output.

You can use this command only until the job begins printing. Once the print job starts running, you can no longer update its parameters using `update_print_requests`.

To alter queued print requests, you must specify which print jobs to update by giving their file names, your user name, the print queue, and the module name on which your print requests are to be processed. In addition, you can specify other arguments to alter print output in numerous ways. When you give an argument to this command, the field is updated to the new value you specify. If you do not specify a new value for an argument, the operating system uses the original value specified.

Example

To change various arguments for your print requests, you could list the print requests in the queue, and then alter whatever arguments you desire. For example, the following long listing of a print job shows the various attributes of the job in the print queue.

```
Request:        01
User:           A_West.Dev
Destination:    A1
Time:           93-01-25 14:48:12 EST
Path Name:      %d#m26_user>Dev>Adam_West>progs>forms

Queue Priority: 5

Attributes: waiting
Title:  forms.c
Header: none.
Footer: none.

Copies:          01 File size:  01 Sort index:   none.
Line length:    00 Page size:  00 Indent:       01
Line numbers:   no Page Breaks:    yes Delete:   yes
Edited:          no Wrap:         yes Controls: canonical
Notify:         yes ready  14:50:20

If you wanted to change certain parameters, either after listing the queued print request, or remembering what attributes of the output you wished to change, you could, for example, change the destination, the indentation, and the number of copies printed. To change those attributes, you would issue the following command.

```
update_print_requests forms -destination b1 -indentation 05 -copies 3
```

Related Information

To display the print processes you have submitted, use the `list_print_requests` command. The print processor gives the print request a queue sequence number, which the
update_print_requests

list_print_requests command displays. See also the command descriptions of print, cancel_print_requests, and display_print_status.
update_process_cmd_limits

Purpose

This command changes the command limits for an existing process or set of processes.

Display Form

```
update_process_cmd_limits

process_name: <process_name>
-user: current_user
-module: current_module
-initial_total_limit:
-initial_heap_limit:
-initial_stack_limit:
-initial_cpu_limit:
-initial_file_limit:
-initial_keep_limit:
-initial_port_limit:
-maximum_total_limit:
-maximum_heap_limit:
-maximum_stack_limit:
-maximum_cpu_limit:
-maximum_file_limit:
-maximum_keep_limit:
-maximum_port_limit:
-ask: yes
```
**update_process_cmd_limits**

**Command Line Form**

```
update_process_cmd_limits [process_name]
[-user user_name]
[-module module_name]
[-initial_total_limit number]
[-initial_heap_limit number]
[-initial_stack_limit number]
[-initial_cpu_limit number]
[-initial_file_limit number]
[-initial_keep_limit number]
[-initial_port_limit number]
[-maximum_total_limit number]
[-maximum_heap_limit number]
[-maximum_stack_limit number]
[-maximum_cpu_limit number]
[-maximum_file_limit number]
[-maximum_keep_limit number]
[-maximum_port_limit number]
[-no_ask]
```

**Arguments**

- **process_name**
  
  The name or star name of existing processes for which you want to change the process command limits. By default, if the value of `-user` is the `current_user` and the value of `-module` is `current_module`, then the command sets the process limits of the current process. Otherwise, the default process name is an asterisk (*), indicating all processes of the specified user on the specified module.

- **-user user_name**

  Specifies a name or user star name indicating one or more users whose processes receive the specified command limits. By default, the command uses your user name. Your process must be privileged to set any limits for another user’s process or to increase any of the maximum limits for your own processes. If your process is not privileged, you can decrease any of the maximum limits or change any of the initial limits for your own processes.

- **-module module_name**

  Specifies the name or star name of modules on which the processes are running. By default, the command uses the current module.

- **-initial_total_limit number**

  Specifies, in bytes, the initial value for a command’s current total virtual-space limit when started in the selected process(es), if the command does not explicitly specify a total virtual-space limit itself.
- **-initial_heap_limit number**
  Specifies, in bytes, the initial value for a command’s current heap-space limit when started in the selected process(es), if the command does not explicitly specify a heap-space limit itself.

- **-initial_stack_limit number**
  Specifies, in bytes, the initial allocation for stack space for a command when started in the selected process(es), if the command does not explicitly specify a stack-allocation size itself.

- **-initial_cpu_limit number**
  Specifies, in seconds, the initial value for a command’s CPU time limit when started in the selected process(es).

- **-initial_file_limit number**
  Specifies, in bytes, the initial value for a command’s stream file size limit when started in the selected process(es).

- **-initial_keep_limit number**
  Specifies, in bytes, the initial value for a command’s keep module size limit when started in the selected process(es).

- **-initial_port_limit number**
  Specifies the initial value for a command’s number of attached ports limit when started in the selected process(es). The number is an integer.

- **-maximum_total_limit number**
  Specifies, in bytes, the upper bound on total virtual-space usage for commands run in the selected process(es). You must be privileged or the root process to increase any process’s maximum total virtual-space limit.

- **-maximum_heap_limit number**
  Specifies, in bytes, the upper bound on heap growth for commands run in the selected process(es). You must be privileged or the root process to increase any process’s maximum heap limit.

- **-maximum_stack_limit number**
  Specifies, in bytes, the upper limit on stack allocations for commands run in the selected process(es). You must be privileged or the root process to increase any process’s maximum stack allocation.

- **-maximum_cpu_limit number**
  Specifies, in seconds, the upper limit on CPU time for commands run in the selected process(es). You must be privileged or the root process to increase any process’s maximum CPU time.

- **-maximum_file_limit number**
  Specifies, in bytes, the upper limit on stream file size for commands run in the selected process(es). You must be privileged or the root process to increase any process’s maximum stream file size.
update_process_cmd_limits

- **maximum_keep_limit number**
  Specifies, in bytes, the upper limit on keep module size for commands run in the
  selected process(es). You must be privileged or the root process to increase any
  process’s maximum keep module size.

- **maximum_port_limit number**
  Specifies the upper limit on the number of attached ports for commands run in the
  selected process(es). The number is an integer. You must be privileged or the root
  process to increase any process’s maximum number of attached ports.

- **no_ask**
  Specifies that the operating system not ask when you specify a star name for
  process_name. The default value is yes. If you do not specify a star name for
  process_name, the command ignores this argument.

**Explanation**

The update_process_cmd_limits command controls the two sets of command limits
associated with a process. The purpose of command limits is to control the amount of a
resource that an executing program can use. A system administrator can adjust the limits to
prevent runaway programs from using excessive resources or simply to enforce a degree of
fairness on the use of system resources.

The two sets of command limits are each composed of the following resource limits:

- The **heap limit** controls the growth of the user heap, which you can access with facilities
  such as the PL/I allocate statement, the C malloc routine, or the $allocate subroutine.

- The **stack limit** controls the space allocated for the user stack, which supplies PL/I
  automatic storage or C auto storage. Note that the stack fence moves when the stack
  limit is changed; thus, the stack limit is actually a stack allocation.

- The **total space limit** controls the total quantity of virtual-address space that a command
  uses. The total space comprises the heap space, the stack space, the space defined by
  the program module, and any $connect_vm_region3 space that does not overlay
  any of the three other types of space.

- The **CPU time limit** controls the maximum number of CPU seconds that a process can
  consume before it is stopped.

- The **stream file size limit** controls the maximum size of a stream file that the process
  can create.

- The **keep module size limit** controls whether or not the system should attempt to create
  a keep module.

- The **number of attached ports limit** controls the number of ports that can be attached by
  the process.

The first set of limits, called the **maximum** set of limits, defines the highest size that any
command run in the process can use for any of these resources.
The second set of limits, called the initial set of limits, is defined by `update_process_cmd_limits`. These define the limits on a command when it first starts, only when the program module for the command does not contain specifications on the limits it requires. These limit values, either from the program module or from the initial set, initialize each command’s current limits. These current limits are checked whenever a command wants to change any of its limits. The command can adjust the current limits up or down as it runs by using the `s$set_current_cmd_limit` subroutine, which is described in the OpenVOS Subroutines manuals. The current limits cannot be set greater than their respective maximum limits.

**Note:** The value 2,147,483,647 represents infinity on modules running releases prior to OpenVOS Release 17.2.x, while the value 9,223,372,036,854,775,807 represents infinity on modules running OpenVOS Release 17.2.x and later. If you invoke `update_process_cmd_limits` on a module running OpenVOS Release 17.2.x or later and attempt to set process limits on a module running OpenVOS Release 17.1.x or earlier, the command returns the error `e$out_of_range` (1038) for arguments whose values fall between the old and current infinity values.

**Explanation of the Arguments**

This section provides detailed information about some of this command’s arguments.

The `update_process_cmd_limits` command provides new values for the maximum and initial values for resource limits to the processes specified by `process_name`, belonging to the user `user_name`, on the module `module_name`.

If your process is privileged or the root process, you can set these limits on any user process, including your own. If your process is not privileged or the root process, you can set these limits only for your own process, and you cannot increase any of the maximum limits, even for your own process.

**Note:** If you specify an infinite value (that is, the value infinity) for any of the numeric arguments, the `update_process_cmd_limits` command no longer enforces an administrative limit for that argument. Specifying the actual value 9,223,372,036,854,775,807 does not necessarily mean an infinite value for modules running OpenVOS Release 17.2.x or later.

The `-initial_total_limit` argument specifies, in bytes, the initial current limit on the total space of any command started in the specified process if you did not specify the maximum program size during binding of the program module. The total space is equal to the size to which the program module space, the user heap, the user stack, and the (non-overlaying, independent) `s$connect_vm_region3` areas can grow. If you did specify the maximum program size when binding the program module, the command uses the bind-time value. You can specify infinity or a value between 131,072 and 2,145,779,712 bytes, inclusive, for this argument. However, this value cannot be greater than the value of the `-maximum_total_limit` argument.

The `-initial_heap_limit` argument specifies, in bytes, the initial limit for the heap of any command started in the specified process if you did not specify the maximum heap size when binding the program module. If you did specify the maximum heap size when binding the program module, the command uses the bind-time value. You can specify infinity or
a value between 32,768 and 2,145,746,944 bytes, inclusive, for this argument. However, this value cannot be greater than the value of the -maximum_heap_limit argument.

The -initial_stack_limit argument specifies, in bytes, the initial allocation for the stack of any command started in the specified process if you did not specify the stack size when binding the object modules that make up the command’s program module. If you did specify the stack size when binding the object modules, the program module uses the bound value. You can specify infinity or a value between 32,768 and 1,072,005,120 bytes, inclusive, for this argument. The default value is 8,388,608 bytes. However, this value cannot be greater than the value of the -maximum_stack_limit argument, nor can it be set to infinity.

The -initial_cpu_limit argument specifies the initial maximum number of CPU seconds that a process can consume before it is stopped. You can specify any non-negative value up to infinity for this argument. However, this value cannot be greater than the value of the -maximum_cpu_limit argument.

The -initial_file_limit argument specifies, in bytes, the initial size of a stream file opened for append or output access. You can specify any non-negative value up to infinity bytes for this argument. However, this value cannot be greater than the value of the -maximum_file_limit argument.

The -initial_keep_limit argument specifies whether the operating system should attempt to create a keep module. If you specify a value of 0, the operating system does not attempt to create a keep module. If you specify any other non-negative value up to infinity bytes, the operating system attempts to create a keep module and limits its size to the specified number of bytes. This value cannot be greater than the value of the -maximum_keep_limit argument.

The -initial_port_limit argument specifies the initial number of ports that can be attached by the process. You can specify a value between 5 and 4096, inclusive. The default value is 4096. However, this value cannot be greater than the value of the -maximum_port_limit argument, nor can it be infinity.

The -maximum_total_limit argument specifies, in bytes, the maximum limit for the stack, heap, program module space, and (independent, non-overlapping) connect_vm_region3 space for the specified process. The value that you specify for this argument must be no less than the value of the -initial_total_limit argument. By default, the minimum allowed value is 131,072 bytes, and the maximum allowed value is either 2,145,779,712 bytes or infinity. (That is, you can set this value to infinity, but if you set it to any other value, it must be less than or equal to 2,145,779,712.)

The -maximum_heap_limit argument specifies, in bytes, the maximum limit for the heap in the specified process. The value that you specify for this argument must be no less than the value of the -initial_heap_limit argument and no greater than the value you specify in the -maximum_total_limit argument. By default, the minimum allowed value is 32,768 bytes, and the maximum allowed value is either 2,145,746,944 bytes or infinity. That is, you can set this value to infinity, but if you set it to any other value, it must be less than or equal to 2,145,746,944.
The `maximum_stack_limit` argument specifies, in bytes, the maximum limit for the stack of the specified process. The value that you specify for this argument must be no less than the value of the `initial_stack_limit` argument and no greater than the value you specify in the `maximum_total_limit` argument. By default, the minimum allowed value is 32,768 bytes, and the maximum allowed value is either 1,072,005,120 bytes or `infinity`. Specifying any value between 1,072,005,120 or `infinity` results in an error. However, setting this argument to `infinity` is not advised, because it prevents a program from raising the current stack limit up to the maximum stack limit.

**Note:** If you have abbreviations or command macros that attempt to set `maximum_stack_limit` to a value that is greater than either of the allowed values, the command fails. Change any such references to a value that is less than or equal to the allowed values.

The `maximum_cpu_limit` argument specifies the maximum number of CPU seconds that a process can consume before it is stopped. The value that you specify for this argument must be no less than the value of the `initial_cpu_limit` argument. The maximum allowed value is `infinity`.

The `maximum_file_limit` argument specifies, in bytes, the maximum size of a stream file opened for append or output access. The value that you specify for this argument must be no less than the value of the `initial_file_limit` argument. The maximum allowed value is `infinity`.

The `maximum_keep_limit` argument specifies whether the operating system should attempt to create a keep module. If you specify a value of 0, the operating system does not attempt to create a keep module. If you specify any other non-negative value up to `infinity` bytes, the operating system attempts to create a keep module and limits its size to the number of bytes specified. The value that you specify for this argument must be no less than the value of the `initial_keep_limit` argument.

The `maximum_port_limit` argument specifies the maximum number of ports that can be attached by the process. The value that you specify for this argument must be no less than the value of the `initial_port_limit` argument, and it must be between 5 and 4096. The default value is 4096. You cannot specify a value of `infinity`.

Your system administrator cannot change the **minimum** allowed values for any of the `maximum_*` arguments. However, your system administrator can change the **maximum** default value for a module for any or all of these arguments. If you are a privileged user, you can increase the value and even specify a value larger than the default maximum value for a module. If you do not specify a value, OpenVOS uses the default maximum limit.

**Process Address Space**

Each OpenVOS process has two gigabytes of address space that can be used for various purposes. OpenVOS recognizes five uses: the program module, dynamically-allocated shared virtual memory (SVM), user heap, user stack, and system data. OpenVOS allocates the address space to these five uses in the order shown in Figure 2-6. Note that the process address space may include other, small subregions that are not shown in Figure 2-6.
Figure 2-6. Default Process Address Space on ftServer Modules

Notes:

1. For the specific ranges and default values related to the process address space, see Table B-1.

2. The boundaries between areas that are shown by solid lines are fixed; you cannot adjust them.

3. The boundaries between areas that are shown by broken lines are fluid; you can adjust them.

4. You can adjust the boundary between the SVM and the user heap by setting the value of high_water_mark in the binder control file. This boundary is established when the program starts execution; you cannot change it during execution.
5. The boundary between the user heap and the user stack is dynamically computed by using the values specified for the current stack and heap command limits.

6. If the length of the SVM is 0, OpenVOS does not allow dynamic creation of SVM in the process.

7. The size of the system data area is subject to change in future OpenVOS releases. Therefore, you should avoid writing software that depends on the location or size of this area.

8. The figure is not shown to scale.

A description of the process address space illustrated in Figure 2-6 follows.

- The program module contains a program’s machine code instructions, unshared static data, shared static data, and (optionally) debugger symbol table.

- The shared virtual memory (SVM) is an area of virtual address space shared by several processes.

- The user heap is an area of virtual address space where the operating system allocates dynamic variables using the s$allocate subroutine or language-specific memory allocation statements. Space in a heap must be allocated and freed dynamically.

- The user stack is an area of virtual address space consisting of an ordered series of stack frames associated with the execution of a program.

- The system data is an area of the virtual address space consisting of data for use by the operating system.

Module Default Command Limits

When a new process is created, it takes its first set of initial and maximum command limits from the module’s default command limits. The system administrator establishes these limits.

You can display the module’s default command limits with the list_default_cmd_limits command. Your system administrator can change the initial and maximum default limits with the update_default_cmd_limits command. Both commands are documented in OpenVOS System Administration: Administering and Customizing a System (R281).

Table 2-36 lists the default values (in bytes) for the initial and maximum module default command limits. This table also lists the lower bounds for the initial and maximum limits. The lower bounds cannot be changed; the maximum or initial limits cannot be less than the lower bounds. Table 2-36 shows the default values and lower bounds for an ftServer module.

**Note:** The values shown in Table 2-36 are the operating system default values. However, system administrators frequently specify lower values by executing the update_default_cmd_limits command in module_start_up.cm. For that reason, you should use the list_default_cmd_limits command to view the limits in use on your module.
The following list describes how the process initial, current, and maximum command limit values are determined, and how actual memory use is controlled.

- The process initial command limits are equal to the values in the Initial Limit column in Table 2-36, or to the values of the -initial_total_limit, -initial_heap_limit, and -initial_stack_limit arguments specified by the update_default_cmd_limits command.

- The current command limits are equal to the -max_program_size, -max_heap_size, and -max_stack_size arguments in the bind command (note that these binder values cannot exceed the module default maximum values). If no values are specified for those arguments, then the current command limits are equal to the values in the Initial Limit column in Table 2-36, or to the values of the -initial_total_limit, -initial_heap_limit, and -initial_stack_limit arguments specified with the update_default_cmd_limits or update_process_cmd_limits commands. Your program can modify the current command limits with the s$set_current_cmd_limit subroutine.

- The actual amount of memory used by a program module changes as it executes, but is constrained by the current command limits.

**Note:** At the beginning of a command’s execution, the OpenVOS program loader reserves stack space for a command equal to the value of the bind -max_stack_size argument or update_process_cmd_limits -initial_stack_limit argument. In contrast, the OpenVOS program loader reserves 32 kilobytes for the command’s heap space, no matter the value specified in the bind -max_heap_size argument or update_process_cmd_limits -initial_heap_limit argument. The actual heap space can grow until it reaches the current heap or total process address space limit. The OpenVOS program loader also reserves space for the code, shared, and unshared static regions.

For example, if you specify a 32 MB process initial stack size, a 32 MB process initial heap size, and a 34 MB process total process address space, only 2 MB of space are actually available for code, shared, and unshared regions, the heap, and dynamically-allocated SVM. If you specify an 8 MB process initial stack size, a
34 MB process initial heap size, and a 34 MB process total process address space, 26 MB of space are actually available for code, shared, and unshared regions, the heap, and dynamically-allocated SVM.

- The process maximum command limits are equal to the values in the **Maximum Limit** column in Table 2-36, or are set by the `-maximum_total_limit`, `-maximum_heap_limit`, and `-maximum_stack_limit` arguments in the `update_default_cmd_limits` or `update_process_cmd_limits` commands. Your program can modify the process maximum command limits with the `s$set_current_cmd_limit` subroutine. If privileged, your process can increase or decrease these limits. If not privileged, your process can only decrease these limits.

If you use the `s$set_current_cmd_limit` subroutine to specify a current limit that is equal to the existing usage of a resource, OpenVOS prevents further increases in size. You cannot specify a current limit that is less than the existing usage.

You can specify a combined current stack limit and a current heap limit that is larger than the 1 GB user address space. For example, if you specify a .8 GB maximum stack size and a .8 GB maximum heap size, the operating system will reserve .8 GB for the stack. The heap will only grow to .2 GB minus the size of the code, shared, and unshared static regions. After this point, heap allocations will fail.

**Figure 2-7** illustrates the effects of setting process initial, current, and maximum command limits. For simplicity, assume that this illustration applies only to one type of limit (either heap, stack, or total). The horizontal lines represent the process initial and maximum default command limits and the lowest allowed limits for a module. The numbered vertical lines represent changes in the execution of a command, or the process limits for a command. These changes are described after the figure. The dotted line indicates the actual memory usage.
**Figure 2-7. Effects of Setting Process Initial, Current, and Maximum Command Limits**

**Examples**

**Example 1.**

The following scenario describes how a program can use the `s$set_current_cmd_limit` subroutine to override the initial default command limits for that program.

1. A command (program module) begins execution. Since no memory limits have been bound into the program module, and no initial process command limits have been specified for the process, OpenVOS sets the current limits to the initial default command limits.

2. When the actual memory usage reaches the current limit, the program module calls the `s$set_current_cmd_limit` subroutine to raise the current limit.

3. Execution of the program module terminates. The current limits also expire.

**Example 2.**

The following scenario describes the procedure for and the effect of explicitly lowering the process maximum command limits for all programs started by the current process. It also shows the use of the `s$set_current_cmd_limit` subroutine to raise the current limit of memory usage.

1. The user invokes the `update_process_cmd_limits` command to set the process maximum command limits lower than the default maximum command limits. Another command (program module) begins execution. Memory limits have been bound into this program module. OpenVOS sets the current limits equal to the binder values.
2. When the actual memory usage reaches the current limit, the program module calls the 
$s$set_current_cmd_limit subroutine to raise the current limit.

3. Execution of the program module terminates. The user-specified binder and current 
limits also expire. The process maximum command limits apply to any command 
beginning execution in this process space.

Example 3.
If you invoke the update_process_cmd_limits command and do not specify a value for 
process_name or specify a star name for process_name, the system issues the following 
prompt.

Verify processes to update command limits. 
Smith.Sales(login)? (yes,no,info)

If you type yes at the prompt, the command limits are changed; if you type no, the command 
limits remain the same. If you specify info, the system displays information about the 
subprocess level, program name, and login time of the process. If the process is interactive, 
the system returns the terminal name from which the process was started. The system does 
not return a terminal name if the process is not interactive or if the process is logged in 
remotely from a module that is not running a current version of the operating system. For 
example:

Logged in at 90-01-19 07:33:26 EDT, sub-process level 0. 
Running emacs.pm on $s1#t1.6

If your process is running a program, the system also displays the name of the program 
module and then issues the prompt again. For example:

Logged in at 90-01-19 07:33:26 EDT, sub-process level 0. 
Running on $s1#t1.6 
Smith.Sales(login)? (yes,no,info)

Related Information
To list the initial and maximum resource limits for an existing process, use the 
list_process_cmd_limits command. For information about setting the heap 
(max_heap_size), stack (max_stack_size), and total (max_program_size) limits in a 
bound program module, see the bind command description. For information about setting 
and listing the default resource limits for all new processes on a module, see the descriptions 
of the update_default_cmd_limits and list_default_cmd_limits commands in 
OpenVOS System Administration: Administering and Customizing a System (R281). For 
information about the s$set_current_cmd_limit and s$get_current_cmd_limit 
subroutines, see the OpenVOS Subroutines manuals.
use_abbreviations

Purpose
This command defines the abbreviations file that the command processor will subsequently use to expand abbreviations in your command lines and debugging commands.

Display Form

```
-------------------------- use_abbreviations --------------------------
abbreviations_file_name: [ ]
-off: [ ]
```

Command Line Form

```
use_abbreviations [abbreviations_file_name]
[-off]
```

Arguments
- `abbreviations_file_name`
  The path name of an abbreviations file. By default, the operating system uses the abbreviations file named `abbreviations` in your home directory. You cannot specify both an abbreviations file name and `-off`.

- `-off`
  Stops using abbreviations. You cannot specify both an abbreviations file name and `-off`.

Explanation
The `use_abbreviations` command compiles an abbreviations file, which is used to expand the abbreviations in commands and debugging requests. If you specify `-off`, the operating system stops replacing abbreviations for the life of the current process or until you give the `use_abbreviations` command again.

If you give a path name, the operating system compiles the abbreviations file `abbreviations_file_name` and uses that set of abbreviations. Otherwise, it compiles the file named `abbreviations` in your home directory and uses that file.

The `use_abbreviations` command is commonly included in a user’s startup command macro.
When using abbreviations, you can prevent expansion of part of a command line by enclosing the part you do not want expanded in apostrophes. You can also precede all or part of your command line with an exclamation point; the operating system removes the exclamation point and does not expand any abbreviations in the following portion of that command line.

**Access Requirements**

You must have read access to the abbreviations file.

**Examples**

The following command starts replacing abbreviations using the abbreviations defined in the file `abbreviations` contained in your home directory.

