# QUICK REFERENCE GDB Version 5

## ial Commands

| | |
|---|---|
| *ram* [*core*] | debug *program* [using coredump *core*] |
| *unction* | set breakpoint at *function* [in *file*] |
| *ist*] | start your program [with *arglist*] |
| | backtrace: display program stack |
| | display the value of an expression |
| | continue running your program |
| | next line, stepping over function calls |
| | next line, stepping into function calls |

## ng GDB

| | |
|---|---|
| | start GDB, with no debugging files |
| *ram* | begin debugging *program* |
| *ram core* | debug coredump *core* produced by *program* |
| lp | describe command line options |

## ng GDB

| | |
|---|---|
| | exit GDB; also q or EOF (eg C-d) |
| PT | (eg C-c) terminate current command, or send to running process |

## g Help

| | |
|---|---|
| | list classes of commands |
| *ss* | one-line descriptions for commands in *class* |
| *nmand* | describe *command* |

## ting your Program

| | |
|---|---|
| *st* | start your program with *arglist* |
| | start your program with current argument list |
| <*inf* >*outf* | start your program with input, output redirected |
| | kill running program |
| | use *dev* as stdin and stdout for next **run** |
| *arglist* | specify *arglist* for next **run** |
| | specify empty argument list |
| gs | display argument list |
| | show all environment variables |
| *var* | show value of environment variable *var* |
| *var string* | set environment variable *var* |
| v *var* | remove *var* from environment |

## Commands

| | |
|---|---|
| | change working directory to *dir* |
| | Print working directory |
| | call "**make**" |
| *nd* | execute arbitrary shell command string |

nd optional arguments . . . show one or more arguments

## Breakpoints and Watchpoints

| | |
|---|---|
| break [*file*:]*line*<br>b [*file*:]*line* | set breakpoint at *line* number [in *file*]<br>eg: **break main.c:37** |
| break [*file*:]*func* | set breakpoint at *func* [in *file*] |
| break +*offset*<br>break -*offset* | set break at *offset* lines from current stop |
| break **addr* | set breakpoint at address *addr* |
| break | set breakpoint at next instruction |
| break . . . if *expr* | break conditionally on nonzero *expr* |
| cond *n* [*expr*] | new conditional expression on breakpoint *n*; make unconditional if no *expr* |
| tbreak . . . | temporary break; disable when reached |
| rbreak *regex* | break on all functions matching *regex* |
| watch *expr* | set a watchpoint for expression *expr* |
| catch *event* | break at *event*, which may be **catch**, **throw**, **exec**, **fork**, **vfork**, **load**, or **unload**. |
| info break | show defined breakpoints |
| info watch | show defined watchpoints |
| clear | delete breakpoints at next instruction |
| clear [*file*:]*fun* | delete breakpoints at entry to *fun*() |
| clear [*file*:]*line* | delete breakpoints on source line |
| delete [*n*] | delete breakpoints [or breakpoint *n*] |
| disable [*n*] | disable breakpoints [or breakpoint *n*] |
| enable [*n*] | enable breakpoints [or breakpoint *n*] |
| enable once [*n*] | enable breakpoints [or breakpoint *n*]; disable again when reached |
| enable del [*n*] | enable breakpoints [or breakpoint *n*]; delete when reached |
| ignore *n count* | ignore breakpoint *n*, *count* times |
| commands *n*<br>[silent]<br>*command-list* | execute GDB *command-list* every time breakpoint *n* is reached. [**silent** suppresses default display] |
| end | end of *command-list* |

## Program Stack

| | |
|---|---|
| backtrace [*n*]<br>bt [*n*] | print trace of all frames in stack; or of *n* frames—innermost if *n*>0, outermost if *n*<0 |
| frame [*n*] | select frame number *n* or frame at address *n*; if no *n*, display current frame |
| up *n* | select frame *n* frames up |
| down *n* | select frame *n* frames down |
| info frame [*addr*] | describe selected frame, or frame at *addr* |
| info args | arguments of selected frame |
| info locals | local variables of selected frame |
| info reg [*rn*]. . . | register values [for regs *rn*] in selected frame; **all-reg** includes floating point |
| info all-reg [*rn*] | |