```
use_abbreviations
```

If you want to change some of your abbreviations, edit the abbreviations file with an editor and issue a `use_abbreviations` command to recompile the file.

**Related Information**

See *OpenVOS Commands User’s Guide (R089)* for a description of abbreviations files.
use_message_file

Purpose
This command sets the path name of the message file to be used for your current login session.

Display Form

```
use_message_file
message_file_name: [ ]
```

Command Line Form

```
use_message_file [message_file_name]
```

Arguments

* `message_file_name`
  The path name of a message file. By default, the operating system uses the standard system message file.

Explanation

The `use_message_file` command sets the path name of a message file to be used for your current login session.

Generally, use a new message file when binding and executing a program that requires messages not in the standard system message file, or to change the text of error messages.

The binder can initialize a variable to a message code when the message is in the standard system message file or when you set a new message file with the `use_message_file` command. The binder assigns an error status code number to an external variable that has the same name as the message name in the current message file. When you set a new message file with the `use_message_file` command, the binder uses the error status code numbers from that message file.

Often, you declare such a variable as external rather than initializing it in the program, to allow the binder to initialize the variable. This initializes variables to the correct values regardless of what message file you use. If you initialize the variable in the program, the binder does nothing to the value of the variable.
You must set a new message file to bind any program module that uses messages not in the standard system message file if you want the binder to initialize the variables; you must set a new message file to execute any such program module. The value of `message_file_name` must be a sequential file indexed by name, number, name-to-number, and number-to-name.

**Related Information**

See the description of the `bind` command for information on how the binder initializes message variables. For information on creating a new message file, see the section “Changing the Text of Error Messages” and the description of the `make_message_file` command in the OpenVOS System Administration: Administering and Customizing a System (R281).
Purpose
This command preprocesses, compiles, and/or binds a C program. This command, which is similar to the `gcc` command, provides compatibility between C programs compiled on OpenVOS and C programs compiled on UNIX.

Display Form
None.

Command Line Form
```
vcc file_name... [option_selection...]
```

Arguments
- `file_name`  
  Required
  The name of one or more files to be preprocessed, compiled, assembled, and/or bound. A file-name extension (for example, `.c`) is required.

- `option_selection`
  Specifies one or more options. Table 2-37 briefly summarizes each option.

  Note: Although the `vcc` command does not support all of the `gcc` command’s options, it does support the most commonly-used options.

Table 2-37. The `vcc` Command Options

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>-ansi</code></td>
<td>If you specify <code>-ansi</code>, the compiler uses strict ANSI-C conformance mode. In this mode, the compiler recognizes trigraphs and restricts all language extensions. In addition, the compiler issues diagnostics for programming constructs that violate the ANSI C Standard’s rules. In this mode, the compiler does not produce an object module if it generates any warning or error messages.</td>
</tr>
<tr>
<td><code>-b machine</code></td>
<td>Compiles for the specified architecture. The <code>machine</code> value is <code>processor_name-stratus-vos</code>, where <code>processor_name</code> is any of the following: <code>i386</code>, <code>i486</code>, <code>i586</code>, or <code>i686</code>. The default value is the host architecture.</td>
</tr>
<tr>
<td><code>-c</code></td>
<td>Compiles the file but does not bind it.</td>
</tr>
</tbody>
</table>
vcc

Table 2-37. The vcc Command Options (Continued)

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-Dname[def]</td>
<td>Predefines a macro from the command line.</td>
</tr>
<tr>
<td>-dynamic</td>
<td>Binds file_name with shared libraries, if they are available.</td>
</tr>
<tr>
<td>-E</td>
<td>Preprocesses the file but does not compile or bind it. By default, output goes to standard output, but you can redirect it with the -o option.</td>
</tr>
<tr>
<td>-fpic -fPIC</td>
<td>During compilation, generates position-independent code that is suitable for inclusion in a shared library. Because the vcc command generates position-independent code by default, specifying -fpic or -fPIC has no additional effect.</td>
</tr>
<tr>
<td>-funsigned-char -fsigned-char</td>
<td>Specifies whether char data items that are declared without an explicit signed or unsigned keyword are signed or unsigned. By default, char data items that are declared without an explicit signed or unsigned keyword are unsigned.</td>
</tr>
<tr>
<td>-g</td>
<td>Creates a production symbol table for use in debugging.</td>
</tr>
<tr>
<td>--help</td>
<td>Displays descriptions of vcc command-line options.</td>
</tr>
<tr>
<td>-I dir</td>
<td>Specifies a directory path name in which to search for include (header) files. The directory specified by dir is searched before the directories in the process’s include library paths list. You can specify more than one directory to search by using the -I option more than once. The directories are searched in order, beginning with the one specified in the leftmost -I option.</td>
</tr>
<tr>
<td>-include pathname</td>
<td>Includes the source file’s path name in the compilation before the main source file or files.</td>
</tr>
<tr>
<td>-l name</td>
<td>Searches for and binds in either libname.so or libname.a. The directories searched include several standard system directories plus any that you specify with the -L option. The binder searches and processes libraries and object files in the order in which they are specified.</td>
</tr>
<tr>
<td>-L dir</td>
<td>Adds dir to the search list specified in -l name. This option is passed to the binder as -search.</td>
</tr>
<tr>
<td>-nodefaultlibs</td>
<td>Specifies that only the libraries you specify are passed to the binder.</td>
</tr>
<tr>
<td>-nostdlib</td>
<td>Specifies that no startup files are passed to the binder. In addition, only the libraries you specify are passed to the binder.</td>
</tr>
<tr>
<td>-opath</td>
<td>Specifies the name of an output file. You can use it to set the name of a compiled object or the name of a bound program to the file name given in path.</td>
</tr>
</tbody>
</table>
| -O[n]                 | Optimizes generated code during compilation. You can also specify an optimization level. Valid values for the optimization level given in n are 0 through 3. If you do not specify -O, the default optimization level is 0. If you specify -O but do not specify n, the default optimization level is 2.
The `vcc` command preprocesses, compiles, assembles, and/or binds a C program, depending on the program’s file-name extension and on the options you specify.

**Figure 2-8** illustrates the phases of the `vcc` command.

**Figure 2-8. Phases of the vcc Command**

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>-shared</code></td>
<td>Produces a shared object that can then be bound with other objects to form an executable file. This object is passed to the binder.</td>
</tr>
<tr>
<td><code>-static</code></td>
<td>Instructs the binder not to bind <code>file_name</code> with shared libraries.</td>
</tr>
<tr>
<td><code>-symbolic</code></td>
<td>Binds references to global symbols when building a shared object, and warns about any unresolved references. This option is passed to the binder as <code>-Bs symbolic</code>.</td>
</tr>
<tr>
<td><code>-traditional</code></td>
<td>If you specify <code>-traditional</code>, the compiler uses a transitional conformance mode, allowing certain usages and programming constructs that were common in some older, “traditional” (pre-ANSI) C compilers.</td>
</tr>
<tr>
<td><code>-U name</code></td>
<td>Undefines a specified predefined compiler macro. The compiler does not issue a diagnostic if the given name is not a predefined macro.</td>
</tr>
<tr>
<td><code>-usage</code></td>
<td>Displays descriptions of <code>vcc</code> command-line options. This is the same as specifying the <code>--help</code> option.</td>
</tr>
<tr>
<td><code>-v</code></td>
<td>Causes the compiler to display the subprocess’s command lines to the terminal before invoking them.</td>
</tr>
<tr>
<td><code>-Wc, argument</code></td>
<td>Passes <code>argument(s)</code> directly to the compiler, replacing the commas between arguments with spaces. There can be no spaces between <code>-Wc</code> and <code>argument</code>.</td>
</tr>
<tr>
<td><code>-Wl, argument</code></td>
<td>Passes <code>argument(s)</code> directly to the binder, replacing the commas between argument with spaces. There can be no spaces between <code>-Wl</code> and <code>argument</code>.</td>
</tr>
</tbody>
</table>

**Explanation**

The `vcc` command preprocesses, compiles, assembles, and/or binds a C program, depending on the program’s file-name extension and on the options you specify.
You must always provide the file’s file-name extension to the `vcc` command, except when you specify the `-o` option with an executable file name. In this case, `vcc` automatically adds a `.pm` extension to the file name.

The `vcc` command processes files based on their file-name extension, as described in the following table.

<table>
<thead>
<tr>
<th>File-Name Extension</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>.c</code></td>
<td>The command considers it a C source file and compiles it into an object file.</td>
</tr>
<tr>
<td><code>.i</code>, <code>.s</code>, <code>.S</code></td>
<td>The command returns an error.</td>
</tr>
<tr>
<td>Any other extension, including <code>.o</code> and <code>.a</code></td>
<td>The command considers it an object file and passes it to the binder.</td>
</tr>
</tbody>
</table>

Like OpenVOS files with the `.obj` suffix, files with the `.o` file-name extension are relocatable object files. However, `.o` files and `.obj` files have different internal formats. As a result, you cannot specify `.o` files with many existing OpenVOS commands (for example, `add_entry_names` and `display_object_module_info`).

Other than the OpenVOS binder, OpenVOS commands do not allow you to work with archives. (An `archive` is a single file holding a collection of other files in a structure that makes it possible to retrieve the original individual files.) To create and work with archives, you must use the GNU tools, which are a separately shipped product.

To maximize compatibility with the `gcc` command, `vcc` differs from the `cc` command as shown in Table 2-38.

**Table 2-38. Differences between the vcc and cc Commands**

<table>
<thead>
<tr>
<th>vcc</th>
<th>cc</th>
</tr>
</thead>
<tbody>
<tr>
<td>Uses an implementation of the <code>setjmp()</code> and <code>longjump()</code> runtimes that is <strong>incompatible</strong> with PL/I label variables and nonlocal goto’s.</td>
<td>Uses an implementation of the <code>setjmp()</code> and <code>longjump()</code> runtimes that is <strong>compatible</strong> with PL/I label variables and nonlocal goto’s.</td>
</tr>
<tr>
<td>By default, uses longmap data allocation.</td>
<td>By default, uses shortmap data allocation.</td>
</tr>
<tr>
<td>Truncates external symbol names to 2048 characters.</td>
<td>Truncates external symbol names to 28 or 32 characters.</td>
</tr>
<tr>
<td>By default, uses UNIX-compatible include file search rules.</td>
<td>By default, does not use UNIX-compatible include file search rules.</td>
</tr>
</tbody>
</table>
Examples

The following command preprocesses, compiles, and binds `sample1.c`, producing `sample1.pm`.

`vcc sample1.c`

The following command preprocesses and compiles `sample1.c`, producing `sample1.o`.

`vcc -c sample1.c`

The following command binds `sample1.o`, `sample2.o`, and `sample3.o`, producing `s.pm`.

`vcc -o s.pm sample1.o sample2.o sample3.o`

The following command preprocesses and compiles `sample1.c`, looking in the directory `../include` for header files before looking in the directories specified in the include library paths. Note that `vcc` accepts POSIX-style and OpenVOS-style path names.

`vcc -c -I../include sample1.c`

The following command binds `sample1.o` and `libsamp.a`, producing `sample1.pm`. The archive `libsamp.a` must be in either the directory `../lib` or in one of the directories specified in the object library paths.

`vcc sample1.o -L../lib -lsamp`

The following command binds `sample1.o`, producing `sample1.pm` and `sample1.map`, passing the `-map` and `-size 64mb` options to the binder.

`vcc -Wl,-map -Wl,-size,64mb sample1.o`

Related Information

See the description of the `cc` command in this manual and also the *OpenVOS Standard C User’s Guide* (R364) for information about the OpenVOS Standard C compiler. See the description of the `bind` command for information about shared libraries. For more information about the GNU tools, see the *GNU Tools for OpenVOS: User’s Guide* (R453) and the most recent version of the software release bulletin for *OpenVOS GNU Tools* (R468).
Purpose

This command provides an alternate version of the Emacs text editor for use within the bash shell. It creates stream files, uses OpenVOS key bindings, and accepts POSIX-style path names.

Display Form

```
------------------------------------ emacs -------------------------------------
file_names:                      
-start_up_path:     current_start_up_path_name
-num_windows:       1
-backup:            no
-keystrokes:        no
-keystrokes_dir:    current_directory
-flow_control:      no
-nls:               no
-dictionary:        current_dictionary_path_name
-organization:      stream
-record_size:       
-character_set:     none
-shift_mode:        none
pathname_style:     posix
-compatibility:     vos
```

Command Line Form

```
vemacs [file_names ...]

[ -start_up_path start_up_path_name ]
[ -num_windows number ]
[ -backup ]
[ -keystrokes ]
[ -keystrokes_dir keystrokes_path_name ]
[ -flow_control ]
[ -nls ]
[ -dictionary dictionary_path_name ]
[ -organization organization ]
[ -record_size record_size ]
[ -character_set character_set ]
[ -shift_mode shift_mode ]
[ -pathname_style style_name ]
[ -compatibility method_name ]
```
Arguments

**Note:** Except for the `-organization`, `-pathname_style`, and `-compatibility` arguments, the arguments of the `vemacs` command are identical to those shown in the description of the `emacs` command. See the description of the `emacs` command for more information about the arguments not shown here.

► `-organization organization`<CYCLE>

Specifies one of the following types of file organization for the new file.

- sequential
- stream
- relative
- fixed

By default, the `vemacs` command creates a stream file. In contrast, the non-POSIX `emacs` command creates a sequential file by default.

► `-pathname_style style_name`<CYCLE>

Determines whether Emacs interprets all path names as POSIX-style (slash-separated or greater-than-separated) path names or as OpenVOS-style (greater-than-separated) path names. Possible values for `style_name` are `posix` or `vos`. By default, the `vemacs` command interprets all path names as POSIX-style path names. In contrast, the non-POSIX `emacs` command interprets all path names as OpenVOS-style path names by default.

This argument applies to all input path names, whether on the command line, given to prompts, or processed by the `<ESC><TAB>` completion action. It also applies to the path names that are arguments to the `-dictionary`, `-start_up_path`, and `-keystrokes_dir` arguments. This argument has no effect on output path names; Emacs always displays OpenVOS-style path names.

► `-compatibility method_name`<CYCLE>

Determines whether Emacs commands and mode settings are initialized to their “traditional” values (that is, GNU Emacs values) or to OpenVOS-specific values. Possible values are `vos` or `traditional`. By default, the `vemacs` command (as well as the non-POSIX `emacs` command) initializes its commands and mode settings to OpenVOS-specific values.

Description

The `vemacs` command is a shell script that provides an alternate version of the Emacs text editor for use within the `bash` shell. When you are within the `bash` shell and specify the `vemacs -form` command, the display form for the `emacs` command appears. However, the default values for the `-organization`, `-pathname_style`, and `-compatibility` arguments differ from those of the non-POSIX `emacs` command.

Access Requirements

You need read access to a file in order to read it into an Emacs buffer. To write the contents of an Emacs buffer to a file, you need modify access to the directory and write access to the file (which can be specified in the default access list for the directory or the access list for the file).
Related Information

See the *VOS Emacs User’s Guide* (R093) for a complete description of Emacs requests. See also the descriptions of the `emacs` and `temacs` commands.
**verify_posix_access**

**Purpose**

The `verify_posix_access` command displays a report for a directory or directory hierarchy that summarizes the POSIX.1 access modes for each object.

**Display Form**

```
------- verify_posix_access -------
root_dir: %s#m29>Marketing>Mary_Smith
-depth:
-long: no
```

**Command-Line Form**

```
verify_posix_access
   [root_dir]
   [-depth number]
   [-long]
```

**Arguments**

- `root_dir`
  The root of the directory tree to analyze. The default value is the current directory.

- `-depth number`
  The number of levels of subdirectories to analyze. A level of 1 indicates that only the specified root directory should be analyzed. By default, the command analyzes all subdirectories. The maximum depth is 15.

- `-long`
  The default report shows only the objects that have no POSIX.1 owner or group. This argument displays the status of all objects in the report.

**Explanation**

This command verifies whether an entire directory tree has valid OpenVOS POSIX access control lists. The OpenVOS POSIX.1 implementation stores the UID and GID access permissions in the OpenVOS access control list (ACL) of an object. Native OpenVOS ACLs are considerably more flexible and complex than the types of ACLs created by POSIX.1 Support. Only certain forms of ACLs are compatible with POSIX.1 Support. This command helps you find objects with invalid or incomplete ACLs.

A directory’s ACL completely specifies the users and groups that have status or modify access to the directory and can operate on its immediate contents. A file’s ACL partially specifies the
users and groups that can execute, read, or write it. Files are not required to have an ACL or even a complete ACL; each directory has a default access control list (DACL) that is searched when a file ACL is empty or contains no ACL terms that match the identity of the user accessing the file.

A valid POSIX.1 ACL contains at least three terms. These terms must be present on each directory ACL, and must be present on the combination of the default ACL and the ACL of a specific file. The terms are:

- `<owner-access-mode> Person.*`
- `<group-access-mode> *.Group`
- `<other-access-mode> *.*`

If you specify one `Person.*` term, that term specifies the owner ID and owner access of the object. If you specify more than one `Person.*` term, the single term with the least-restrictive access specifies the owner ID and owner access. If multiple person terms have the same access, the owner ID is undefined.

If you specify one `*.Group` term, that term specifies the group ID and group access of the object. If you specify more than one `*.Group` term, the single term with the least-restrictive access specifies the group ID and group access. If multiple group terms have the same access, the group ID is undefined.

If you specify a `*.*` term, that term specifies other access. If you do not specify any such term, other access is null.

OpenVOS POSIX.1 can always determine Owner, Group and Other access permissions. Any user whose IDs do not match terms on an ACL (or on the combination of the ACL and default ACL) has no access to the object. The challenge is to find a single Owner and Group for an object.

Each line of the report is the status of a single object. The columns are `Owner Access`, `Group Access`, `Other Access`, `Object Type`, and `Object Name`. Each of the access columns is further subdivided into OpenVOS access mode, POSIX access mode, and a name. When a unique Owner or Group name cannot be determined, the respective column is filled with a question mark (`?`).

The first entry displayed for each directory is the DACL. This list should always represent a valid POSIX ACL, since files frequently do not contain their own ACL.

**Access Permission**

This command requires status or modify permission on a directory to analyze and report on it.
verify_posix_access

Examples

!list

Files: 12, Blocks: 0
e 0 e-group
e 0 e-other
e 0 e-owner
n 0 n-group
n 0 n-other
n 0 n-owner
r 0 r-group
r 0 r-other
r 0 r-owner
w 0 w-group
w 0 w-other
w 0 w-owner

!display_access_list *

%sw#m1_user>Demo>Guest>test>e-group
 e *.Stratus
 e *.*

%sw#m1_user>Demo>Guest>test>e-other
 e *.*

%sw#m1_user>Demo>Guest>test>e-owner
 e Guest.*
e *.Stratus
e *.*

%sw#m1_user>Demo>Guest>test>n-group
 n *.Stratus
 n *.*

%sw#m1_user>Demo>Guest>test>n-other
 n *.*

%sw#m1_user>Demo>Guest>test>n-owner
 n Guest.*
 n *.Stratus
 n *.*

%sw#m1_user>Demo>Guest>test>r-group

(Continued on next page)
verify_posix_access

(Continued)

```
r *.Stratus
r *

%sw#m1_user>Demo>Guest>test>r-other
r *

%sw#m1_user>Demo>Guest>test>r-owner
r Guest.*
r *.Stratus
r *

%sw#m1_user>Demo>Guest>test>w-group
w *.Stratus
w *

%sw#m1_user>Demo>Guest>test>w-other
w *

%sw#m1_user>Demo>Guest>test>w-owner
w Guest.*
w *.Stratus
w *

!verify_posix_access test -long
```

<table>
<thead>
<tr>
<th>Owner</th>
<th>Access</th>
<th>Group</th>
<th>Other</th>
<th>Obj</th>
<th>Object</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>--------</td>
<td>--------</td>
<td>-------</td>
<td>-------</td>
<td>-----</td>
<td>--------</td>
</tr>
<tr>
<td>w</td>
<td>rwx</td>
<td>Guest</td>
<td>n</td>
<td>---</td>
<td><em>.</em></td>
</tr>
<tr>
<td>e</td>
<td>--x</td>
<td>Stratus</td>
<td>e</td>
<td>--x</td>
<td><em>.</em></td>
</tr>
<tr>
<td>e</td>
<td>--x</td>
<td>Stratus</td>
<td>e</td>
<td>--x</td>
<td><em>.</em></td>
</tr>
<tr>
<td>n</td>
<td>---</td>
<td>Stratus</td>
<td>n</td>
<td>---</td>
<td><em>.</em></td>
</tr>
<tr>
<td>n</td>
<td>---</td>
<td>Stratus</td>
<td>n</td>
<td>---</td>
<td><em>.</em></td>
</tr>
<tr>
<td>n</td>
<td>---</td>
<td>Stratus</td>
<td>n</td>
<td>---</td>
<td><em>.</em></td>
</tr>
<tr>
<td>r</td>
<td>r-x</td>
<td>Stratus</td>
<td>r</td>
<td>r-x</td>
<td><em>.</em></td>
</tr>
<tr>
<td>r</td>
<td>r-x</td>
<td>Stratus</td>
<td>r</td>
<td>r-x</td>
<td><em>.</em></td>
</tr>
<tr>
<td>r</td>
<td>r-x</td>
<td>Stratus</td>
<td>r</td>
<td>r-x</td>
<td><em>.</em></td>
</tr>
<tr>
<td>w</td>
<td>rwx</td>
<td>Stratus</td>
<td>w</td>
<td>rwx</td>
<td><em>.</em></td>
</tr>
<tr>
<td>w</td>
<td>rwx</td>
<td>Stratus</td>
<td>w</td>
<td>rwx</td>
<td><em>.</em></td>
</tr>
<tr>
<td>w</td>
<td>rwx</td>
<td>Guest</td>
<td>w</td>
<td>rwx</td>
<td><em>.</em></td>
</tr>
</tbody>
</table>

(Continued on next page)
verify_posix_access

(Continued)

<table>
<thead>
<tr>
<th>Owner</th>
<th>Group</th>
<th>Other</th>
<th>Obj</th>
<th>Object</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>e</td>
<td>e</td>
<td>Stratus</td>
<td>e</td>
<td><em>.</em> file e-group</td>
</tr>
<tr>
<td>e</td>
<td>e</td>
<td>Stratus</td>
<td>e</td>
<td><em>.</em> file e-other</td>
</tr>
<tr>
<td>n</td>
<td>n</td>
<td>Stratus</td>
<td>n</td>
<td><em>.</em> file n-group</td>
</tr>
<tr>
<td>r</td>
<td>r</td>
<td>Stratus</td>
<td>r</td>
<td><em>.</em> file r-group</td>
</tr>
<tr>
<td>w</td>
<td>w</td>
<td>Stratus</td>
<td>w</td>
<td><em>.</em> file w-group</td>
</tr>
<tr>
<td>w</td>
<td>w</td>
<td>Stratus</td>
<td>w</td>
<td><em>.</em> file w-other</td>
</tr>
</tbody>
</table>

%sw#m1_user>Demo>Guest>test

---

2-828
verify_save

Purpose
This command verifies that an object or set of objects, saved with the save_object or save command, is restorable.

Display Form

--- verify_save ---
tape_device_or_port_name: 
pathnames: ...
-volume_id: short
-output: yes
-verify_next_reels: no
-unattended: yes

Command Line Form

verify_save tape_device_or_port_name
   [pathnames]...
   [-volume_id volume_id]
   [-output output_type]
   [-no_verify_next_reels]
   [-unattended]

Arguments

► tape_device_or_port_name Required
The name of the tape device or port attached to the tape drive or disk file that contains the saved object or set of saved objects to be verified.

► pathnames
One or more path names or star names of saved objects to be verified. By default, the command verifies all objects on the save file or tape.

► -volume_id volume_id
Specifies the volume ID of the tape containing the saved object, or of the first tape in a set of tapes containing the saved object or objects. If the port is attached to a disk file, this argument is ignored. By default, the verify_save command prompts you for a volume ID.

► -output output_type [CYCLE]
Specifies the type of information to be reported by verify_save. The possible values for output_type are none, short, and long. The short report displays the names of the saved objects verified. The long report also includes each save_block from the
The `verify_save` command verifies that a save file, a save tape or series of save tapes, or an object (a directory, file, or link) saved earlier with a `save` or `save_object` command can be read later and restored with a `restore` or `restore_object` command.

If your port is attached to a tape drive, the `verify_save` command mounts the tape, unless it has already been mounted (with the `mount_tape` command). When the tape is already mounted, the `verify_save` command checks the volume ID specified against the volume ID of the tape. If they are different, the command asks if you want to use the volume ID of the tape. If so, it replaces the volume ID specified with the volume ID of the tape; otherwise, the command aborts. You specify only the first volume ID, since the `save` or `save_object` commands store the volume ID of any subsequent save tapes at the end of each save tape.

You must switch tapes manually when saved objects are located on more than one tape.

When specifying the path name of an object to be verified, use its path name at the time it was saved. This path name identifies the object on the save file or tape.

If the saved object is a directory, the `verify_save` command verifies that all objects in that directory (all files, links, and subdirectories) can be read and restored later.

You can use the `verify_save` command on DAE files.

**Access Requirements**

You need read access to the tape drive or disk file attached to your port.

**Examples**

Suppose `a_port` is a port attached to a tape drive on which you have mounted a tape volume containing saved objects. The following command displays a short report listing the names of all the objects saved on that volume that can be read and restored.

```
verify_save a_port
```
The report might look like this.

<table>
<thead>
<tr>
<th>save_file root_path:</th>
<th>%s1#d02&gt;Sales&gt;Smith&gt;reports</th>
</tr>
</thead>
<tbody>
<tr>
<td>directory:</td>
<td>%s1#d02&gt;Sales&gt;Smith&gt;reports</td>
</tr>
<tr>
<td>file_name:</td>
<td>%s1#d02&gt;Sales&gt;Smith&gt;weekly_report.FW739</td>
</tr>
<tr>
<td>file_name:</td>
<td>%s1#d02&gt;Sales&gt;Smith&gt;weekly_report.FW740</td>
</tr>
<tr>
<td>file_name:</td>
<td>%s1#d02&gt;Sales&gt;Smith&gt;weekly_report.FW741</td>
</tr>
<tr>
<td>file_name:</td>
<td>%s1#d02&gt;Sales&gt;Smith&gt;weekly_report.FW742</td>
</tr>
<tr>
<td>file_name:</td>
<td>%s1#d02&gt;Sales&gt;Smith&gt;weekly_report.FW743</td>
</tr>
<tr>
<td>file_name:</td>
<td>%s1#d02&gt;Sales&gt;Smith&gt;weekly_report.FW744</td>
</tr>
<tr>
<td>file_name:</td>
<td>%s1#d02&gt;Sales&gt;Smith&gt;weekly_report.FW746</td>
</tr>
<tr>
<td>file_name:</td>
<td>%s1#d02&gt;Sales&gt;Smith&gt;weekly_report.FW747</td>
</tr>
</tbody>
</table>

In another example, if you specify the -output long argument for a saved object that contains expandable directories, the output displays the *_save_block types.

```
verify_save port -output long

1 global_save_block (version: 1)
    save_file root_path: %s1#d02>Sales>Jones

2 directory_save_block (version: 2)
    object id: 1
    directory: %s1#d02>Sales>Jones

...  
3 dir_limit_info1_save_block (version: 1)
    object id: 2
    version: 3
    max entries: 32700
    dft max entries: 0
    max blocks: 8720
    dft max blocks: 0
    path: %s1#d02>Sales>Jones
```

**Related Information**

See the descriptions of the list_save_tape, save_object, and restore_object commands for more information on backing up objects. For information on system backup, see the discussion on making and retrieving backup and the save and restore command descriptions in the *OpenVOS System Administration: Backing Up and Restoring Data* (R285). See also the command descriptions of copy_tape, create_tape_volumes, dismount_tape, display_tape_params, dump_tape, mount_tape, position_tape, save_object, set_second_tape, set_tape_drive_params, set_tape_mount_params, set_tape_file_params, and write_tape.
**verify_system_access**

**Purpose**

This command checks that you can use the resources of a system to access files and to start processes in that system.

**Display Form**

```
----------------------------- verify_system_access -----------------------------
  system_name: [ ]
  -privileged: as registered
  -password: 
  -priority: 
```

**Command Line Form**

```
verify_system_access system_name
  [-privileged]
  [-password password]
  [-priority priority]
```

**Arguments**

- **system_name**  
  The name of the system you want to use.

- **-privileged**  
  Checks that you can perform a privileged operation in the specified system. The operating system tells you if you cannot perform a privileged operation. By default, the operating system does not warn you if you cannot perform a privileged operation.

- **-password password**  
  Specifies a password to use the system. You need not give it again during your current interactive process. By default, the operating system asks you for it, unless you are not required to supply one. The password you enter when prompted does not appear on your terminal.

  **Note:** Abbreviations are not expanded.

- **-priority priority**  
  Checks that you can start a process with this priority. The operating system tells you if you cannot start a process with this priority.
**Explanation**

The `verify_system_access` command checks your access rights to the resources of another system.

Before you can start a process with the `start_process` command, run a batch job with the `batch` command, print a file with the `print` command, or access files in a system other than your current system, the operating system must check that you have the proper access rights. The `verify_system_access` command confirms your access rights. The operating system retains the results of this check so that when you issue any command that accesses or uses a module on the remote system, the operating system can complete the command.

If a remote system does not require you to supply a password, then you do not need to issue the `verify_system_access` command to use the system.

**Examples**

The following command checks your access rights to the system `s1`.

```
verify_system_access s1 -password xskk
```

If you do not have the necessary access, the operating system displays this message.

```
You are not a registered user of the target system.
```

**Related Information**

See the command descriptions of `start_process`, `print`, and `batch`. For more information about getting and setting the access for particular files and directories, see the command descriptions of `display_access`, `display_access_list`, `display_default_access_list`, and `give_default_access`.
vospath

Purpose
This command converts a POSIX path name into an OpenVOS path name.

Display Form
None.

Command Line Form
vospath [path_name] [--help] [--version]

Arguments

- **path_name**
  The name of a file, or a relative or absolute path name.

- **--help**
  Displays a brief help message that describes the command.

- **--version**
  Displays a brief version string.

Explanation
The `vos_path` command converts the `path_name` argument, which is a relative or full POSIX path name, into a full OpenVOS path name. The resultant OpenVOS path name always begins with a percent-sign character (%).

If `path_name` has the form `/dev/null`, the result is `%sys#null`, where `%sys` is the current system. Otherwise, `path_name` is expanded into a full OpenVOS path name, which processes and removes any dot components (., ..).

The three arguments are all mutually exclusive.

This command is primarily for use with interactive shells such as `bash`. Users who are experienced with OpenVOS commands may prefer to use the `(vos_path)` command function instead of the `vospath` command.
**Examples**

The following command converts a POSIX path name into an OpenVOS path name.

```
vos_path /system/gnu_library/bin
```

```
%es#mRaid4>system>gnu_library>bin
```

**Related Information**

See the description of the `posixpath` command and the description of the `(vos_path)` command function.
walk_dir

Purpose
This command executes a command or starts a process in a specified directory and its subdirectories.

Display Form

<table>
<thead>
<tr>
<th>Argument</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>directory_name</td>
<td>current_directory</td>
</tr>
<tr>
<td>-depth</td>
<td>-</td>
</tr>
<tr>
<td>-down_command</td>
<td>-</td>
</tr>
<tr>
<td>-down_start_process</td>
<td>no</td>
</tr>
<tr>
<td>-up_command</td>
<td>-</td>
</tr>
<tr>
<td>-up_start_process</td>
<td>no</td>
</tr>
<tr>
<td>-brief</td>
<td>-</td>
</tr>
</tbody>
</table>

Command Line Form

walk_dir [directory_name]
  [-depth depth_number]
  [-down_command down_command_line]
  [-down_start_process]
  [-up_command up_command_line]
  [-up_start_process]
  [-brief]

Arguments

- **directory_name**
  The path name of the directory at the top of the hierarchy. By default, the command uses the current directory. The walk_dir command visits the subdirectories of directory_name.

- **-depth depth_number**
  Sets the number of levels from the root directory that the command visits. By default, walk_dir visits all the directories in the hierarchy.

- **-down_command down_command_line**
  Specifies a command line to be executed in each directory visited before visiting any of its subdirectories. If the command line contains spaces, semicolons, or command functions, the entire line must be enclosed in apostrophes. Unless you specify...
-down_start_process, you can only use internal commands in the command line. Either -down_command or -up_command is required.

**-down_start_process**
Starts a subprocess to execute the command line down_command_line. By default, your process directly executes the command line.

**-up_command up_command_line**
Specifies a command line to be executed in each directory visited after visiting its subdirectories. If the command line contains spaces, semicolons, or command functions, the entire line must be enclosed in apostrophes. Unless you specify -up_start_process, you can only use internal commands in the command line. Either -up_command or -down_command is required.

**-up_start_process**
Starts a subprocess to execute the command line up_command_line. By default, your process directly executes the command line.

**-brief**
Suppresses the display of each directory that matches a star name, before the command executes. By default, the walk_dir command displays the names of the directories.

**Explanation**

The walk_dir command executes a command or starts a subprocess to execute a command in the subdirectories of directory_name. The walk_dir command executes the command line down_command_line and the command line up_command_line once in every directory of the hierarchy. (If you do not specify either command line, the walk_dir command does nothing.)

To execute the commands in all the directories, the walk_dir command changes the current directory successively to each subdirectory in the subhierarchy, and then issues the command. It executes the down_command_line in any directory in the hierarchy before it executes the down_command_line in any directory below it. It executes up_command_line in any directory in the hierarchy after it executes the up_command_line in all directories below it. Unless you use the -down_start_process or -up_start_process arguments, you can only use internal commands in the command line. Use the help command to display a list of these commands.

The directory specified in directory_name is the root directory. The depth of the root directory is 1, the depth of any directory it contains is 2 (relative to the root), the depth of a directory contained by a level-2 directory is 3, and so forth. By default, the walk_dir command walks to the bottom of the subhierarchy.

The walk_dir command does not chase links. If a directory contains links to other directories, the command line specified in the walk_dir command is not executed in the linked directories.

**Access Requirements**

You need status access to a directory to walk through its subhierarchy.
Examples

To display all of the procedure names of the PL/I source modules in the directory hierarchy
%s1#d02>Sales>Jones, use the following command.

```
walk_dir %s1#d02>Sales>Jones -down_command 'list *.pl1'
```

The command visits all the directories contained in the directory and displays the following
information.

```
%s1#d02>Sales>Jones
list: No match for star name. %s1#d02>Sales>Jones>*.pl1
%s1#d02>Sales>Jones>memos
list: No match for star name. %s1#d02>Sales>Jones>memos>*.pl1
%s1#d02>Sales>Jones>quarterly
Files: 1, Blocks: 1
w      1 sales_report.pl1
```
where_command

Purpose
This command tells you the type of a given command and the full path name of an external command or command macro. The command uses your process’s command library path names to locate the command.

Display Form

```
command_name: command_name
-use_abbreviations: no
```

Command Line Form

```
where_command command_name
    [-use_abbreviations]
```

Arguments

- **command_name**
  The name of a command. It can be an internal command, an external command, a command macro, or any other program module.

- **-use_abbreviations**
  According to your command search library paths, expands first and all abbreviations before the search for the command is performed. By default, the command does not expand abbreviations before the search occurs.

Explanation
The `where_command` command tells you the location of the command found using the current command library path names. The search follows standard search rules, looking first for an internal command, then through the command libraries, searching each directory for a command macro, then for a program module.

The argument `command_name` can be an internal command, an external command, a command macro, or any user program or command macro. Unless the command is an internal command, the operating system displays its full path name.

The operating system also displays the type of the command. The possible values are internal command, external command, and command macro.
where_command

When you give the -use_abbreviations argument, the command expands all abbreviations, including first token abbreviations, before it performs the search, according to the command search library paths defined in your process.

Examples

For the following examples, suppose you have this abbreviation in your abbreviations file.

```
first ls -by list -header
```

If you invoke where_command without the -use_abbreviations argument, it does not expand the abbreviation ls, and it does not find the list command.

```
where_command ls
where_command: Object not found. ls
```

However, if you give the -use_abbreviations argument, where_command expands the ls abbreviation, and it finds the list command.

```
where_command ls -use_abbreviations
list: (internal command)
```

Related Information

For a complete description of the search rules, see OpenVOS Commands User’s Guide (R089).
where_path

Purpose

This command locates an object specified by a path name. When the path name is a link, the command displays the ultimate target of the path name, a statement whether the target exists, and the name of the processing module containing the target. When the path name does not include a link, the command displays the full path name.

Display Form

<table>
<thead>
<tr>
<th>path_name:</th>
<th>where_path</th>
</tr>
</thead>
<tbody>
<tr>
<td>-link: no</td>
<td></td>
</tr>
</tbody>
</table>

Command Line Form

```
where_path path_name
    [-link]
```

Arguments

- **path_name**
  - Required
  - The path name of an object. When the path name is a link, the command displays the full path name of its ultimate target.

- **-link**
  - Displays all links encountered while resolving the path name. By default, the where_path command displays only the ultimate target of the link.

Explanation

The where_path command displays the full path name of the ultimate target of the path name `path_name`. The primary use of the where_path command is to determine the ultimate target of a link, so it is most often used when `path_name` is a link name.

When you specify `-link`, where_path displays all the links in the chain between the specified link name and its ultimate target, including any links embedded in path names. Embedded links are displayed in an indented fashion, to indicate the level in the hierarchy at which the link was encountered. In the case of a circular link, the command displays the links causing the circular link and reports the error accordingly. The command also tells you the name of the module of the ultimate target and whether the target object exists.
**Examples**

Consider the following command.

```
where_path jones_customers
```

If the object does exist, then this message appears.

```
%s1#d03>Sales>east>Jones>customers is a directory on %s1#m1.
```

If the indicated path name is the ultimate target of the link *jones_customers*, but the object does not exist, the command displays this message.

```
%s1#d02>Sales>east>Jones>customers does not exist but would be on +%s1#m1.
```

If *acc* is a link in *%s1#d02* to a directory on another module, the command might display this message.

```
%s1#d04>Accounting>journal is a directory on %s1#m4.
```

If the ultimate target of a link is also its immediate target, `-link` has no effect on the result of the command. If there are intervening links, `-link` is necessary to display them.

```
acc -> %s1#d02>Sales>journal  
journal -> %s1#d04>Accounting>journal  
%s1#d04>Accounting>journal is a directory on %s1#m4.
```
who_locked

Purpose
This command displays the names of the processes and their owners that currently have locked a given file or device.

Display Form

Command Line Form

```
who_locked path_names
    [-brief]
```

Arguments

- path_names
  Required
  The path name or star name of a file, or the path name of a device. The who_locked command tells you who locked the file or who currently has the device attached.

- brief
  Displays only those files that are locked. By default, the who_locked command displays the names of all files (locked and unlocked).

- process_id
  Displays the process ID of the locking process. The process ID is displayed as an 8-digit hexadecimal value.

Explanation
The who_locked command displays the names of the processes and the names of the users owning the processes that currently have locked the specified file or device. The output includes the path name of each process and the module on which it is running. The who_locked command also tells you whether the file or device is locked for reading or writing. The who_locked command does not display the names of those who have opened the file for dirty input if there are other locks on the file being used (such as implicit locks or record locks).
Examples

The following example shows the output from the command line.

```
who_locked >Overseer>watch_net*.
```

```
watch_net.cm:
  Object is read locked by Overseer.System (WatchNet) on module %s1#m2 executing watch_kernel.pm.
watch_net.out:
  Object is implicitly locked by Overseer.System (WatchNet) on module %s1#m2 executing watch_kernel.pm.
watch_net.out.old:
  Object is not locked.
```

The following example shows the output when you specify the `-process_id` argument.