## Execution Control

| | |
|---|---|
| continue [*count*]<br>c [*count*] | continue running; if *count* specified, ignore this breakpoint next *count* times |
| step [*count*]<br>s [*count*] | execute until another line reached; repeat *count* times if specified |
| stepi [*count*]<br>si [*count*] | step by machine instructions rather than source lines |
| next [*count*]<br>n [*count*] | execute next line, including any function calls |
| nexti [*count*]<br>ni [*count*] | next machine instruction rather than source line |
| until [*location*] | run until next instruction (or *location*) |
| finish | run until selected stack frame returns |
| return [*expr*] | pop selected stack frame without executing [setting return value] |
| signal *num* | resume execution with signal *s* (none if 0) |
| jump *line*<br>jump **address* | resume execution at specified *line* number or *address* |
| set var=*expr* | evaluate *expr* without displaying it; use for altering program variables |

## Display

| | |
|---|---|
| print [/*f*] [*expr*]<br>p [/*f*] [*expr*] | show value of *expr* [or last value **$**] according to format *f*: |
| x | hexadecimal |
| d | signed decimal |
| u | unsigned decimal |
| o | octal |
| t | binary |
| a | address, absolute and relative |
| c | character |
| f | floating point |
| call [/*f*] *expr* | like **print** but does not display **void** |
| x [/*Nuf*] *expr* | examine memory at address *expr*; optional format spec follows slash |
| *N* | count of how many units to display |
| *u* | unit size; one of<br>**b** individual bytes<br>**h** halfwords (two bytes)<br>**w** words (four bytes)<br>**g** giant words (eight bytes) |
| *f* | printing format. Any **print** format, or<br>**s** null-terminated string<br>**i** machine instructions |
| disassem [*addr*] | display memory as machine instructions |

## Automatic Display

| | |
|---|---|
| display [/*f*] *expr* | show value of *expr* each time program stops [according to format *f*] |
| display | display all enabled expressions on list |
| undisplay *n* | remove number(s) *n* from list of automatically displayed expressions |
| disable disp *n* | disable display for expression(s) number *n* |
| enable disp *n* | enable display for expression(s) number *n* |
| info display | numbered list of display expressions |

## ssions

| | |
|---|---|
| | an expression in C, C++, or Modula-2 (including function calls), or: |
| | an array of *len* elements beginning at *addr* |
| | a variable or function *nm* defined in *file* |
| *ldr* | read memory at *addr* as specified *type* |
| | most recent displayed value |
| | *n*th displayed value |
| | displayed value previous to $ |
| | *n*th displayed value back from $ |
| | last address examined with **x** |
| | value at address $_ |
| | convenience variable; assign any value |
| **ues** [*n*] | show last 10 values [or surrounding $*n*] |
| **v** | display all convenience variables |

## ol Table

| | |
|---|---|
| **ress** *s* | show where symbol *s* is stored |
| **c** [*regex*] | show names, types of defined functions (all, or matching *regex*) |
| [*regex*] | show names, types of global variables (all, or matching *regex*) |
| *expr*] *xpr*] | show data type of *expr* [or **$**] without evaluating; **ptype** gives more detail |
| *pe* | describe type, struct, union, or enum |

## Scripts

| | |
|---|---|
| *cript* | read, execute GDB commands from file *script* |
| *md* *mand-list* | create new GDB command *cmd*; execute script defined by *command-list* |
| | end of *command-list* |
| *cmd* *-text* | create online documentation for new GDB command *cmd* |
| | end of *help-text* |

## s

| | |
|---|---|
| *ignal act* | specify GDB actions for *signal*: |
| | announce signal |
| **t** | be silent for signal |
| | halt execution on signal |
| | do not halt execution |
| | allow your program to handle signal |
| | do not allow your program to see signal |
| **nals** | show table of signals, GDB action for each |

## ging Targets

| | |
|---|---|
| *ype param* | connect to target machine, process, or file |
| **get** | display available targets |
| *aram* | connect to another process |
| | release target from GDB control |

## Controlling GDB

| | |
|---|---|
| **set** *param value* | set one of GDB's internal parameters |
| **show** *param* | display current setting of parameter |