```
who_locked >system>command_library>overseer.pm -process_id overseer.pm
```

```
overseer.pm:
  Object is read locked by Overseer.System (TheOverseer, pid 01208060) on module %es#m125 executing overseer.pm.
```

Related Information

See also the description of the `list_users` command.

For information about opening a file for dirty input, see the `$open` subroutine in the OpenVOS Subroutines manuals.
**write_tape**

**Purpose**
This command writes a file or files to the tape volume on the specified tape drive or on the tape drive connected to the specified port.

**Display Form**

```
---------------------------------- write_tape ----------------------------------
  tape_device_or_port_name:  file_names:  -file_ids:  -raw_stream_files:
```

**Command Line Form**

```
write_tape  tape_device_or_port_name
  file_names...
  [-file_ids  file_ids...]
  [-raw_stream_files  [record_size]]
```

**Arguments**

- **tape_device_or_port_name**  
  Required
  The name of the tape device, or the name of the port attached to the tape drive, holding the tape to which files are to be written.

- **file_names**  
  Required
  One or more names or star names of disk files that write_tape is to write to the tape. The names cannot be extended names.

- **-file_ids  file_ids**
  Specifies the file IDs that write_tape is to write in the file labels of the tape files when it writes to a labeled tape. If you give more file IDs than file names (or file names that match file_names), the command disregards the extra file IDs. If you give more file names than file IDs, the command uses the names of the files for the missing tape file IDs.

- **-raw_stream_files [record_size]**
  Reads the disk file to the tape in increments of record_size characters using the s$read_raw subroutine. The record_size is the record size of the resulting tape file; it must be an integer. If you do not specify a record size, the size of the increment is 512. This argument can be used only when writing stream files. By default, the command reads the disk file to the tape a record at a time.
write_tape

Explanation

The write_tape command writes the files selected by *file_names* to the tape volume mounted on the specified tape drive or on the tape drive connected to the specified port.

If you have not yet used the *attach_port* command to attach the port, the write_tape command implicitly attaches a port. If you have not yet used the *mount_tape* command to mount the tape, the write_tape command implicitly mounts the tape before executing. When execution is completed, if write_tape implicitly mounted a tape, it implicitly dismounts the tape. If it implicitly attached a port, it implicitly detaches the port. For more information, see the Explanation section in the *mount_tape* command description.

When write_tape mounts the tape implicitly during multiple write operations, the tape is rewound after the command executes; therefore, each execution overwrites the results of the previous execution. Each time you issue the command, the following message is displayed: Overwrite the existing content of this tape? Respond with yes or no to this question.

The write_tape command writes the first file at the current position of the tape. Therefore, for consecutive write operations to the same tape, use *mount_tape* to mount the tape so that the tape is not rewound between operations. After writing all the files listed under *file_names*, write_tape writes an end-of-volume mark and leaves the tape positioned to the end of the volume. You cannot access data after the end-of-volume mark unless you treat the tape as unlabeled.

The write_tape command determines the organization of the tape files it writes from the default tape parameters for the port specified in *tape_device_or_port_name*.

The write_tape command stores the character set information, that is, the character set and shift mode, with the file that it writes to tape.

The write_tape command writes only the records of a file. It does not write any indexes or attributes of the file.

Access Requirements

By default, you have write access to a tape device. If your system administrator restricts access to the tape device, you need write access to write to the tape device.

Related Information

See also the command descriptions of *copy_tape*, *create_tape_volumes*, *dismount_tape*, *display_tape_params*, *dump_tape*, *list_save_tape*, *mount_tape*, *position_tape*, *read_tape*, *restore_object*, *save_object*, *set_second_tape*, *set_tape_drive_params*, *set_tape_mount_params*, *set_tape_file_params*, and *verify_save*. 
Tape parameters control all aspects of the use of tape drives. Tape parameters allow you to use a tape drive in a specific way, to define special properties of a mounted tape, or determine how an object is output to tape. The tape parameters are divided into three types: drive, mount, and file. Tape drive parameters are in effect for the entire time the port is attached. Tape mount parameters are in effect only while the tape is mounted. Tape file parameters are only in effect while a tape file is open.

The tape drive, mount, and file parameters are divided into default, user, and actual values. The relationship between tape parameter types and values is shown in Table A-1. Default tape values are system defaults and are never changed. The user sets the user tape values. User tape values overlay the defaults and are in effect for a limited period, as defined in Table A-1. Actual tape values are the user values overlaid on the default values and the default values that have not been overlaid by user values. In the case of the mount and file parameters, they may also include values from the tape label. The actual values are used for every tape operation.

The relationship between the tape drive, mount, and file parameters and the default, user, and actual values is shown in Table A-1. The duration heading in Table A-1 is the length of time the default, user, and actual values are in effect.

Table A-1. Setting Tape Parameters

<table>
<thead>
<tr>
<th>Parameter Type</th>
<th>How Set</th>
<th>When Set</th>
<th>Duration Default Values</th>
<th>User Values</th>
<th>Actual Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Drive</td>
<td>set_tape_drive_params</td>
<td>Any time after port is explicitly or implicitly attached.</td>
<td>System defaults. In effect unless user value set.</td>
<td>You set. In effect for attached port.</td>
<td>Default plus user values. In effect for attached port.</td>
</tr>
<tr>
<td>Mount</td>
<td>set_tape_mount_params</td>
<td>After port attached with attach_port but before tape mount.</td>
<td>System defaults. In effect unless user value set or tape label used.</td>
<td>You set. In effect for mounted tape.</td>
<td>Default plus user values or tape label values. In effect for mounted tape.</td>
</tr>
</tbody>
</table>
The duration of user and default tape drive, tape mount, and tape file values is further illustrated in Figure A-1.

<table>
<thead>
<tr>
<th>Parameter Type</th>
<th>How Set</th>
<th>When Set</th>
<th>Duration Default Values</th>
<th>User Values</th>
<th>Actual Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mount</td>
<td>mount_tape</td>
<td>During tape mount.</td>
<td>System defaults. In effect unless user value set or tape label used.</td>
<td>You set. In effect for mounted tape.</td>
<td>Default plus user values or tape label values. In effect for mounted tape.</td>
</tr>
<tr>
<td>File</td>
<td>set_tape_file_params</td>
<td>After tape mounted with mount_tape.</td>
<td>System defaults. In effect unless user value set or tape label used.</td>
<td>You set. In effect for open file.</td>
<td>Default plus user values or tape label values. In effect for open file.</td>
</tr>
</tbody>
</table>
You can use the `set_tape_drive_params`, `set_tape_mount_params`, and `set_tape_file_params` commands to change the parameter values for a tape operation so that you can use a tape drive in a specific way, define special properties of a mounted tape, or determine how an object is output to tape. All of these parameters have system default values.

**Note:** Except when you are writing tapes for use by non-Stratus systems, you usually will have no need to change these default values or to use the `set_tape_drive_params`, `set_tape_mount_params`, or `set_tape_file_params` commands.

Setting and displaying tape parameters with the `set_tape_drive_params`, `set_tape_mount_params`, `set_tape_file_params`, and `display_tape_params` commands is discussed below.
Setting and Displaying Tape Parameter Values

**Setting Tape Drive Parameters**
You can set four tape drive parameter values with the `set_tape_drive_params` command, including values that determine the following: how a tape is positioned at the end of a tape operation, whether a tape remains loaded at the end of a tape operation, what kind of messages a tape user receives, and whether a tape is rewound during a multivolume tape operation.

**Note:** If you reset the user tape drive parameters with the `set_tape_drive_params` command, the new user values are overlaid on the existing user and default values.

**Setting Tape Mount Parameters**
You can set the tape mount parameter values with the `set_tape_mount_params` or `mount_tape` commands. You use the `set_tape_mount_params` command to change parameters prior to mounting a tape; it determines most of the `mount_tape` command’s user values. You use the `mount_tape` command to change the mount parameters during a tape mount. The `set_tape_mount_params` command allows you to change six mount tape parameter values; the `mount_tape` command allows you to change eight mount tape parameter values; both commands allow you to change the tape format, density, volume ID, owner ID, and access rights.

**Note:** If you reset the user tape mount parameters with the `set_tape_mount_params` or `mount_tape` command, the new user values are overlaid on the existing user and default values.

**Setting Tape File Parameters**
You can set eight tape file parameter values with the `set_tape_file_params` command including, most importantly, the file format, block length, record length, and blocking factor.

**Note:** If you reset the user tape file parameters with the `set_tape_file_params` command, the new user values are overlaid on the existing user and default values.

**Displaying Tape Parameters**
You can display the tape drive, mount, and file parameters and their default, user, and actual values with the `display_tape_params` command. The display form of the `set_tape_drive_params`, `set_tape_mount_params`, and `set_tape_file_params` commands do not show any default values for [CYCLE] field parameter values.
Appendix B: General OpenVOS Software Limits and Numerical Definitions

This appendix contains a list of general OpenVOS software limits and numerical definitions.

Table B-1 contains a list of general OpenVOS software limits.

Table B-1. General OpenVOS Software Limits

<table>
<thead>
<tr>
<th>Subject</th>
<th>Description</th>
<th>Limit</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abbreviations</td>
<td>Maximum length of an abbreviation name</td>
<td>32 characters</td>
<td></td>
</tr>
<tr>
<td>Address Space Size</td>
<td>Maximum size of the user address space</td>
<td>2 GB</td>
<td></td>
</tr>
<tr>
<td>Blocks: Maximum in a Non-extent File</td>
<td>Sequential</td>
<td>523,783</td>
<td>Extended sequential files must have extents</td>
</tr>
<tr>
<td></td>
<td>Stream, 64-bit stream, relative, fixed</td>
<td>523,792</td>
<td></td>
</tr>
<tr>
<td>Blocks: Maximum in an Extent File</td>
<td>Sequential</td>
<td>523,783</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Extended sequential [N]</td>
<td>( \min ((523,792 \times E) - 9, (524,288 \times N)) )</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Fixed</td>
<td>( \min ((523,792 \times E), (524,288 \times N)) )</td>
<td></td>
</tr>
<tr>
<td></td>
<td>64-bit stream (with fixed extents)</td>
<td>523,792 \times E</td>
<td></td>
</tr>
<tr>
<td></td>
<td>64-bit stream (with flexible extents)</td>
<td>131,870,736</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Relative</td>
<td>( \min ((523,792 \times E), (524,288 \times (N+2))) )</td>
<td></td>
</tr>
<tr>
<td>Command Functions</td>
<td>Maximum length of a returned value</td>
<td>256 characters</td>
<td></td>
</tr>
</tbody>
</table>
### Table B-1. General OpenVOS Software Limits (Continued)

<table>
<thead>
<tr>
<th>Subject</th>
<th>Description</th>
<th>Limit</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Command Lines</td>
<td>Maximum length of a command line interpreted by command and command macro processor</td>
<td>32,767 characters</td>
<td>Only with sufficient available kernel memory.</td>
</tr>
<tr>
<td></td>
<td>Maximum length of a command line entered on a terminal or virtual terminal</td>
<td>300 characters</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Maximum length of individual words and quoted strings in command lines</td>
<td>256 characters</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Maximum number of command or command macro parameters</td>
<td>60 parameters</td>
<td>The actual limit is determined by the screen size and whether the display form uses two columns</td>
</tr>
<tr>
<td></td>
<td>Maximum number of lines of source code in an OpenVOS compilation unit</td>
<td>2,147,483,647</td>
<td>This limit includes the source file and all include files referenced by the compilation unit.</td>
</tr>
<tr>
<td>Directories</td>
<td>Maximum number of disk blocks that can be assigned to a directory</td>
<td>527 blocks (settable to less)</td>
<td>Expandable limit is 8720 blocks and 32,700 entries (settable to less)</td>
</tr>
<tr>
<td></td>
<td>Maximum depth of directory tree</td>
<td>25 directories</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Maximum length</td>
<td>255 characters if version 2 extended names is supported; otherwise, 32 characters</td>
<td></td>
</tr>
<tr>
<td>Error Codes</td>
<td>Range of values for codes used by OpenVOS</td>
<td>1 to 32,767</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Range of values for codes you can use</td>
<td>-1 to -32,768</td>
<td></td>
</tr>
</tbody>
</table>
Table B-1. General OpenVOS Software Limits *(Continued)*

<table>
<thead>
<tr>
<th>Subject</th>
<th>Description</th>
<th>Limit</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Events</td>
<td>Maximum number of events per module</td>
<td>524,287 events</td>
<td>You can increase this value to 32,767 with the <code>set_tuning_parameters</code> command</td>
</tr>
<tr>
<td></td>
<td>For a tasking program, maximum number of events per task</td>
<td>By default, 8704 events</td>
<td>For information on changing the number of events, see the description of the <code>-max_events_per_task</code> argument of the <code>set_tuning_parameters</code> command in <em>OpenVOS System Administration: Administering and Customizing a System</em> (R281)</td>
</tr>
<tr>
<td></td>
<td>For a nontasking program, maximum number of events per process</td>
<td>By default, 8704 events</td>
<td></td>
</tr>
<tr>
<td>Files</td>
<td>Maximum number of concurrent opens</td>
<td>32,767</td>
<td></td>
</tr>
<tr>
<td>File Names</td>
<td>Maximum length of a file name</td>
<td>32 characters</td>
<td>Specified values less than the minimum value are replaced by the minimum value; specified values greater than the maximum value are replaced by the maximum value</td>
</tr>
<tr>
<td></td>
<td>Maximum length of a file name if neither version 1 nor version 2 extended-names support is enabled; otherwise, 255 characters</td>
<td></td>
<td></td>
</tr>
<tr>
<td>High-Water Mark</td>
<td>Range of values for the high_water_mark binder control file directive</td>
<td>Ranges from 08000000x to 78000000x; the default is 40000000x</td>
<td></td>
</tr>
</tbody>
</table>
### Table B-1. General OpenVOS Software Limits (Continued)

<table>
<thead>
<tr>
<th>Subject</th>
<th>Description</th>
<th>Limit</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Indexes</td>
<td>Maximum length of a key field</td>
<td>For embedded-key indexes, the maximum length is the total size of the specified key components (up to 1280 bytes) For item, separate-key, embedded-separate-key indexes, the maximum length is 1280 bytes by default</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Maximum size of an index</td>
<td>523,792 blocks</td>
<td>For non-extent indexes</td>
</tr>
<tr>
<td></td>
<td></td>
<td>134,090,752 blocks</td>
<td>For extent indexes</td>
</tr>
<tr>
<td>Links</td>
<td>Maximum number of nested links</td>
<td>16 links</td>
<td></td>
</tr>
<tr>
<td>Memory</td>
<td>Number of pages initially assigned to user heap</td>
<td>8 pages</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Maximum amount of memory that can be devoted to SVM regions located outside of the program module itself</td>
<td>Without modifying high_water_mark: 939,524,096 bytes (equal to 229,376 pages, 896 MB, or 1 GB - 128 MB) Maximizing high_water_mark: 1,879,048,192 bytes (equal to 458,752 pages, 1792 MB, or 2 GB - 256 MB) The address starts at 08000000x and stops at high_water_mark (or at 40000000x, if high_water_mark is not specified)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Kernel virtual memory</td>
<td>2 GB</td>
<td></td>
</tr>
<tr>
<td>Network</td>
<td>Maximum number of modules per system (connected by subrings)</td>
<td>32 modules</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Maximum number of systems per cluster (connected by backbone rings)</td>
<td>32 systems</td>
<td></td>
</tr>
</tbody>
</table>
Table B-1. General OpenVOS Software Limits (Continued)

<table>
<thead>
<tr>
<th>Subject</th>
<th>Description</th>
<th>Limit</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Path Names</td>
<td>Maximum length</td>
<td>256 characters</td>
<td></td>
</tr>
<tr>
<td>Pipe Files</td>
<td>Maximum size</td>
<td>17 blocks</td>
<td></td>
</tr>
<tr>
<td>Ports</td>
<td>Maximum number that can be attached to a process</td>
<td>4096 ports</td>
<td></td>
</tr>
<tr>
<td>Preprocessor Symbols</td>
<td>Maximum length of preprocessor symbol</td>
<td>256 characters</td>
<td>For more information, see the description of the registration_admin command in <em>OpenVOS System Administration: Registration and Security</em> (R283)</td>
</tr>
<tr>
<td>Processes</td>
<td>Maximum number of OpenVOS processes (and subprocesses) on a module</td>
<td>4,095 processes</td>
<td></td>
</tr>
<tr>
<td>Process Scheduler</td>
<td>Interrupt time</td>
<td>100 milliseconds</td>
<td>Depends on CPU type</td>
</tr>
<tr>
<td></td>
<td>Time slice range</td>
<td>2 to 4000 milliseconds</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Dead-CPU time-out value</td>
<td>1 minute</td>
<td>Determines the maximum amount of real time for which a CPU may process instructions without rescheduling. OpenVOS uses the dead-CPU time-out value to detect a dead (or nonresponsive) CPU, as the name indicates.</td>
</tr>
<tr>
<td>Program Modules</td>
<td>Maximum size</td>
<td>128 MB</td>
<td></td>
</tr>
<tr>
<td>Queues</td>
<td>Maximum message length (direct queue)</td>
<td>3,072 bytes</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Maximum message length (virtual queue)</td>
<td>32,767 bytes</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Maximum message length for queue on same module as calling process (server and message queues)</td>
<td>$2^{23}$ bytes</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Maximum message length for queue on different module than calling process</td>
<td>$2^{20}$ bytes</td>
<td></td>
</tr>
</tbody>
</table>
### Table B-1. General OpenVOS Software Limits (Continued)

<table>
<thead>
<tr>
<th>Subject</th>
<th>Description</th>
<th>Limit</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Records</td>
<td>Maximum size of records in stream and sequential files</td>
<td>32,767 bytes</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Range of length values in deleted sequential records</td>
<td>-2 to -16,386</td>
<td></td>
</tr>
<tr>
<td>Star Names</td>
<td>Maximum number of components</td>
<td>17</td>
<td></td>
</tr>
<tr>
<td>Stack Frames</td>
<td>Maximum number of bytes available for a function’s initial stack frame</td>
<td>2,147,483,584 bytes</td>
<td></td>
</tr>
<tr>
<td>System Data</td>
<td>Maximum size</td>
<td>2 MB</td>
<td>Ends at 7FE6000x</td>
</tr>
<tr>
<td>User Heap</td>
<td>Minimum and maximum sizes</td>
<td>Minimum: 32,768 bytes</td>
<td>Starts at high_water_mark</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Maximum if high_water_mark is not specified: 896 MB</td>
<td>(or at 40000000x, if high_water_mark is not specified) and can extend to 7FE60000x</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Maximum if high_water_mark is set to 08000000: 1918 MB</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Maximum if high_water_mark is set to 78000000: 126 MB</td>
<td></td>
</tr>
<tr>
<td>User Stack</td>
<td>Minimum and maximum sizes</td>
<td>Minimum: 32,768 bytes</td>
<td>Starts at 7FE60000xx and</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Maximum: 1022 MB</td>
<td>cannot be extended lower than</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>400000000x; the default size is</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>64 MB</td>
</tr>
<tr>
<td>Virtual Circuit Facility</td>
<td>Maximum number of address extensions</td>
<td>255</td>
<td></td>
</tr>
</tbody>
</table>

Table B-2 contains a list of OpenVOS numerical definitions.
### Table B-2. OpenVOS Numerical Definitions

<table>
<thead>
<tr>
<th>Subject</th>
<th>Description</th>
<th>Definition</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Blocks</td>
<td>The size of a block</td>
<td>4096 bytes</td>
<td></td>
</tr>
<tr>
<td>Extents</td>
<td>Size, in blocks</td>
<td>A multiple of 8</td>
<td>For statically allocated extents (SAE) files</td>
</tr>
<tr>
<td></td>
<td></td>
<td>A power of 2 between 8 and 256</td>
<td>For dynamically allocated extents (DAE) files</td>
</tr>
<tr>
<td>File Record Length (with shared virtual memory)</td>
<td>Record length</td>
<td>4096 bytes</td>
<td></td>
</tr>
<tr>
<td>Time Measurement</td>
<td>Micro-jiffies</td>
<td>1/65,536 jiffy (a jiffy is 1/65,536th of a second) or 232.83 picoseconds</td>
<td></td>
</tr>
</tbody>
</table>
Appendix C:
Reducing Program Module Size When Using Shared Virtual Memory Databases

When applications need high-speed access to a shared database, it is often useful to create that database in virtual memory rather than on disk. Applications with this requirement often have several processes updating a database and many more functioning in an inquiry mode. Such shared virtual memory databases are often tens or hundreds of megabytes in size.

In the past, the standard method for creating a shared virtual memory database was to create a data object module with the create_data_object command. This command allows the shared region defined by the object module to occupy the same sequence of virtual memory addresses in each process. The object module created by this command is bound in at the same position (usually first) in each database application.

Unfortunately, the data object module is not merely a placeholder for shared virtual memory space. The data object module and the program modules in which the data object module is bound both use unneeded disk space. For example, if you have a dozen applications that share a 20 MB memory-resident database, the data object module wastes 240 MB of disk space.

An alternative to the create_data_object command is available; it eliminates the use of disk space by program modules with shared virtual memory databases. This alternative uses two bind directives (define and high_water_mark) to store the address and size of the virtual memory database used by a set of applications. However, this method has two minor disadvantages. You must manually determine the values for these bind directives. In addition, these manually-determined values may become invalid when you make small changes to any application in the set of applications that use the shared virtual memory database.

This appendix discusses the following topics:

- “Using create_data_object to Organize Virtual Memory”
- “Using Bind Directives to Organize Virtual Memory”
- “Using Bind Directives to Create a Shared Virtual Memory Database”
Using **create_data_object** to Organize Virtual Memory

This section describes the organization of user virtual memory on ftServer modules when you create a shared virtual memory database with the **create_data_object** command.

**Figure C-1** shows the organization of user virtual memory on an ftServer module. The OpenVOS program loader puts the program module, which includes the code and static data regions, and the symbol table and maps in the lower user memory locations. Also included as part of the program module is a virtual memory space created with the **create_data_object** command. Reserved pages for this space are located between the lowest address and the start of the compiled program.

The heap and stack are **not** contained in the program module and do not have copies stored on the disk. The OpenVOS program module loader defines the initial stack and heap space requirements for a program module as the program module begins to execute.

On an ftServer module, the heap begins at an address defined by a default value (0x40000000) or by the **high_water_mark** bind directive, and it grows toward higher virtual memory addresses. The stack occupies higher addresses in the address space and grows downward toward the heap.
This section describes how to use the `define` and `high_water_mark` bind directives to organize user virtual memory on ftServer modules. The procedure for moving a shared virtual memory database out of a program module is described in “Using Bind Directives to Create a Shared Virtual Memory Database.”

Figure C-2 shows the organization of user program virtual memory on an ftServer module when you use the `define` and `high_water_mark` bind directives, and the shared virtual memory database follows the program module and is contiguous with the heap. When you move a shared virtual memory database to this location, the `bind` command no longer allocates disk space for the shared virtual memory database.

**Note:** The unused pages for alignment that precede the shared virtual memory database are explained in “Using Bind Directives to Create a Shared Virtual Memory Database.”
Moving a shared virtual memory database to this location also has no effect on the size of the database and does not require additional paging space on the disk.

Figure C-2. Shared Virtual Memory Database Not in a Program Module

Using Bind Directives to Create a Shared Virtual Memory Database

This section discusses the following topics:

- “Making and Checking the Calculations”
- “Reacting to Application and Compiler Changes”
- “Using More Than One Shared Virtual Memory Region”
- “Example of Binder Control File Changes”

To create a shared virtual memory database, you must specify the address of the shared virtual memory region (SVMR) and the beginning of the heap or stack (high_water_mark) in the

C-4  Appendix C: Reducing Program Module Size When Using Shared Virtual Memory Databases
binder control file, and modify your application to access these values. All program modules sharing a region of virtual memory must map the same file records into the same range of virtual memory pages.

When you specify high_water_mark, the following rules apply:

- SVMRs must always have the same addresses, regardless of whether or not you provide these addresses in the call to $\texttt{s$connect\_vm\_region3}$.

- Programs that use statically defined SVMRs can define the address of the regions anywhere above the program module and below the heap. Therefore, for such programs, you may need to adjust high_water_mark for large or numerous regions.

- To avoid problems, make sure that programs sharing a statically-defined SVMR have the same high_water_mark values.

- All programs sharing a dynamic SVMR must have the same high_water_mark values so that the given address is available in all of these programs.

- If a program uses both statically defined and dynamically allocated SVMRs, the high_water_mark value must be the same for all involved programs.

- The high_water_mark value for each program must be large enough to accommodate the SVMR used by that particular program.

Use the following procedure to move a shared virtual memory database.

1. For each application (program module) that uses the same space-reserving object modules generated by create_data_object, issue the display_program_module command with the name of the currently used program module and the -header and -module_map arguments. Write down the name of the program module and the values of \texttt{n\_vm\_pages} (a decimal value) and \texttt{user\_boundary} (a hexadecimal value), which are listed in the program module header. Also, using the object module map, write down the code length (a hexadecimal value) of object modules created with the create_data_object command. For more information on the create_data_object and display_program_module commands, see the command descriptions in this manual.

   Note: The value of high_water_mark shown in the program module header should be zero.

2. To determine the size of the largest program module, multiply the largest \texttt{n\_vm\_pages} value by 4096 bytes per page, convert the value to hexadecimal, and subtract from it the size of the create_data_object object module(s).

3. To determine the SVMR value, add the size of the largest program module (from Step 2) to the user_boundary value.

   Stratus recommends that you round up the initial value of SVMR by several pages so that small source code or compiler changes will not force you to completely recalculate the shared virtual memory values in the binder control files. For more information
about how application and compiler changes can affect this value, see “Reacting to Application and Compiler Changes.”

Notes:

1. No tool exists that automatically establishes the value of SVMR for multiple program modules, and takes account of the fact that each program module probably has a different amount of storage allocated to the code and static data. You must determine the SVMR value by using the display_program_module command.

2. If a program module is not the largest program module in the set, some virtual memory space in this program module is used to align the shared virtual memory used by a set of applications. This space is equal to the difference in size of this particular program module and the largest one in the set sharing the virtual memory. In Figure C-2, this space is called “Unused Pages for Alignment.”

4. In each affected application’s binder control file, remove the module directives that reference object modules created with the create_data_object command.

Use the define bind directive to specify a symbol and an address for the SVMR. Use the object_name that you specified with the create_data_object command as the name of the define symbol. Your application’s use of this symbol is identical to its use of the data object module(s) created with the create_data_object command.

To ensure that SVMRs do not extend into the area reserved for the heap, compare the high_water_mark value to the highest shared virtual memory address plus the size of the data object module(s) that you created with the create_data_object command.

5. Rebind each application that has a changed binder control file.

Making and Checking the Calculations

You can perform these calculations using hexadecimal or decimal numbers. The advantage of using hexadecimal numbers is that all multiples of the 4096-byte page size end in 000x.

Note: All hexadecimal addresses require a 0 as the first digit. The binder fails if the first digit is A through F.

Always check your calculations. OpenVOS does not verify the value that you specify for the SVMR overlays the heap or the stack. If a shared virtual memory file accidentally overlays the heap or stack, the results of continued computation cannot be predicted.

Reacting to Application and Compiler Changes

The values associated with this procedure may change over the life-cycle of the applications. Changes in program module size can occur any time any part of an application is recompiled and rebound. Even simple maintenance actions, such as changing the name of an error code, can affect the page boundary values. New compiler releases may generate different code
sequences for the same construct, even when the same compiler options are employed. For this mechanism to work, all programmers for the affected applications must check that the shared virtual memory values used in the bind control files reflect the current size of the application program modules. Stratus recommends that you round up the initial value of SVMR by several pages so that small source code or compiler changes will not force you to completely recalculate the shared virtual memory values in the binder control files.

**Using More Than One Shared Virtual Memory Region**

If your applications share more than one shared virtual memory region, you can define several instances of SVMR such as SVMR1, SVMR2, and SVMR3. Regardless of how many regions you define, each binder control file can contain only one `high_water_mark` statement.

**Example of Binder Control File Changes**

The following example shows binder control files that use the same data object created with the `create_data_object` command. It then shows how these binder control files are modified to add the values for SVMR.

**Binder Control Files Using a Created Data Object**

The following is a binder control file for a C program that uses a `SHARED_VM_REGION` data object created with the `create_data_object` command.

```plaintext
name: c_example;
size: 64mb;
modules: SHARED_VM_REGION page_aligned,
         c_example;
end;
```

The following is a binder control file for a COBOL program that uses a `SHARED_VM_REGION` data object created with the `create_data_object` command.

```plaintext
name: cobol_example;
size: 64mb;
modules: SHARED_VM_REGION page_aligned,
         cobol_example;
end;
```

The following is a binder control file for a PL/I program that uses a `SHARED_VM_REGION` data object created with the `create_data_object` command.

```plaintext
name: pl1_example;
size: 64mb;
modules: SHARED_VM_REGION page_aligned,
         pl1_example;
end;
```
### Changing Binder Control Files to Use SVMR

Before adding the SVMR values to the binder control files of applications that use the same shared virtual memory database, calculate the SVMR values. In this example, the `display_program_module` command is run against the C, COBOL, and PL/I program modules, and the following values are recorded.

<table>
<thead>
<tr>
<th></th>
<th>C</th>
<th>COBOL</th>
<th>PL/I</th>
</tr>
</thead>
<tbody>
<tr>
<td>n_vm_pages</td>
<td>4608</td>
<td>5120</td>
<td>4976</td>
</tr>
<tr>
<td>user_boundary</td>
<td>00008000x</td>
<td>00008000x</td>
<td>00008000x</td>
</tr>
<tr>
<td>SHARED_VM_REGION code size</td>
<td>1000000x</td>
<td>1000000x</td>
<td>1000000x</td>
</tr>
</tbody>
</table>

To determine the size of the SVMR, use the `n_vm_pages` value from the COBOL application, since it is the largest `n_vm_pages` value. Multiply this value by 4096, convert it to hexadecimal, from it subtract the size of the `SHARED_VM_REGION` code pages, and then add the `user_boundary` value.

\[
\text{SVMR} = ((\text{hexadecimal } 5120 \times 4096) - 1000000x + 8000x)
\]

\[
\text{SVMR} = 1400000x - 1000000x + 8000x
\]

\[
\text{SVMR} = 408000x
\]

To determine whether you need to adjust the `high_water_mark` value, you must calculate the ending address of the SVMR by adding the SVMR to the `SHARED_VM_REGION` code size (in this case, `408000x + 1000000x`). Because `1408000x` is lower than the default `high_water_mark` value (`4000000x`), you do not need to adjust the `high_water_mark` value.

**Note:** For processes with multiple SVMRs, calculate the ending address of the SVMR with the highest starting address.

The following is a binder control file for a C program that uses a data object created with the `define` directive.

```c
name: c_example;
size: 64mb;
define: SHARED_VM_REGION address (408000x);
modules: c_example;
end;
```

The following is a binder control file for a COBOL program that uses a data object created with the `define` directive.

```cobol
name: cobol_example;
size: 64mb;
define: SHARED_VM_REGION address (408000x);
modules: cobol_example;
end;
```
The following is a binder control file for a PL/I program that uses a data object created with the `define` directive.

```pli
name: pl1_example;
size: 64mb;
define: SHARED_VM_REGION address (408000x);
modules: pl1_example;
end;
```

**Using `high_water_mark` to Adjust the Size Reserved for the SVMR**

You can use the `high_water_mark` bind directive to increase or decrease the size reserved for the SVMR, but, typically, you would use it only in the following circumstances:

- To connect very large or very numerous SVMRs. In this case, you would increase `high_water_mark`.
- Under conditions that require maximum stack and heap space. In this case, you would decrease `high_water_mark`.

By default, the `high_water_mark` value is 1 GB (40000000x). This value must be greater than 128 MB (8000000x) and less than 2 GB - 128 MB (78000000x). If you specify a value that is less than 128 MB, OpenVOS silently (without any messages or errors) sets `high_water_mark` to 128 MB. Likewise, if you specify a value that is greater than 2 GB - 128 MB, OpenVOS silently sets `high_water_mark` to 2 GB - 128 MB.

If you set `high_water_mark` to 128 MB, the program cannot use dynamic SVMRs.

If a program uses both statically defined and dynamically allocated SVMRs, the static regions should either remain below 128 MB or grow from 128 MB toward higher addresses.

Typically, all programs that plan to use a common set of VM regions should have the same `high_water_mark` value. Otherwise, there is no guarantee of correct program behavior.

The following example illustrates how to adjust the `high_water_mark` value. It allows approximately 1.5 GB of reserved space for SVMRs and places a SVMR at the 128 MB address.