Parameters understood by **set** and **show**:

| | |
|---|---|
| **complaint** *limit* | number of messages on unusual symbols |
| **confirm** *on/off* | enable or disable cautionary queries |
| **editing** *on/off* | control **readline** command-line editing |
| **height** *lpp* | number of lines before pause in display |
| **language** *lang* | Language for GDB expressions (**auto**, **c** or **modula-2**) |
| **listsize** *n* | number of lines shown by **list** |
| **prompt** *str* | use *str* as GDB prompt |
| **radix** *base* | octal, decimal, or hex number representation |
| **verbose** *on/off* | control messages when loading symbols |
| **width** *cpl* | number of characters before line folded |
| **write** *on/off* | Allow or forbid patching binary, core files (when reopened with **exec** or **core**) |
| **history ...** **h ...** | groups with the following options: |
| **h exp** *off/on* | disable/enable **readline** history expansion |
| **h file** *filename* | file for recording GDB command history |
| **h size** *size* | number of commands kept in history list |
| **h save** *off/on* | control use of external file for command history |
| **print ...** **p ...** | groups with the following options: |
| **p address** *on/off* | print memory addresses in stacks, values |
| **p array** *off/on* | compact or attractive format for arrays |
| **p demangl** *on/off* | source (demangled) or internal form for C++ symbols |
| **p asm-dem** *on/off* | demangle C++ symbols in machine-instruction output |
| **p elements** *limit* | number of array elements to display |
| **p object** *on/off* | print C++ derived types for objects |
| **p pretty** *off/on* | struct display: compact or indented |
| **p union** *on/off* | display of union members |
| **p vtbl** *off/on* | display of C++ virtual function tables |

| | |
|---|---|
| **show commands** | show last 10 commands |
| **show commands** *n* | show 10 commands around number *n* |
| **show commands +** | show next 10 commands |

## Working Files

| | |
|---|---|
| **file** [*file*] | use *file* for both symbols and executable; with no arg, discard both |
| **core** [*file*] | read *file* as coredump; or discard |
| **exec** [*file*] | use *file* as executable only; or discard |
| **symbol** [*file*] | use symbol table from *file*; or discard |
| **load** *file* | dynamically link *file* and add its symbols |
| **add-sym** *file addr* | read additional symbols from *file*, dynamically loaded at *addr* |
| **info files** | display working files and targets in use |
| **path** *dirs* | add *dirs* to front of path searched for executable and symbol files |
| **show path** | display executable and symbol file path |
| **info share** | list names of shared libraries currently loaded |

## Source Files

| | |
|---|---|
| **dir** *names* | add directory *names* to front of source path |
| **dir** | clear source path |
| **show dir** | show current source path |
| **list** | show next ten lines of source |
| **list -** | show previous ten lines |
| **list** *lines* | display source surrounding *lines*, specified as: |
| [*file*:]*num* | line number [in named file] |
| [*file*:]*function* | beginning of function [in named file] |
| **+***off* | *off* lines after last printed |
| **-***off* | *off* lines previous to last printed |
| ******address* | line containing *address* |
| **list** *f*,*l* | from line *f* to line *l* |
| **info line** *num* | show starting, ending addresses of compiled code for source line *num* |
| **info source** | show name of current source file |
| **info sources** | list all source files in use |
| **forw** *regex* | search following source lines for *regex* |
| **rev** *regex* | search preceding source lines for *regex* |

## GDB under GNU Emacs

| | |
|---|---|
| **M-x gdb** | run GDB under Emacs |
| **C-h m** | describe GDB mode |
| **M-s** | step one line (**step**) |
| **M-n** | next line (**next**) |
| **M-i** | step one instruction (**stepi**) |
| **C-c C-f** | finish current stack frame (**finish**) |
| **M-c** | continue (**cont**) |
| **M-u** | up *arg* frames (**up**) |
| **M-d** | down *arg* frames (**down**) |
| **C-x &** | copy number from point, insert at end |
| **C-x SPC** | (in source file) set break at point |

## GDB License

| | |
|---|---|
| **show copying** | Display GNU General Public License |
| **show warranty** | There is NO WARRANTY for GDB. Display full no-warranty statement. |