```pli
name: hwm_example;
define: SHARED_VM_REGION address (8000000x);
high_water_mark: 60000000x;
modules: hwm_example;
end;
```
OpenVOS Commands Reference Manual (R098)

Index

Miscellaneous
#pragma options and the cc command, 1-121
#replace statement, debugger, 1-243
64-bit stream files, 1-212

A
-A compiler option, 1-113
Abbreviations, 1-497, 1-812
debugger requests, 1-243
expanding, 1-349, 1-565, 1-759, 1-812
file, 1-812
first token, 1-242, 1-565
limits, B-1
local, 1-466
preventing expansion of, 1-813
(abs) command function, 1-4
Access
copying, 1-634
default, 1-307, 1-444
to devices, 1-36, 1-197, 1-297, 1-399, 1-441,
1-551, 1-639, 1-846
to directories, 1-297, 1-307, 1-440
to files, 1-297, 1-440, 1-474
to objects, 1-297
to tapes, 1-541, 1-545, 1-718
Access codes, 1-439, 1-444
execute, 1-440, 1-444
for directories, 1-5, 1-42
for files, 1-5, 1-42
modify, 1-440
null, 1-440, 1-441, 1-444
read, 1-440, 1-441, 1-444
status, 1-440
undefined, 1-307, 1-445
write, 1-440, 1-441, 1-444
(access) command function, 1-5
Access control, 1-307, 1-440, 1-441, 1-634
displaying access, 1-296
give_access command, 1-439
give_default_access
command, 1-444
giving access, 1-439, 1-444
remove_access command, 1-642
remove_default_access command, 1-644
removing access, 1-642, 1-644
Access control lists (ACLs), 1-296
comparing, 1-158
default, 1-307, 1-444
displaying, 1-299, 1-307
entries, 1-634
new directories, 1-205
Access requirements
for binding, 1-64
for compilation, 1-99, 1-131, 1-156, 1-436,
1-587, 1-599
for preprocessing, 1-610
to a form definition file, 1-410, 1-575
Actual tape parameters, 1-389, A-1
add_entry_names command, 1-2
add_library_path command, 1-6
add_paging_file command, 1-211, 1-323
add_profile command, 1-10
Address space
maximum size, B-1
(after) command function, 1-7
Aliases, 1-535
-align_mod16 binder argument, 1-46
Alignment, 1-46
faults, 1-628
mapping rules, 1-86, 1-92
profiling faults, 1-48
Allocating file space, 1-679
analyze_pc_samples command, 1-15
bind requirements, 1-22, 1-29
boot partition number, 1-16
CPU usage report, 1-26
eamples of using, 1-20
measuring standard deviation, 1-31
PC sample statistics report, 1-25
sample output, 1-21
use with harvest_pc_samples
command, 1-450
uses raw data files, 1-16
when to use, 1-450
analyze_system  
dump_eit request, 1-18
ANSI C conformance and the cc  
command, 1-124
ANSI labels, 1-719
ANSI rules  
checking OpenVOS C programs  
conformance to, 1-90, 1-101
checking Pascal programs conformance  
to, 1-580
-ansi_replace compiler argument, 1-151
-ansi_rules compiler argument, 1-90, 1-101
Apostrophes in command lines, 1-1, 1-13
args machine-mode request, 1-268
args source-mode request, 1-250
Arithmetic expressions in command lines, 1-13
Arithmetic, floating point, 1-14
ascii character set, 1-177, 1-210, 1-731
ASCII tape translation, 1-236, 1-541, 1-718
(ask) command function, 1-8
Assembly  

code  
displaying program module  
pseudo, 1-370
language  
listings, 1-88, 1-120, 1-147, 1-150, 1-326, 1-580, 1-592
Assigning a device to a process, 1-649
attach_default_output command, 1-33
attach_port command, 1-35
Attributes  
of directories  
comparing, 1-158
of files, 1-337, 1-731
comparing, 1-164
setting, 1-173, 1-177
pipe files, 1-697
Authors of files  
comparing, 1-158, 1-162
Automatic mounting, 1-500, 1-501, 1-544
B  
Backing up objects  
list_save_tape command, 1-500
restore_object command, 1-653
save_object command, 1-658
verify_save command, 1-829
-backup compiler argument, 1-407
Backup files, 1-403, 1-407, 1-412, 1-463
-basic compiler argument, 1-407, 1-572
batch command, 1-37
Batch control files, 1-39, 1-790
Batch processing, 1-37, 1-39, 1-106, 1-301
canceling a batch request, 1-106
canceling a device reservation, 1-108
entering a batch request, 1-37
identifying an active process, 1-478
listing a batch request, 1-477
modifying a batch request, 1-788
moving a device reservation, 1-550
reserving a device, 1-649
Batch queues, 1-38, 1-106, 1-301, 1-789
listing, 1-477
.batch suffix, 1-40
(before) command function, 1-10
BINARY tape translation, 1-236, 1-541
bind command, 1-42
use by analyze_pc_samples  
command, 1-22
Bind maps, 1-47, 1-56
.bind suffix, 1-64
Binder  
case sensitivity, 1-54
controlling directory searches, 1-54
displaying statistics from, 1-57
naming of program modules, 1-55
resolution of values in, 1-52
syntax of numerical values, 1-53
Binder arguments  
-align_mod16, 1-46
-define_main, 1-46
-load_in_kernel, 1-47
-load_point, 1-44, 1-59
-map, 1-47
-max_heap_size, 1-44, 1-60
-max_program_size, 1-45
-max_stack_size, 1-45
-no_dynamic_tasking, 1-47
-number_of_tasks, 1-47
.pm_name, 1-45
-private_heap, 1-48
-private_stack, 1-48
-processor, 1-45, 1-56
-profile_alignment_faults, 1-48
-references_kernel, 1-48
-relocatable, 1-49
,retain_all, 1-49
-size, C-5
-stack_fence_size, 1-45, 1-60
-stack_size, 1-46
-statistics, 1-49
-subroutines_are_functions, 1-49
-table, 1-49
-target_module, 1-46
-version, 1-46

Index-2
Index

Binder control files, 1-43, 1-64
- comments, 1-65
- define directive, 1-65
- delimiters, 1-65
- end directive, 1-65
- entry directive, 1-65
- high_water_mark directive, 1-66
- load_point directive, 1-66
- max_heap_size directive, 1-66
- max_program_size directive, 1-66
- max_stack_size directive, 1-66
- modules directive, 1-66
- name directive, 1-67
- number_of_tasks directive, 1-67
- options directive, 1-67, 1-70
- preprocessing of, 1-55
- processor directive, 1-70
- region_load_point directive, 1-70
- retain directive, 1-71
- search directive, 1-71
- section directive, 1-72
- size directive, 1-72
- stack_fence_size directive, 1-73
- stack_size directive, 1-73
- synonym directive, 1-73
- syntax, 1-64
- variable_arg_count directive, 1-73
- variables directive, 1-73
- visibility directive, 1-75

Binder preprocessing, 1-44

Binding, 1-42
- access requirements, 1-64
- bind maps, 1-47
- for harvest_pc_samples command, 1-450
- for profile command, 1-626
- shared virtual memory databases, 1-200
- statistics, 1-49

Block environments in the debugger, 1-246

Blocks
- size, B-7

Boot partition number
- specifying in analyze_pc_samples command, 1-16

(break) command function, 1-11

Break level, 1-77

break machine-mode request, 1-267, 1-268
break source-mode request, 1-250
break_process command, 1-77

Breakpoints, 1-244, 1-250, 1-268, 1-565

Broadcasting messages, 1-351

Buffers, 1-401
- read only, 1-401

bundle command, 1-79

(byte) command function, 1-12

C

c command, 1-84
- -c compiler argument, 1-408, 1-572
- c debugger mode, 1-249, 1-253
- .c suffix, 1-91, 1-121
- c_preprocess command, 1-100

Cache
- using all physical memory, 1-701

(calc) command function, 1-14

Calculating an expression, 1-14

call source-mode request, 1-251

call_thru command, 1-103

Calling process, 1-759

cancel_batch_requests command, 1-106

cancel_device_reservation command, 1-108

cancel_print_requests command, 1-109

Canceling
- batch requests, 1-106
- device reservations, 1-108
- print requests, 1-109

Canonical control sequences, 1-618, 1-796

c command, 1-111

(ceil) command function, 1-16

change_current_dir command, 1-132, 1-134

change_password command, 1-134

Changing
- passwords, 1-536
- text file attributes, 1-177, 1-731
- working directories, 1-132, 1-134

Character sets, 1-161
- ascii, 1-177, 1-210, 1-731
- chinese1, 1-177, 1-210, 1-731
- chinese2, 1-177, 1-210, 1-731
- hangul, 1-177, 1-210, 1-731
- kanji, 1-177, 1-210, 1-731
- katakana, 1-177, 1-210, 1-731
- latin_1, 1-177, 1-210, 1-731
- latin_9, 1-177, 1-210, 1-731
- national_language_support, 1-413
- shift modes, 1-210, 1-414
- simplified_chinese, 1-177, 1-210, 1-731
- user_dbcs, 1-177, 1-210, 1-731

Character strings, 1-2

Characters, flow-control, 1-415
Index

- check compiler argument, 1-579, 1-591, 1-117, 1-149, 1-428
- check_conformance compiler argument, 1-580
- check_overflow compiler argument, 1-580
- check_uninitialized compiler argument, 1-581, 1-593
- check_ansi compiler argument, 1-90, 1-101
- check Enumeration compiler argument, 1-91, 1-116
- check_incompatible compiler argument, 1-117
check_posix command, 1-137
and optimization level, 1-581
optimization level, 1-593
optimization level affects, 1-430
pascal command, 1-585
performed at specific optimization levels, 1-149
pl1 command, 1-598
Checking for conformance to ANSI rules, 1-580
Checking for POSIX compliance, 1-137
chinese1 character set, 1-177
chinese2 character set, 1-177
Circular links, 1-466
clear machine-mode request, 1-268
clear source-mode request, 1-251, 1-252, 1-256, 1-260
clone_dir command, 1-139
clone_file command, 1-142
COBOL
    compilation levels, 1-147
data types, 1-747
co BOL command, 1-144
- cobol compiler argument, 1-408, 1-572
cobol debugger mode, 1-249
.co BOL suffix, 1-151
Code region
definition of, 1-807
Codes
status, 1-329
Collation sequences, 1-230, 1-742
Command functions, 1-1
(abs), 1-4
(access), 1-5
(after), 1-7
(ask), 1-8
(before), 1-10
(break), 1-11
(byte), 1-12
(calc), 1-14
(ceil), 1-16
(command_status), 1-17
(concat), 1-18
(contents), 1-19
(copy), 1-20
(count), 1-21
(current_dir), 1-22
(current_module), 1-23
(date), 1-24
(date_time), 1-28
(decimal), 1-29
(directory_name), 1-30
(end_of_file), 1-31
(exists), 1-32
(extended_names), 1-34
(extended_names_version), 1-35
(file_info), 1-37
(floor), 1-39
(given), 1-40
(group_name), 1-41
(has_access), 1-42
(hexadecimal), 1-44
(home_dir), 1-45
(index), 1-46
(iso_date), 1-47
(iso_date_time), 1-50
(language_name), 1-53
(length), 1-54
limits, B-1
(lock_type), 1-55
(locked), 1-56
(ltrim), 1-57
(master_disk), 1-58
(max), 1-59
(message), 1-60
(min), 1-62
(mod), 1-63
(module_info), 1-64
(module_name), 1-67
(name_string), 1-68
(object_name), 1-69
(online), 1-70
(path_name), 1-71
(person_name), 1-72
(posix_path), 1-73
(process_dir), 1-75
(process_info), 1-77
(process_type), 1-78
(quote), 1-79
(rank), 1-80
(referencing_dir), 1-81
(reverse), 1-82
(rtrim), 1-83
(search), 1-84
Index

Command libraries, 1-7, 1-284, 1-486, 1-687, 1-839
adding entries, 1-7
defining path names, 1-687
deleting path names, 1-285
listing path names, 1-486
Command limits
initial, 1-498
maximum, 1-498
module default, 1-498
Command lines
length limits, B-2
Command macros, 1-839
(command_status) command
function, 1-17
Command types, 1-456
Commands
external, 1-839
internal, 1-565, 1-839
Communication links, 1-104
list_gateways command, 1-484
compare_dirs command, 1-157
compare_files command, 1-160
use with links, 1-164
Comparing
directory attributes, 1-158
file attributes, 1-164
logical records, 1-159
Compilation
access requirements, 1-99, 1-131, 1-156, 1-436, 1-587, 1-599
debugging and, 1-245
disk requirements, 1-92, 1-121, 1-151, 1-430, 1-581, 1-594
errors, 1-98, 1-130, 1-155, 1-435, 1-599
information, 1-352
memory requirements, 1-92, 1-121, 1-151, 1-430, 1-581, 1-594
Compilation levels of COBOL, 1-147
assembly language, 1-580
cross reference, 1-578
nesting levels, 1-581
Compiler arguments
-A, 1-113
-ansi_replace, 1-151
-ansi_rules, 1-90, 1-101
-backup, 1-407
-basic, 1-407
-c, 1-408
-check, 1-117, 1-149, 1-428, 1-579, 1-591
-check_ansi, 1-90, 1-101
-check_conformance, 1-580
-check_enumeration, 1-91, 1-116
-check_incompatible, 1-117
-check_overflow, 1-580
-cobol, 1-408
-compress, 1-119
-control, 1-43
-cpu_profile, 1-88, 1-150, 1-429, 1-580, 1-592
-D, 1-113, 1-817
-debugging_mode, 1-150
-default_char, 1-89, 1-116
-default_class, 1-150
-default_comp, 1-150
-default_sign, 1-145
-definition_files, 1-100
-E, 1-113
-edit, 1-407
-fixedoverflow, 1-89, 1-118, 1-429, 1-592
-force_write, 1-407
-fortran, 1-408
-fortran66, 1-429
-full, 1-88, 1-95, 1-120, 1-147, 1-429, 1-580, 1-592
-g, 1-113, 1-122, 1-129, 1-817
-g. See also -production_table
-I, 1-113, 1-817
-include, 1-114
-include_files, 1-89
- into, 1-406
- language, 1-147
- library, 1-407
- linted, 1-90, 1-97
- list, 1-86, 1-95, 1-101, 1-119, 1-147,
  1-427, 1-578, 1-590
- M, 1-113
- main, 1-150
- mapcase, 1-88, 1-98, 1-117, 1-129,
  1-149, 1-155, 1-428, 1-579, 1-591
- mapping_rules, 1-86, 1-92, 1-115,
  1-126, 1-146, 1-152, 1-409, 1-427,
  1-578, 1-590
- nesting, 1-89, 1-95, 1-118, 1-581, 1-592
- O, 1-113, 1-122, 1-817
- o, 1-113, 1-817
- optimization_level, 1-90, 1-96,
  1-149, 1-154, 1-430, 1-581, 1-593
- optimize, 1-87, 1-149, 1-428, 1-579,
  1-591
- P, 1-113
- pascal, 1-408
- pl1, 1-408
- pl1_template, 1-408
- prefix, 1-407
- processor, 1-86, 1-92, 1-101, 1-115,
  1-125, 1-146, 1-152, 1-409, 1-427,
  1-431, 1-578, 1-582, 1-590, 1-595
- produce_symtab, 1-409
- production_table, 1-87, 1-97,
  1-148, 1-153, 1-428, 1-579, 1-584,
  1-591
- profile, 1-88, 1-149, 1-428, 1-579,
  1-591
- q, 1-113
- recursive, 1-430
- registers, 1-90
- search, 1-44
- segmentation, 1-151
- short_integer, 1-430
- short_logical, 1-430
- show_include, 1-120
- show_macros, 1-89, 1-120
- silent, 1-88, 1-101, 1-150, 1-429,
  1-580, 1-592
- sort_into_by_alignment, 1-409
- statistics, 1-88, 1-101, 1-119, 1-150,
  1-429, 1-580, 1-592
- store_args, 1-90, 1-120, 1-593
- suppress_diag, 1-86, 1-114
- system_programming, 1-89, 1-118,
  1-581, 1-593
- table, 1-87, 1-97, 1-118, 1-122, 1-129,
  1-148, 1-153, 1-428, 1-578, 1-584,
  1-591
- truncate_to, 1-117, 1-129
- type_checking, 1-91, 1-93, 1-116,
  1-127
- U, 1-113, 1-818
- u, 1-114
- w, 1-114, 1-818
- w, 1-114
- X, 1-114, 1-124
- xref, 1-87, 1-95, 1-119, 1-147, 1-427,
  1-578, 1-590
- xref_format, 1-148

Compiler diagnostics
  severity 0, 1-98, 1-130, 1-155, 1-435, 1-586,
  1-599
  severity 1, 1-98, 1-130, 1-155, 1-435, 1-586,
  1-599
  severity 2, 1-98, 1-130, 1-155, 1-435, 1-586,
  1-599
  severity 3, 1-98, 1-130, 1-155, 1-435, 1-586,
  1-599
  severity 4, 1-98, 1-130, 1-155, 1-435, 1-586,
  1-599

Compiler optimizations
  checking uninitialized variables, 1-155

Compiler statistics, 1-88, 1-119, 1-150, 1-429,
  1-580, 1-592

Compilers, 1-3
  C, 1-84
  COBOL, 1-144
  errors, 1-155, 1-586
  FORTRAN, 1-425
  OpenVOS Standard C, 1-111
  Pascal, 1-576
  PL/I, 1-588

See also bind command, debug command,
  Optimization levels, Program
  modules
  -compress compiler argument, 1-119

Computational types, 1-150
  (concat) command function, 1-18
  Connecting to a remote host, 1-103
  consolidate_dir command, 1-169
  constant debugger-request argument, 1-265
  (contents) command function, 1-19
  continue machine-mode request, 1-268
  continue source-mode request, 1-252
  -control compiler argument, 1-43

Control files
  batch, 1-39, 1-790
  bind, 1-43
binder, 1-64

sort, 1-745

Control sequences

canonical, 1-618, 1-796
generic, 1-618, 1-796

convert_stream_file command, 1-173
convert_text_file command, 1-173, 1-177

Converting
stream files, 1-173
Converting fixed files to stream files, 1-238
(copy) command function, 1-20

copy_dir command, 1-180
removes expiration dates, 1-181
use with links, 1-184

copy_file command, 1-187
removes expiration dates, 1-188
use with links, 1-191

copy_tape command, 1-195

Copying
directories, 1-182
directory access, 1-634
files, 1-190
fixed length records, 1-191
indexed files, 1-192
tapes, 1-195

Copying pipes, 1-182
(count) command function, 1-21
cpio format, 1-714, 1-719
cpiof format, 1-714, 1-719
cpp command, 1-198

CPU time, 1-10, 1-665, 1-758
CPU usage, 1-665

-cpu_profile compiler argument, 1-88, 1-150, 1-429, 1-580, 1-592

create_data_object command, 1-200, C-1

and binder, 1-200
shared virtual memory databases, 1-201

create_deleted_record_index command, 1-202

create_dir command, 1-205
create_file command, 1-207
create_index command, 1-226
create_record_index command, 1-233
create_tape_volumes command, 1-235

Creating
directories, 1-205
files, 1-207
form letters, 1-770
links, 1-2, 1-465
links between directories, 1-468
processes, 1-534
subprocesses, 1-537

Creating a main function, 1-46
Creation date of files, 1-181, 1-188
Cross-reference listings, 1-87, 1-119, 1-147, 1-427, 1-578, 1-590

Current
block environment, 1-245, 1-246
date, 1-24, 1-47
directory, 1-22, 1-132
environment, 1-245, 1-246
line, 1-246, 1-247
module, 1-23, 1-304
statement, 1-247
task, 1-247
time, 1-27, 1-50

(current_dir) command function, 1-22

(current_module) command function, 1-23

Cursor format, 1-726
cvt_fixed_to_stream command, 1-238
cvt_stream_to_fixed command, 1-239

D

-D compiler option, 1-113, 1-817

DAE files. See Dynamically-allocated extents (DAE) files

DAE indexes. See Dynamically-allocated extents (DAE) indexes

Data alignment, 1-126, 1-153, 1-583, 1-595
Data compression, 1-542, 1-659, 1-710
Data links, external, 1-4
Data names, external, 1-3
Data types, 1-747

Date and time
displaying, 1-305
expiration, 1-181, 1-188, 1-675

set_language command affects format of, 1-684

(date) command function, 1-24
(date_time) command function, 1-28

-dd suffix, 1-776, 1-742

debug command, 1-241

- production_table compiler argument affects, 1-87, 1-148, 1-579, 1-591

using command macros, 1-243
using other commands, 1-243

Debug process sets, 1-564

Debugger modes, 1-250
e, 1-249, 1-253
cobol, 1-249
fortran, 1-250
machine, 1-261
pascal, 1-250
pl1, 1-250
Debugger requests, 1-250, 1-267
    args, 1-250, 1-268
    break, 1-250, 1-267, 1-268
    call, 1-251
    clear, 1-251, 1-268
    clearw, 1-252
    continue, 1-252, 1-268
    disassemble, 1-252
    display, 1-252, 1-253
    dump, 1-253
    env, 1-254
    help, 1-255
    keep, 1-255
    list, 1-256
    listw, 1-256
    position, 1-256
    quit, 1-257
    regs, 1-257
    return, 1-257
    set, 1-258
    source, 1-258
    source-path, 1-259
    start, 1-259
    step, 1-260
    symbol, 1-260
    task_status, 1-260
    trace, 1-260
    watch, 1-260
    where, 1-261

Debugger-request arguments
    constant, 1-265
    endian_specifier, 1-270
    expression, 1-249, 1-250
    -include, 1-256
    label, 1-248
    line, 1-248
    number, 1-248
    relational_expression, 1-265
    request_list, 1-248, 1-251, 1-255
    substring, 1-248
    variable, 1-249

Debugging, 1-241, 1-564
    arguments used for, 1-247
    current block environment, 1-245
    current line, 1-246, 1-247
    current statement, 1-247
    current task, 1-247
    debugger request level, 1-242
    include files, 1-256, 1-258
    internal commands and, 1-243
    internal subroutines, 1-584
    machine mode, 1-261
    object mode, 1-261
    registers and, 1-257, 1-262
    server processes, 1-564
    source mode, 1-245
    without symbol table, 1-261
    -debugging_mode compiler argument, 1-150
    (decimal) command function, 1-29
    decode_vos_file command, 1-272
decrypt command, 1-274
    Default access, 1-307, 1-444
    Default access control lists (DACLs), 1-307, 1-444, 1-644
    entries, 1-634
    new directories, 1-205
    Default batch queues, 1-106
    Default character sets of text files, 1-338
    Default output ports, 1-33, 1-287
    Default print queues, 1-361
    Default tape parameters, 1-388, 1-710, 1-718, A-1
    -default_char compiler argument, 1-89, 1-116
    -default_class compiler argument, 1-150
    -default_comp compiler argument, 1-150
    -default_sign compiler argument, 1-145
    Deferred printing, 1-616, 1-794
    Deferred processing, 1-790
    define compiler directive and the
    debugger, 1-243
    $define preprocessor statement, 1-607
    -define_main binder argument, 1-46
    Definition files, 1-85, 1-100
    field, 1-407
    form, 1-406, 1-569
    -definition_files compiler
    argument, 1-100
    delete_dir command, 1-276
    delete_file command, 1-279
    creating safeguards, 1-280
    delete_index command, 1-282
    delete_library_path command, 1-284
    Deleted records, 1-202
    Deleting
directories, 1-276
    files, 1-279
    expiration date, 1-675
    indexes, 1-282
    links, 1-786
    print requests, 1-109
    user access, 1-642, 1-644
Index

detach_default_output command, 1-287

detach_port command, 1-288

Detaching ports, 1-287, 1-288, 1-538

Device access lists, 1-296

Devices

access, 1-36, 1-197, 1-399, 1-441, 1-551, 1-639, 1-846

displaying path names, 1-481

freeing, 1-108

information, 1-311

printers, 1-364

reserved, 1-550

terminals, 1-392, 1-721

types, 1-482

USB, 1-393

Dictionaries, 1-402, 1-413

user, 1-413

Direct queues, 1-215

Directing output

to a device, 1-752

to a file, 1-33, 1-158, 1-163, 1-292, 1-752

to a printer, 1-333

to a terminal, 1-33, 1-292

Directories, 1-319

access, 1-307, 1-440

access control lists, 1-205

comparing attributes, 1-158

copying, 1-182

copying access, 1-634

creating, 1-205

current, 1-132, 1-303

default open options

displaying, 1-309

setting, 1-667

deleting, 1-276

home, 1-132, 1-536

libraries, 1-284

limits, B-2

linking, 1-468

listing, 1-472

listing contents, 1-472

moving, 1-553

recreating indexes, 1-555

renaming, 1-647

replacing, 1-553

restoring, 1-654

root, 1-519

status, 1-319

Directory management

backing up objects, 1-660

change_current_dir command, 1-132

compare_dirs command, 1-157

copy_dir command, 1-180

create_dir command, 1-205

delete_dir command, 1-276

display_current_dir command, 1-303

display_default_open_options command, 1-309

display_open_options command, 1-358

links, 1-466

list command, 1-473

move_dir command, 1-552

rename command, 1-646

restoring directories, 1-654

save_object command, 1-660

search paths, 1-686, 1-689

set_default_open_options command, 1-667

set_open_options command, 1-691

walk_dir command, 1-836

(directory_name) command

function, 1-30

disassemble source-mode request, 1-252

Disk space

retaining, 1-781

Disks

displaying information, 1-321, 1-322

blocks, 1-327, 1-474

usage, 1-327, 1-473

paging partition, 1-211, 1-323

dismount_tape command, 1-236, 1-289

display command, 1-291

Display forms, display of, 1-726

display source-mode request, 1-252, 1-253

and optimized code, 1-253

and recursive procedures, 1-253

display_access command, 1-296

display_access_list command, 1-299

display_batch_status command, 1-301

display_current_dir command, 1-303

display_current_module command, 1-304

display_date_time command, 1-305

display_default_open_options command, 1-309

display_device_info command, 1-311

display_dir_status command, 1-319

display_disk_info command, 1-321

display_disk_usage command, 1-327
Index

display_error command, 1-329
  displaying default alignment mapping value
  with, 1-86, 1-115, 1-146, 1-427,
  1-578, 1-590
  displaying default processor value
  with, 1-45, 1-86, 1-115, 1-146,
  1-427, 1-578, 1-590
display_file command, 1-331
display_file_status command, 1-337
display_line command, 1-349
display_notices command, 1-351
display_object_module_info
  command, 1-352
display_open_options command, 1-358
display_print_defaults
  command, 1-361
display_print_status command, 1-363
display_program_module
  command, 1-366
display_system_usage command, 1-382
display_tape_params command, 1-388,
  1-393
display_terminal_parameters
  command, 1-392
display_usb_info command, 1-393
Displaying
  date and time, 1-305
  device path names, 1-481
  directory contents, 1-472
  library paths, 1-485
  literal text, 1-349
  modules in a system, 1-487
  port information, 1-489
  print requests, 1-494
  systems, 1-502
  tape volume contents, 1-504
  user information, 1-508
Displaying the end of a file, 1-764, 1-765
dump source-mode request, 1-253
dump_file command, 1-394
dump_record command, 1-396
dump_tape command, 1-397
Dumping
  files, 1-394
  files from tape, 1-397
  indexes, 1-394
  records, 1-396
Dynamic tasking
  definition of, 1-47
Dynamically-allocated extents (DAE)
  files
    creating, 1-209, 1-215
    recovering from failures, 1-218
    indexes, creating, 1-228

E
  -E compiler option, 1-113
  EBCDIC tape translation, 1-236, 1-541, 1-718
  edit command, 1-400
  -edit compiler argument, 1-407
  edit_form command, 1-405
Editing
  buffers, 1-401
  forms, 1-405
  text files, 1-400
  windows, 1-401
Editing modes
  verbose, 1-463
Editors
  edit command, 1-400, 1-401
  forms, 1-569
  line_edit command, 1-462
Electronic messages, 1-661
$else preprocessor statement, 1-607
$elseif preprocessor statement, 1-607
emacs command, 1-411, 1-767, 1-821
  spell checking, 1-402
Emacs startup file
  specifying, 1-412
Embedded replacement references, 1-773
Embedded-key indexes, 1-227
Embedded-separate-key indexes, 1-227
encode_vos_file command, 1-417
encrypt command, 1-420
(end_of_file) command function, 1-31
$endif preprocessor statement, 1-607
enforce_region_locks command, 1-274,
  1-420, 1-423
Entry map
  creating, 1-49
Entry point
  specifying, 1-55
Entry points, 1-2
eenv source-mode request, 1-254
Environment variables
  setting, 1-664
Error codes
  range of values, B-2
Error messages, 1-329
  from compilation, 1-155, 1-586
.error suffix, 1-98, 1-130, 1-155, 1-435,
  1-586, 1-599, 1-777
Errors
  C preprocessor, 1-102
  compilation, 1-98, 1-130, 1-155, 1-435,
  1-599
Events
limits, B-3 .ex_c suffix, 1-100
Exclamation points, 1-813
preventing abbreviation expansion, 1-813
(exists) command function, 1-32
Expanding abbreviations, 1-349, 1-565, 1-759, 1-812
in debugger requests, 1-243
Expiration dates
copy_dir command removes, 1-181, 1-676
copy_file command removes, 1-188, 1-676
files, 1-276, 1-280, 1-675
move_dir command removes, 1-553, 1-676
move_file command preserves, 1-559, 1-676
Explicit attachment, 1-500, 1-543
Explicit mounting, 1-500, 1-543
expression debugger-request argument, 1-249, 1-250
Expressions
calculating, 1-14
conversion rules, 1-14
evaluation rules, 1-14
in command lines, 1-13
precedence of operators, 1-15
Extended names
enabling for a program module, 1-62
-extended_names argument of
bind, 1-47, 1-65
legacy, 1-35, 1-62
version 1, 1-35, 1-62
version 2, 1-35, 1-62
Extended sequential files, 1-38, 1-1, 1-208, 1-345, 1-490, 1-732, 1-746
(extended_names) command function, 1-34
(extended_names_version) command function, 1-35
Extent-based files, 1-208
advantages and disadvantages, 1-215
and index size, 1-223
cannot be truncated, 1-781
converting to, 1-223
creating, 1-217
definition of, 1-216
file entry for, 1-215
fixed, 1-222
problems with disk fragmentation, 1-217
relative, 1-220
Extent-based indexes, 1-228
Extents, size, B-7
External commands, 1-839
External data names, 1-3
External entries, 1-356
External references, 1-3
External variables, 1-54
messages, 1-54
reading, 1-437
setting, 1-677
shared, 1-54
static variables, 1-73
F
Faults
alignment, 1-628, 1-629
page, 1-628, 1-629
Fences, 1-45
Field definition files, 1-407
Field descriptors, 1-746
Field statements, 1-746
File attributes, comparing, 1-164
File entry
and extent-based files, 1-215
definition of, 1-215
File management
backing up objects, 1-660, 1-816, 1-829
compare_files command, 1-160
convert_stream_file command, 1-173
convert_text_file command, 1-177
copy_file command, 1-187
create_file command, 1-207
creation date, 1-553, 1-559
delete_file command, 1-279
display_dir_status command, 1-319
display_file_status command, 1-337
expiration date, 1-675
file status, 1-423, 1-675, 1-705
links, 1-466
list command, 1-473
move_file command, 1-558
rename command, 1-646
restoring files, 1-654, 1-816, 1-829
save_object command, 1-660
set_file_allocation command, 1-679
set_text_file command, 1-731
Index

sort command, 1-741
truncate_file command, 1-781

File names
- limits, B-3
- maximum length, 1-207

File space, reusing, 1-233

(file_info) command function, 1-37

Files
- abbreviations, 1-812
- access, 1-440, 1-474
- allocating disk space, 1-679
- attributes, 1-337, 1-423, 1-473, 1-675, 1-697, 1-705, 1-731
- backing up, 1-816, 1-829
- backup, 1-403, 1-407, 1-412, 1-463
- batch control, 1-790
- comparing authors, 1-158, 1-162
- converting to extent, 1-223
- copying, 1-190
- creating, 1-207
- creation date, 1-181, 1-188, 1-553, 1-559
- data definition, 1-771
- default character set, 1-224
- definition, 1-85, 1-100, 1-210
- deleting, 1-279
- displaying part of, 1-764, 1-765
- displaying the end of, 1-764, 1-765
- dumping, 1-394
- editing, 1-400
- expiration dates, 1-276, 1-280, 1-675
- extended sequential, 1-38, 1-1, 1-208, 1-345, 1-490, 1-732, 1-746
- extent-based, 1-208, 1-216
- field definition, 1-407
- file entry, 1-215
- finding, 1-519, 1-521, 1-528, 1-531
- fixed, 1-214, 1-219
- form definition, 1-406, 1-569
- implicit locking, 1-681
- index types, 1-229
- indexes, 1-191, 1-226, 1-282
- indexing deleted records, 1-203
- keystrokes, 1-402, 1-463
- limits, B-3
- links to, 1-466
- listing, 1-472
- log, 1-751, 1-760
- maximum block number without extents, 1-219
- merging, 1-741
- message, 1-7, 1-814
- moving, 1-553, 1-558
- number of records in, 1-208

open options
- displaying, 1-358
- setting, 1-691
- organization, 1-208, 1-210
- overwriting, 1-403
- packing, 1-191, 1-552, 1-559, 1-654
- pipe, 1-697
- preventing accidental deletion of, 1-280
- protecting, 1-675, 1-681, 1-695, 1-705, 1-843
- queue, 1-214
- raw, 1-617
- reading from tape, 1-637
- record size, 1-208
- region locking, 1-423
- relative, 1-213, 1-219
- renaming, 1-647
- restoring, 1-654, 1-816, 1-829
- reusing space, 1-202
- safety switches, 1-705
- save, 1-829
- sequential, 1-211, 1-219
- setting attributes, 1-173, 1-177
- shift mode, 1-224
- shorthand, 1-403
- size, 1-473
- sort control, 1-745
- sorting, 1-741
- status, 1-319, 1-337, 1-705
- stream, 1-212, 1-219, 1-423
- stream64, 1-1, 1-213
- truncating, 1-559, 1-781
- unformatted, 1-774
- updating indexes, 1-229, 1-683
- value, 1-770
- writing to tape, 1-846

Finding files, 1-519
Finding indexed files, 1-521
Finding large files, 1-528
Finding stream files, 1-531

First token abbreviations, 1-242, 1-565

Fixed files, 1-214
- converting from stream, 1-239
- converting to stream files, 1-238
- maximum number of blocks with extents, 1-222
- maximum number of records, 1-219
- fixedoverflow compiler argument, 1-592, 1-89, 1-118, 1-429

Fixed-point arithmetic overflow, 1-89, 1-118
- fixedoverflow compiler argument, 1-573, 1-575

Floating-point arithmetic, 1-14
Index

(floor) command function, 1-39
Flow-controlled terminals
  disallowed keystroke assignments, 1-415
Footers, 1-616
-force_write compiler argument, 1-407, 1-571
Form definition files, 1-406, 1-569
  access requirements, 1-410, 1-575
  migrating, 1-409
Form letters
  creating, 1-770
  text_data_merge command, 1-770
Forms editor, 1-405, 1-569
Forms Management System
  Forms Editor, 1-573
  request level, 1-573
fortran command, 1-425
-fortran compiler argument, 1-408, 1-572
fortran debugger mode, 1-250
.fortran suffix, 1-430
-fortran66 compiler argument, 1-429
Freesing devices, 1-108
ftServer modules
  compiler optimizations, 1-95
  optimization levels, 1-153
-full compiler argument, 1-580, 1-592, 1-88, 1-95, 1-120, 1-147, 1-429
Functions, command, 1-1

G
-g (production table) compiler option, 1-113, 1-122, 1-129, 1-817
Gateways, listing, 1-484
Generic control sequences, 1-618, 1-796
get_external_variable command, 1-437
give_access command, 1-439
give_default_access command, 1-444
  (given) command function, 1-40
  Greenwich mean time, 1-734
  (group_name) command function, 1-41

H
handle_sig_dfl command, 1-446
hangul character set, 1-177, 1-210, 1-731
harvest_pc_samples command, 1-16, 1-448
  binding for use of, 1-450
  data accuracy, 1-452
  data collection, 1-451
  execution restrictions, 1-450
  length of time to collect data, 1-453
  use with analyze_pc_samples command, 1-450
  when to use, 1-450
(has_access) command function, 1-42
Headers, 1-616
  displaying program module, 1-372
Heap
  controlling growth of, 1-57
  definition of, 1-807
Heap space
  size, 1-44
Help
  finding the name of a command, 1-457
help command, 1-456
help source-mode request, 1-255
(hexadecimal) command function, 1-44
high_water_mark binder control file directive, 1-66, C-2, C-5, C-7, C-9
(home_dir) command function, 1-45
Home directories, 1-132, 1-536

I
-I compiler option, 1-113, 1-817
I/O
  attaching ports, 1-35
  detaching ports, 1-288
  language conventions, 1-36
  redirecting output, 1-33, 1-294, 1-334
  start_logging command, 1-752
I/O ports, 1-108, 1-490
  attaching, 1-35
  default output port, 1-33, 1-159, 1-287
  detaching, 1-288
  explicit attachment, 1-195
  implicit attachment, 1-195
IBM MVS labels, 1-719
IBM OS/VS labels, 1-719
$if preprocessor statement, 1-607
Implicit attachment, 1-500, 1-544
Implicit locking, 1-681
Implicit mounting, 1-196, 1-504, 1-544
#include compiler argument, 1-114
Include files and debugging, 1-256, 1-258
Include libraries, 1-7, 1-284, 1-486, 1-687
  adding entries, 1-7
  defining path names, 1-687
  deleting path names, 1-285
  listing path names, 1-486
#include source-mode request, 1-256
#include_files compiler argument, 1-89
Index

(index) command function, 1-46
Index flags, 1-683
Index keys, 1-227
  duplicate keys, 1-228
  embedded keys, 1-228
  embedded-separate keys, 1-228
  null keys, 1-228
  sorting (collation) sequence, 1-228
Index statements, 1-749
Indexed files, 1-282, 1-394, 1-395
  copying, 1-192
  dump index, 1-394, 1-395
  index types, 1-227
  keyed index, 1-227
  locating, 1-521
Indexes
  deleting, 1-202, 1-282
  dumping, 1-394
  embedded-key, 1-227
  embedded-separate-key, 1-227
  extent-based, 1-223, 1-228
  item, 1-227
  of files, 1-191, 1-226
  preallocating space, 1-228
  record, 1-202
  recreating, 1-184, 1-187, 1-561
  separate-key, 1-227
  updating, 1-229
Indexing text files, 1-229
Infix operators in command lines, 1-13
Initial command limits, 1-498, 1-808
Interactive processes, 1-509, 1-536
Internal commands, 1-565, 1-839
  -into compiler argument, 1-406, 1-574
Inverse video, 1-725
  (iso_date) command function, 1-47
  (iso_date_time) command function, 1-50
Item indexes, 1-227

K
  kanji character set, 1-177, 1-210, 1-731
  katakana character set, 1-177, 1-210, 1-731
  Keep files, 1-78
  Keep module, 1-242, 1-255
  keep source-mode request, 1-255
  Kernel loadable programs, 1-47, 1-49
  Kernel processes, 1-509
  Kernel references, 1-48
  Key clicks, 1-727
  Key statements, 1-748
  Keys, data type of, 1-230
  Keystrokes files, 1-402, 1-412, 1-463
    creating multiple, 1-413
    kill command, 1-459
    .kp suffix, 1-78, 1-242

L
  label debugger-request argument, 1-248
  Labeled tapes, 1-504, 1-715
  -language compiler argument, 1-147
  Language support
    file attributes, 1-177, 1-731
    set_language command, 1-684
    (language_name) command function, 1-53
    latin_1 character set, 1-177, 1-210, 1-731
    latin_9 character set, 1-177, 1-210, 1-731
    ldd command, 1-461
  Legacy extended names, 1-35, 1-62
  (length) command function, 1-54
  Levels of compilation for COBOL, 1-147
  Libraries, 1-7, 1-687
    adding entries, 1-6
    command, 1-6, 1-284, 1-485, 1-686, 1-839
    defining path names, 1-686
    deleting entries, 1-284
    directories, 1-6, 1-284, 1-485, 1-686
    include, 1-6, 1-284, 1-485, 1-686
    listing entries, 1-485
    message, 1-6, 1-284, 1-485, 1-686
    object, 1-2, 1-6, 1-284, 1-485, 1-686
    setting path names, 1-686
    shared, 1-50, 1-51, 1-75, 1-368
      creating, 1-64
      listing, 1-461, 1-483
      naming conventions, 1-63, 1-370
      setting breakpoints, 1-251
    -library compiler argument, 1-407, 1-571
  Library paths, 1-6, 1-284, 1-686
    checking, 1-687
    displaying, 1-485

Limits
  general OpenVOS, B-1
  line debugger-request argument, 1-248
  line_edit command, 1-462
  link command, 1-465
  link_dirs command, 1-468
  Links, 1-465, 1-841
    circular, 1-466
    communication, 1-104
    creating, 1-2
    deleting, 1-786
    limits, B-4
    listing, 1-472, 1-474
    nested, 1-466
    renaming, 1-647
    specifying in move_file command, 1-561
targets, 1-184, 1-466, 1-474
to devices, 1-466
to entry points, 1-3
unlinking, 1-466, 1-786
use in compare_files command, 1-164
use in copy_dir command, 1-184
use in copy_file command, 1-191
use in move_dir command, 1-555
-linted compiler argument, 1-90, 1-97
list command, 1-471
-list compiler argument, 1-578, 1-590, 1-86,
   1-95, 1-101, 1-119, 1-147, 1-427
list source-mode request, 1-256
$list suffix, 1-95, 1-130, 1-153, 1-432, 1-583,
   1-596
list_batch_requests command, 1-477
list_devices command, 1-481
list_dynamic_dependencies command, 1-483
list_gateways command, 1-461, 1-483,
   1-484
list_library_paths command, 1-485
list_modules command, 1-487
list_port_attachments command, 1-489
list_print_requests command, 1-494
list_process_cmd_limits command, 1-497
   and update_process_cmd_limits command, 1-498
list_save_tape command, 1-500
list_systems command, 1-502
list_tape command, 1-504
list_terminal_types command, 1-506
list_users command, 1-508
Listing
device path names, 1-482
directories, 1-472
files, 1-472
gateways, 1-484
links, 1-472, 1-474
modules, 1-487
ports, 1-489
processes, 1-508
shared libraries, 1-461, 1-483
systems, 1-502
tape contents, 1-504
Listings
assembly language, 1-88, 1-120, 1-147,
   1-150, 1-432, 1-580, 1-592
compilation, 1-86, 1-119, 1-145, 1-427,
   1-578, 1-583, 1-590
cross-reference, 1-87, 1-119, 1-147, 1-427,
   1-578, 1-590
object, 1-153, 1-430, 1-583
source, 1-153, 1-432, 1-583
Literal text, displaying, 1-349
Load point, 1-44, 1-58, 1-59
-load_in_kernel binder argument, 1-47
-load_point binder argument, 1-44, 1-59
load_point binder control file directive, 1-66
Local abbreviations, 1-466
   links, 1-466
locate_expansible_dirs
   command, 1-517
locate_files command, 1-517, 1-519,
   1-526
locate_indexed_files command, 1-521
locate_large_dirs command, 1-526
locate_large_files command, 1-528
locate_stream_files command, 1-531
Locating an object, 1-841
Locating indexed files, 1-521
(lock_type) command function, 1-55
(locked) command function, 1-56
Locking
files, 1-681
   regions, 1-423
Locks
   identifying, 1-843
Log files, 1-751
   start_logging command, 1-751
   stop_logging command, 1-760
Logging in, 1-534
   remotely, 1-103
Logging out, 1-538
Logical expressions in command lines, 1-13
Logical records
   comparing, 1-159
login command, 1-534
logout command, 1-538
(ltrim) command function, 1-57

M
-M compiler option, 1-113
machine debugger mode, 1-261
   PL/I syntax in, 1-265
Macros, 1-839
-main compiler argument, 1-150
-map binder argument, 1-47
.map suffix, 1-56
-mapcase compiler argument, 1-579, 1-591,
   1-88, 1-98, 1-117, 1-129, 1-149, 1-155,
   1-428
.pl1 command, 1-598
Index

displaying default with `display_error` command, 1-86, 1-578, 1-590

Maps, bind, 1-56

(master_disk) command function, 1-58
(max) command function, 1-59

-max_heap_size binder argument, 1-44, 1-60
max_heap_size binder control file directive, 1-66

-max_program_size binder argument, 1-45, 1-59
max_program_size binder control file directive, 1-66

-max_stack_size binder argument, 1-45
max_stack_size binder control file directive, 1-66

Maximum command limits, 1-498
Maximum heap size, specifying, 1-60

Memory pool

  start_process command, 1-759

Merging files, 1-741

(message) command function, 1-60

Message libraries, 1-7, 1-284, 1-486, 1-687
adding entries, 1-7
defining path names, 1-687
deleting path names, 1-285
listing path names, 1-486

Message queue, 1-214

Messages
  as external variables, 1-54
  at task completion, 1-39, 1-619, 1-796
  broadcasting, 1-351
  compilation errors, 1-592
displaying, 1-351
error, 1-329
send_message command, 1-661
sending, 1-662
system, 1-351
tape information, 1-236, 1-289, 1-541, 1-710

Micro-jiffies, B-7

(min) command function, 1-62
(mod) command function, 1-63
Modifying print requests, 1-792
Module default command limits, 1-498, 1-807

(module_info) command function, 1-64
(module_name) command function, 1-67

Modules
  current, 1-304
  keep, 1-242, 1-255
  listing, 1-487
  object, 1-121, 1-352
  processing, 1-304
  program, 1-54
mount_tape command, 1-236, 1-540, A-4
move_device_reservation command, 1-550
move_dir command, 1-552
  and expiration dates, 1-553
  pipe files, 1-553
  specifying links in, 1-555
move_file command, 1-558
  and expiration dates, 1-559
  pipe files, 1-560
  specifying links in, 1-561

Moving
  directories, 1-553
  files, 1-553, 1-558
mp_debug command, 1-564

mp_debug requests
  exclude_process, 1-566
  help, 1-568
  include_process, 1-566
  list_processes, 1-566
  mp_login, 1-566
  quit, 1-568
  restart, 1-567
  start_process, 1-567
  stop, 1-568
  suspend_process, 1-567
  use_process, 1-567
Multivolume tape files, 1-707

N

(name_string) command function, 1-68
National Language Support, 1-110, 1-338, 1-731
character sets, 1-413
reading a tape, 1-639
writing a tape, 1-846
Nested links, 1-466

-nesting compiler argument, 1-581, 1-592, 1-89, 1-95, 1-118
Nesting levels
  in OpenVOS C programs, 1-89
  in OpenVOS Pascal programs, 1-581
  in OpenVOS PL/I programs, 1-592
  in OpenVOS Standard C programs, 1-118
Network gateways, 1-484
Network limits, B-4
Network, X.25, 1-105
nls_edit_form command, 1-569
-no_backup compiler argument, 1-571
-no_dynamic_tasking binder argument, 1-47
-no_edit compiler argument, 1-571
-no_include source-mode request, 1-256
-no_optimize compiler argument, 1-96, 1-154
-no_pll_template compiler argument, 1-572
Noninteractive processes, 1-37, 1-757
Nonprinting characters, 1-617, 1-794
Nonstandard labels, 1-719
Notify messages, 1-662
number debugger-request argument, 1-248

O
-o compiler option, 1-113, 1-122, 1-817
-o compiler option, 1-113, 1-817
Object libraries, 1-7, 1-284, 1-486, 1-687
adding entries, 1-7
defining path names, 1-687
deleting path names, 1-285
listing path names, 1-486
Object listings, 1-153, 1-430
Object modules, 1-430
binding, 1-54
common, 1-627
creating, 1-200
displaying information about, 1-352
external references, 1-3
performance information, 1-625
(object_name) command function, 1-69
Object libraries, 1-7, 1-284, 1-486, 1-687
adding entries, 1-7
defining path names, 1-687
deleting path names, 1-285
listing path names, 1-486
Object listings, 1-153, 1-430
Object modules, 1-430
binding, 1-54
common, 1-627
creating, 1-200
displaying information about, 1-352
external references, 1-3
performance information, 1-625
(object_name) command function, 1-69
Objects, locating, 1-841
Offline systems, 1-503
One-way direct queues, 1-215
One-way server queues, 1-214
(online) command function, 1-70
Online help, 1-456
Online systems, 1-503
Open options, 1-309, 1-359, 1-667, 1-691
OpenVOS
analyzing, 1-19
Operands in command lines, 1-13
Operators
in command lines, 1-13
order of evaluation, 1-15
precedence in command lines, 1-13
Optimization level, specifying, 1-597
-optimization_level compiler argument, 1-581, 1-593
Optimization levels
cc command, 1-122, 1-123
checking uninitialized variables, 1-116, 1-149, 1-430, 1-585
default, 1-96
display debugger request, 1-253
ftServer modules, 1-95, 1-153, 1-584, 1-596
-no_optimize compiler argument, 1-96, 1-154, 1-585, 1-591, 1-597
profile command requirements, 1-97, 1-154, 1-434, 1-585, 1-597
set debugger request, 1-258
specifying, 1-96, 1-154, 1-434, 1-585
-optimization_level compiler argument, 1-579, 1-591, 1-87, 1-149, 1-428
Optimized code
depth debugging, 1-244, 1-253, 1-258
Options to the cc command, 1-121
Organization statements, 1-750
Organization, files, 1-208
.out suffix, 1-758, 1-33, 1-38, 1-40
Output
directing, 1-292
redirecting, 1-334
Overwriting files, 1-403
P
-P compiler option, 1-113
Packing files, 1-191
copying a directory, 1-184, 1-187
moving a directory, 1-552
moving a file, 1-559
restoring an object, 1-654
Page faults, 1-628, 1-704
Paging partitions, 1-211, 1-323
Parameters
mount_tape command, A-4
set_tape_drive_params command, 1-389, A-4
set_tape_file_params command, 1-389, A-4
set_tape_mount_params command, 1-389, A-4
Parentheses, in command lines, 1-1, 1-13
Partial display of files, 1-764, 1-765
pascal command, 1-576, 1-581
  -check_uninitialized compiler argument, 1-585
-pascal compiler argument, 1-408, 1-572
pascal debugger mode, 1-250
.pascal suffix, 1-581
Passing data, 1-697
Passwords
  changing, 1-536
  logging in, 1-535
  verifying system access, 1-832
Path names
  add_library_path command, 1-6
  change_current_dir command, 1-132
  create_dir command, 1-205
  (current_dir) command function, 1-22
  (current_module) command function, 1-23
  delete_library_path command, 1-284
  (directory_name) command function, 1-30
  (home_dir) command function, 1-45
  link command, 1-465
  list_library_paths command, 1-485
  maximum length, B-5
  (path_name) command function, 1-71
  (posix_path) command function, 1-73
  (process_dir) command function, 1-75
  (referencing_dir) command function, 1-81
  set_library_paths command, 1-686
  set_line_wrap_width command, 1-689
  (vos_path) command function, 1-106
  where_command command, 1-839
  where_path command, 1-841
  (where_path) command function, 1-108
  (path_name) command function, 1-71
PC. See Program counter system
Performance information
  CPU time, 1-629
  execution coverage, 1-629
  frequency of statement execution, 1-629, 1-633
  page faults taken, 1-629
  profile command, 1-625
  (person_name) command function, 1-72
Pipe files
  attributes, 1-697
  copying, 1-182
  limits, B-5
  move_dir command, 1-553
  move_file command, 1-560
  set_pipe_file command, 1-697
PL/I data types, 1-747
pl1 command, 1-588
  -check_uninitialized compiler argument, 1-598
  -define compiler argument, 1-594
  -mapcase compiler argument, 1-598
  -production_table compiler argument, 1-598
  -table compiler argument, 1-598
  -pl1 compiler argument, 1-408, 1-572
  pl1 debugger mode, 1-250
  .pl1 suffix, 1-593
  -pl1_template compiler argument, 1-408, 1-574
  .pm suffix, 1-55, 1-78
See also Program modules
Ports
  attached file types, 1-490
  attaching, 1-35
  attributes, 1-491
  default output, 1-33, 1-287
  detaching, 1-287, 1-288, 1-538
  displaying information, 1-489
  explicit attachment, 1-543
  I/O, 1-658
  implicit attachment, 1-544
  limits, B-5
  listing attachments, 1-490
  reattaching, 1-287
  position source-mode request, 1-256
  position_tape command, 1-601, 1-604, 1-834
POSIX
  changing signal behavior, 1-446
  checking for compliance, 1-137
  verifying the validity of a directory tree’s ACLs, 1-824
  (posix_path) command function, 1-73
  posixpath command, 1-604
Precision of floating-point arithmetic, 1-14
Predefined preprocessor variables
  c, 1-92
  cc, 1-126
  cobol, 1-152
  fortran, 1-431
Index

pascal, 1-583
pl1, 1-595
-prefix compiler argument, 1-407, 1-571
Prefix operators, in command lines, 1-13
Prelogin processes, 1-510
preprocess_file command, 1-606
Preprocessing
  access requirements, 1-610
  binder, 1-44
  files, 1-607
Preprocessor statements, 1-594, 1-607
Preprocessor symbols
  limits, B-5
Preprocessor variables, 1-92, 1-102
  IA-32, 1-92, 1-102, 1-126, 1-152, 1-431,
  1-583, 1-595
Primary user, 1-482
print command, 1-614
Print queues, 1-109, 1-362, 1-494, 1-614
  default, 1-361
  display_print_status command, 1-363
Print requests, modifying, 1-792
Printer information, 1-363, 1-494
Printers
  directing output to, 1-333
Printing, 1-614
  canceling, 1-109
  deferred, 1-794
  deferring, 1-616
  entering print requests, 1-614
  footers, 1-616
  headers, 1-616
  listing requests, 1-494
  queue priority, 1-618, 1-796
  wrapping, 1-618, 1-796
Priority
  batch process, 1-38
  of login process, 1-536
  of processes, 1-699
  queue, 1-38
Priority levels
  of batch requests, 1-789
  setting for processes, 1-758
Privileged processes, 1-327, 1-535, 1-758, 1-789
  batch, 1-38
Process address space
  setting highest boundary of, 1-58
  specifying size of, C-5
Process control, 1-665
  (process_dir) command function, 1-75
  (process_info) command function, 1-77
  (process_type) command function, 1-78
Processes
  assigning devices, 1-649
  batch, 1-37, 1-39, 1-108
  creating, 1-534
  current command limits, 1-808
  debugging multiple, 1-564
  initial command limits, 1-808
  interactive, 1-509, 1-536
  kernel, 1-509
  listing, 1-508
  maximum command limits, 1-809
  maximum number, B-5
  memory pool for, 1-759
  noninteractive, 1-39, 1-757
  prelogin, 1-510
  priority, 1-38, 1-536, 1-699, 1-758, 1-789
  privileged, 1-327, 1-535, 1-758, 1-789
  scheduler limits, B-5
  setting code and data size of, 1-59
  slave, 1-564
  starting, 1-759
  states, 1-566
  stopping, 1-761
  suspending, 1-739
  system, 1-510
  terminating, 1-538
Processing
  deferred, 1-790
  tapes, 1-290
Processing module, 1-304
-prefix binder argument, 1-45
  displaying default with display_error
  command, 1-45
-prefix compiler argument, 1-578, 1-590,
  1-86, 1-92, 1-101, 1-115, 1-125, 1-146,
  1-152, 1-409, 1-427, 1-431, 1-572,
  1-582, 1-595
  displaying default with display_error
  command, 1-86, 1-578, 1-590
-produce_symtab compiler argument, 1-409
Production symbol tables, 1-87, 1-149
-production_table compiler
  argument, 1-579, 1-584, 1-591
  Production tables, 1-87, 1-122, 1-149
-production_table compiler
  argument, 1-87, 1-97, 1-148, 1-153,
  1-428
  effect on debug command, 1-87, 1-148,
  1-579, 1-591
pl1 command, 1-598
profile command, 1-622
  and compare_files command, 1-627
  and display_program_module
  command, 1-627

OpenVOS Commands Reference Manual (R098)  Index-19
Index

Index-20

binding for, 1-626
combined and differential mode, 1-628
combined mode, 1-626
compilation requirements, 1-625
differential mode, 1-627
differential mode, 1-627
diffrential, non-combined mode, 1-630
executing program modules for, 1-626
preparing to use, 1-625
- profile compiler argument, 1-628
profile command, 1-628
.profile suffix, 1-10, 1-623
-profile_alignment_faults binder argument, 1-48
Profes
add_profile command, 1-10
profile command, 1-622
Program counter (PC) system, 1-15
illustration of, 1-452
Program counter system, 1-448
Program modules, 1-54
analyzing, 1-19
analyzing relationship to OpenVOS, 1-20
displaying header, 1-372
displaying pseudo-assembly code, 1-370
maximum size, B-5
naming, 1-55
See also Optimization levels
selecting page to display, 1-370
specifying in analyze_pc_samples command, 1-16
specifying maximum size, 1-45
target, 1-450
Prompts
displaying, 1-640
setting, 1-703, 1-724
tape mounting, 1-544, 1-546
propagate_access command, 1-634
Protecting files, 1-675, 1-681, 1-695, 1-705
Queues
batch, 1-38, 1-106, 1-301, 1-477, 1-789
maximum length, B-5
print, 1-109, 1-363, 4-94, 1-614
priority, 1-618, 1-789, 1-796
quit source-mode request, 1-257
(quote) command function, 1-79
R
RAM usage, 1-701
(rank) command function, 1-80
Raw files, 1-617
Read only buffers, 1-401
read_tape command, 1-637
Reading tape files, 1-637
ready command, 1-640
Ready messages, 1-640, 1-703
Record indexes, 1-202
deleted, 1-202
Record mapping, 1-233
Records
deleted, 1-202
dumping, 1-396
length, B-7
limits, B-6
size of, 1-208
specifying number of, 1-208
Recreating indexes
copying a directory, 1-184, 1-187
copying a file, 1-191
moving a directory, 1-555
moving a file, 1-561
-recursive compiler argument, 1-430
Recursive subroutines, 1-430
Reference modifier
definition of, 1-249
-references_kernel binder argument, 1-48
(referencing_dir) command function, 1-81
Region locking, 1-423
region_load_point binder control file directive, 1-70
Registers and debugging, 1-257, 1-262
-registers compiler argument, 1-90
Registration files, 1-537
regs source-mode request, 1-257
relational_expression
debugger-request argument, 1-265, 1-270
Q
-q compiler options, 1-113
Queue files, 1-214
message, 1-214
one-way direct, 1-215
one-way server, 1-214
two-way direct, 1-215
two-way server, 1-214

Index-20
Index

Relative files, 1-213
  maximum number of blocks with
  extents, 1-220
  maximum number of records, 1-219
-relocatable binder argument, 1-49
Remote logins, 1-103
  initiating, 1-103
  terminating, 1-538
remove_access command, 1-642
remove_default_access
  command, 1-644
rename command, 1-646
Renaming, 1-646
  copy_dir command, 1-184
  copy_file command, 1-191
  directories, 1-647
  files, 1-647
  links, 1-647
  move_dir command, 1-553
  move_file command, 1-560
Replacement references
  embedded, 1-773
Replacing directories, 1-553
request_list debugger-request
  argument, 1-248, 1-251, 1-255
reserve_device command, 1-649, 1-650
  canceling, 1-108
reset_eof command, 1-650
Restarting a sleeping process, 1-739
.restore suffix, 1-656
restore_object command, 1-653
Restoring
  objects, 1-500, 1-829
  saved objects, 1-655, 1-829
Restrictions on binding external variables, 1-584
retain binder control file directive, 1-71
  -retain_all binder argument, 1-49
return source-mode request, 1-257
Reusing file space, 1-202, 1-233
(reverse) command function, 1-82
Rewinding tapes, 1-601, 1-604, 1-834
Rings, write, 1-542
Root directory, 1-519
(rtrim) command function, 1-83

S
SAE files. See Statically-allocated extents (SAE)
  files
SAE indexes. See Statically-allocated extents
  (SAE) indexes
Safety switches, 1-705
Save files, 1-829
Save tapes, 1-500, 1-829
save_object command, 1-658
Saving objects, 1-500
  to tape or disk, 1-659, 1-829
Scrolling, 1-727
(search) command function, 1-84
-search compiler argument, 1-44
Search lists, 1-7, 1-285, 1-486, 1-687, 1-839
section binder control file directive, 1-72
  -segmentation compiler argument, 1-151
send_message command, 1-661
Sending messages, 1-662
Separate-key indexes, 1-227, 1-683
Sequential files, 1-211
  maximum record byte offset, 1-219
Server queues, 1-214
set command, 1-664
set macro statement, 1-15
set source-mode request, 1-258
  and optimized code, 1-258
set_cpu_time_limit command, 1-665
set_default_open_options
  command, 1-667
set_dir_limits
  command, 1-672
set_dir_type command, 1-672
setExpirationDate command, 1-669,
  1-672, 1-675
  preventing file deletion with, 1-280
set_external_variable
  command, 1-677
set_file_allocation
  command, 1-679
set_implicit_locking
  command, 1-681
set_index_flags
  command, 1-683
set_language
  command, 1-684
set_library_paths
  command, 1-686, 1-689
set_open_options command, 1-691
set_owner_access command, 1-309,
  1-358, 1-667, 1-691, 1-694
set_pipe_file command, 1-697
set_priority command, 1-699
set_ram_file command, 1-701
set_ready command, 1-703
set_safety_switch
  command, 1-705
  allows file editing, 1-706
  preventing file deletion with, 1-280
  preventing file truncation with, 1-706
set_second_tape command, 1-707
set_tape_drive_params
  command, 1-389, 1-709, A-4
set_tape_file_params command, 1-389,
  1-541, 1-712, A-4
set_tape_mount_params
  command, 1-389, 1-717, A-4

OpenVOS Commands Reference Manual (R098)  Index-21
set_terminal_parameters
command, 1-721
argument order processing, 1-729
(terminal_name) command
function, 1-721
set_text_file command, 1-731
set_time_zone command, 1-734
Setting breakpoints, 1-244, 1-250, 1-268
Setting tape
  drive parameters, 1-709
  file parameters, 1-712
  mount parameters, 1-717
Shared external variables, 1-54
Shared libraries, 1-50, 1-51, 1-75, 1-368
creating, 1-64
listing, 1-461, 1-483
naming conventions, 1-63, 1-370
setting breakpoints, 1-251
Shared static region
definition of, 1-807
Shared variables, 1-74
Shared virtual memory databases
  create_data_object
    command, 1-201, C-1
    high_water_mark bind directive, 1-66
    in program module, C-2
    outside program module, C-3
Shift modes, 1-210, 1-338, 1-414
  -short_integer compiler argument, 1-430
  -short_logical compiler argument, 1-430
Shorthand files, 1-403
  -show_include compiler argument, 1-120
  -show_macros compiler argument, 1-89,
    1-120
  -silent compiler argument, 1-580, 1-592,
    1-88, 1-101, 1-150, 1-429
  simplified_chinese character set, 1-177,
    1-210, 1-731
  size binder argument, 1-58, C-5
  size binder control file directive, 1-72
Size limits for text files, 1-404
Slave processes, 1-564
sleep command, 1-738
Sleeping processes, restarting, 1-739
  (software_purchased) command
    function, 1-85
sort command, 1-740
Sort control files, 1-745
.sort_exc suffix, 1-742
Sort keys, 1-742
  -sort_into_by_alignment compiler
    argument, 1-409, 1-574
Sorting
  files, 1-741
  statistics, 1-742
Source listings, 1-153, 1-432
Source modules, 1-85, 1-112, 1-145, 1-426,
  1-589
  expanded, 1-100, 1-198
source source-mode request, 1-258
  and recursive procedures, 1-258
source_path source-mode request, 1-259
Spaces in command lines, 1-13
Specifying
  alignment, 1-46
  heap space size, 1-44
  kernel loadable programs, 1-47
  load point, 1-44, 1-59
  maximum heap size, 1-60
  maximum program size, 1-45
  maximum stack size, 1-45
  process address space size, C-5
  stack fence size, 1-45
  stack size, 1-46
  target module, 1-46
Spell checker, emacs command, 1-413
Spooler requests, 1-614
  canceling, 1-110
  spooler_configuration.v1.tin file,
    -pass_thru argument, 1-619
Spoolers, 1-362, 1-364
Stack fences, 1-45
  size, 1-60
Stack frames
  limits, B-6
Stack size, 1-46
  maximum, 1-45
  -stack_fence_size binder argument, 1-45,
    1-60
  stack_fence_size binder control
    directive, 1-73
  -stack_size binder argument, 1-46
  stack_size binder control directive, 1-73
Stacks
  controlling growth of, 1-57
  definition of, 1-807
Standard deviation of transaction execution
times, 1-31
Star names, 1-190
start source-mode request, 1-259
start_logging command, 1-751
start_process command, 1-757
  memory pool, 1-759
start_up command macro, 1-759, 1-812
Starting processes, 1-757
Static external variables
  reading, 1-437
Static region
  definition of, 1-807
Static tasks
  definition of, 1-47
  multiple, 1-59
  setting fence size, 1-59
  setting stack size, 1-59
  single, 1-59
  specifying stack fence size, 1-60
Statically-allocated extents (SAE)
  files
    creating, 1-215
    recovering from failures, 1-218
  indexes, 1-230
Statistics
  binder, 1-49
  compiler, 1-88, 1-119, 1-150, 1-429, 1-580, 1-592
  PC sampler, 1-25
  preprocessor, 1-101
  profile, 1-632
  sorting, 1-742
  -statistics binder argument, 1-49
Status codes, 1-329
Status lines, 1-662, 1-724
  step source-mode request, 1-260
  stop_logging command, 1-760
  stop_process command, 1-761
Stopping processes, 1-761
Storage system management, 1-322
  -store_args compiler argument, 1-90, 1-120, 1-593
Storing objects, 1-500
  on tape or disk, 1-659
StrataDOC, 1-456
Stream files, 1-212, 1-423
  64-bit, 1-212
  converting, 1-173
  converting from fixed files, 1-238
  converting to fixed, 1-239
  maximum record byte offset, 1-219
Stream64 files, 1-1, 1-213
  (string) command function, 1-86
Subprocesses, 1-537
  logging into, 1-536
  -subroutines_are_functions binder argument, 1-49
Subsequent processes
  logging into, 1-536
  (substitute) command function, 1-90
  (substr) command function, 1-91
substring debugger-request argument, 1-248
Substrings, 1-2
Subsystems, 1-536
  logging into, 1-536
Suffixes
  .batch, 1-40
  .bind, 1-64
  .c, 1-91, 1-121
  .cobol, 1-151
  .dd, 1-742, 1-776
  .ex_c, 1-100
  .fortran, 1-430
  .kp, 1-78, 1-242
  .list, 1-95, 1-130, 1-153, 1-432, 1-583, 1-596
  .map, 1-56
  .out, 1-33, 1-38, 1-40, 1-758
  .pascal, 1-581
  .pl1, 1-593
  .pm, 1-55, 1-78
  .profile, 1-10, 1-623
  .restore, 1-656
  .sort_exc, 1-742
  .tdm_out, 1-771
  .tin, 1-775
  -suppress_diag compiler argument, 1-86, 1-114
Suppressing warning messages, 1-580
Suspending a process, 1-739
Symbol source-mode request, 1-260
synonym binder control file directive, 1-73
System
  dictionaries, 1-402
  listing, 1-502
  listing modules, 1-487
  messages, 1-351, 1-662
  performance, 1-508
  processes, 1-510
  registration files, 1-537
  usage, 1-382
  (system_info) command function, 1-92
  (system_name) command function, 1-94
  -system_programming compiler argument, 1-89, 1-118, 1-581, 1-593
Index

T
Tab settings, 1-725
-table binder argument, 1-49
-table compiler argument, 1-578, 1-584,
  1-591, 1-87, 1-96, 1-97, 1-118, 1-122,
  1-148, 1-153, 1-154, 1-428
pli command, 1-598
-tail_file command, 1-764
Tape drive parameters, A-1
  setting, 1-709
Tape drives, 1-500, 1-707
cartridge, 1-236
Tape file formats
  ANSI value, 1-715
  IBM value, 1-715
Tape file parameters, A-1
  setting, 1-712
Tape files
  reading, 1-637
  writing, 1-845
Tape labels, 1-236, 1-541, 1-542
Tape marks
  definition of, 1-719
Tape mount parameters, A-1
  setting, 1-717
Tape mounting
  automatic, 1-500
  explicit, 1-500
Tape parameter values, 1-388, 1-393, 1-710,
  1-718, A-1
Tape processing
  access rights, 1-541, 1-545, 1-718
  copying tapes, 1-195
  dismounting tapes, 1-289
  listing tapes, 1-500, 1-504
  mounting tapes, 1-540
  position_tape command, 1-601,
    1-604, 1-834
  reading tapes, 1-637
  set_second_tape command, 1-707
  set_tape_file_params command, 1-712
  set_tape_mount_params command, 1-717
  setting tape defaults, 1-709
  verify_save command, 1-816, 1-829
  writing tapes, 1-845
Tapes
  automatic mounting, 1-501, 1-544
  backing up, 1-500
  contents, 1-504
  copying, 1-195
  dismounting, 1-289, 1-290

disposition at closing, 1-710
dumping, 1-397
explicit mounting, 1-543
I/O, 1-637, 1-709, 1-712, 1-717, 1-845
I/O ports, 1-388, 1-540
implicit mounting, 1-196, 1-504, 1-544
initializing, 1-235
labels, 1-236, 1-541, 1-542
listing contents, 1-504
mounting, 1-540
multivolume tape files, 1-707
port attachment, 1-544
processing, 1-389, 1-637, 1-709, 1-712,
  1-717, 1-845
rewinding, 1-544, 1-601, 1-604, 1-710,
  1-834
save, 1-500, 1-829
translation mode, 1-236, 1-541, 1-718
unloading, 1-289, 1-542, 1-543, 1-710
tar tape format, 1-714, 1-719
-target_module binder argument, 1-46
Targets of links, 1-184, 1-466, 1-474
task_status source-mode request, 1-260
  .tdm_out suffix, 1-771
temacs command, 1-767
  (terminal_name) command function, 1-97
  set_terminal_parameters command, 1-721
Terminal parameters, 1-729
  audible keys, 1-727
  break, 1-725
  break action, 1-727
  continue, 1-724
  cursor, 1-726
  display of display forms, 1-726
  escape character, 1-724
  insert/overlay mode, 1-726
  interrupt key, 1-727
  inverse video, 1-725
  invoking window manager, 1-728
  line length, 1-723
  pause, 1-724
  prompt, 1-724
  reuse of terminal memory pages, 1-727
  scroll screen, 1-727
  setup name, 1-724
  status line, 1-724
  system message, 1-724
  tabs, 1-725
  terminal type, 1-724, 1-729
  type ahead lines, 1-727
  use of <RETURN> key, 1-728
  use of CANCEL key, 1-728
  window terminal, 1-727

Index-24
Index

Terminal types, 1-506
(terminal_info) command function, 1-96
Terminals
  virtual, 1-104
Terminating processes, 1-538
Text buffers, 1-401
Text editors
  edit command, 1-400
  emacs command, 1-411
  line_edit command, 1-462
  temacs command, 1-767
  vemacs command, 1-821
Text files
  default character set, 1-338, 1-731
  editing, 1-400
  indexing, 1-229
  preprocessing, 1-606
  reading from tape, 1-639
  setting attributes, 1-177, 1-731
  shift mode, 1-338, 1-731
  size limit, 1-404
  unformatted, 1-774
  writing to tape, 1-846
Text substitution, 1-770
  text_data_merge command, 1-770
  (time) command function, 1-98
Time zones, 1-734
  codes, 1-734
Time zones supported in OpenVOS, 1-734
  .tin suffix, 1-775
trace source-mode request, 1-260
Transactions
  definition of, 1-18
  measuring standard deviation of execution time, 1-31
  (translate) command function, 1-100
  translate_links command, 1-778
Translation mode, tapes, 1-236, 1-541, 1-718
  (trunc) command function, 1-101
  truncate_file command, 1-781
  extents, 1-781
  -truncate_to compiler argument, 1-117, 1-129
Truncating files, 1-559
Two-way direct queues, 1-215
Two-way server queues, 1-214
  -type_checking compiler argument, 1-91, 1-93, 1-116, 1-127
Types of commands, 1-456

U
  -U compiler option, 1-113, 1-818
  -u compiler option, 1-114
unbundle command, 1-783
$undefine preprocessor statement, 1-607
Undefined access, 1-5, 1-307, 1-445
Unformatted text files, 1-774
(uniqie_string) command function, 1-102
UNIX tape formats
  cpio, 1-714, 1-719
  cpio, 1-714, 1-719
  tar, 1-714, 1-719
Unlabeled tapes, 1-715, 1-719
unlink command, 1-786
Unlinking links, 1-786
  (unquote) command function, 1-103
Unreachable code
  preventing elimination of, 1-96, 1-124, 1-597
Unshared variables, 1-74
Update_batch_requests
  command, 1-788
  update_print_requests
  command, 1-792
  update_process_cmd_limits
  command, 1-799
  and list_process_cmd_limits
  command, 1-498
Updating file indexes, 1-683
USB devices
  displaying information, 1-393
use_abbreviations command, 1-497, 1-812
use_message_file command, 1-814
User dictionaries, 1-413
User information, displaying, 1-508
User loads, 1-508
User registrations, 1-700
User tape parameters, 1-388, A-1
user_dbcs character set, 1-177, 1-210, 1-731
  (user_name) command function, 1-104
V
Value files, 1-770
variable debugger-request argument, 1-249
variable_arg_count directive, 1-73
Variables
  checking uninitialized, 1-116
  external, 1-54, 1-437, 1-677
  shared, 1-74
  unshared, 1-74
variables directive, 1-73
vcc command, 1-816
vcpp command. See cpp command, 1-198
vemacs command, 1-821
(verify) command function, 1-105
Index

verify_posix_access command, 1-824
verify_save command, 1-829
verify_system_access command, 1-832
Version 1 extended names, 1-35, 1-62
Version 2 extended names, 1-35, 1-62
-version binder argument, 1-46
Virtual terminals, 1-104
visibility directive, 1-75
(vos_path) command function, 1-106
vospath command, 1-834
vsleep command. See sleep
command, 1-738
vsort command. See sort command, 1-740

W
-W compiler option, 1-114, 1-818
-w compiler option, 1-114
Waking up a sleeping process, 1-739
walk_dir command, 1-836
Watchpoints
clearing, 1-252
displaying information about, 1-256
setting, 1-260
where source-mode request, 1-261
where_command command, 1-839
where_path command, 1-841
(where_path) command function, 1-108
who_locked command, 1-843
Window terminals
breaks, 1-727
insert/overlay mode, 1-728
invoking, 1-728
pause, 1-728
reuse of terminal memory pages, 1-727
set_terminal_parameters
command, 1-722
tab, 1-729
use of [RETURN] key, 1-728
use of CANCEL key, 1-728
Windows
edit command, 1-401
emacs command, 1-412
Word processors
edit command, 1-401
emacs command, 1-411
temacs command, 1-767
vemacs command, 1-821
Working directories, 1-132, 1-134, 1-303
Write rings, 1-542
write_tape command, 1-845
Writing files to tape, 1-845

X
-X compiler options, 1-114, 1-124
X.25 network, 1-105
-xref compiler argument, 1-578, 1-590, 1-87,
1-95, 1-119, 1-147, 1-427
-xref_format compiler argument, 1-148