GDB QUICK REFERENCE

**Essential Commands**

- **gdb program** [core] ... debug program (using core dump core)
- **gdb program function** ... set break at function in file
- **run** [argv] ... start your program with argv
- **bt** ... show backtrace
- **p expr** ... display the value of an expression
- **c** ... continue running your program
- **n** ... next line, stepping over function calls
- **s** ... next line, stepping into function calls

**Starting GDB**

- **gdb** ... start GDB, with no debugging files
- **gdb program** ... start debugging program
- **gdb program core** ... start debugging core produced by program
- **gdb --help** ... describe command line options

**Stopping GDB**

- **quit** ... exit GDB; also q or EXIT (eg C-c)
- **INTERRUPT** ... (eg C-c) terminate current command, or send to running process

**Getting Help**

- **help** ... list classes of commands
- **help class** ... online descriptions for commands in class
- **help command** ... describe command

**Executing your Program**

- **run argv** ... start your program with argv
- **run** ... start your program with current arguments
- **run ... <inf> ousf** ... start your program with input, output redirected
- **kill** ... kill running program
- **tty dev** ... use dev as stdin and stdout for next run
- **set args argv** ... set argv once for next run
- **set args** ... specify empty argument list
- **show argv** ... display argument list
- **show env** ... show all environment variables
- **show env var** ... show value of environment variable var
- **set env var string** ... set environment variable var
- **unset env var** ... remove var from environment

**Shell Commands**

- **cd dir** ... change working directory to dir
- **pwd** ... Print working directory
- **make** ... call 'make'
- **shell cmd** ... execute arbitrary shell command string

**Breakpoints and Watchpoints**

- `break [file]:line` ... set breakpoint at line number
- `break main.c:37` ... set breakpoint at main.c:37
- `break [file]:func` ... set breakpoint at function
- `break +offset` ... set break at offset lines from current stop
- `break -offset` ... break conditionally on numbers even
- `break ... if expr` ... new conditional expression on breakpoint
- `break ... if expr` ... set breakpoint conditionally on numbers even
- `cond n [expr]` ... new conditional expression on breakpoint
- `cond n [expr]` ... set breakpoint conditionally on numbers even
- ` breakpoints [expr]` ... break on all functions matching regex
- `watch expr` ... set a watchpoint for expression
- `catch x` ... break at C++ handler for exception x
- `info breakpoints` ... show defined breakpoints
- `info watchpoints` ... show defined watchpoints
- `clear` ... delete breakpoints at next instruction
- `clear [file]:line` ... delete breakpoints on source line
- `delete [breakpoint number]` ... delete breakpoints
- `disable [breakpoint number]` ... disable breakpoints
- `enable [breakpoint number]` ... enable breakpoints
- `ignore n count` ... ignore breakpoint n, count times
- `commands [silent]` ... execute GDB commands-list every time breakpoint is reached
- `silent` ... suspend display of breakpoints
- `command-list` ... end of command-list

**Program Stack**

- `backtrace [file]` ... print trace of all frames in stack; or of n frames
- `bt [N]` ... print trace of all frames in stack; or of n frames
- `frame [n]` ... select frame number n or frame at address n; if 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 at addr
- `info args` ... arguments of selected frame
- `info locals` ... local variables of selected frame
- `info reg [m]` ... registers values for regs in selected frame
- `info all-reg [m]` ... frame; all-reg includes 'printing point'
- `info catch` ... exception handlers active in selected frame

**Execution Control**

- `continue [count]` ... continue running; if count specified, ignore this breakpoint next count times
- `c [count]` ... continue running; if count specified, ignore this breakpoint next count times
- `step [count]` ... execute until next line reached; repeat count times if specified
- `s [count]` ... step by machine instructions rather than source lines
- `si` ... continue next line, including any function calls
- `next [count]` ... continue next line, including any function calls
- `n [count]` ... continue next line, including any function calls
- `ni [count]` ... continue next line, including any function calls
- `until [location]` ... run until next instruction (or location)
- `finish` ... run until selected stack frame returns
- `return [expr]` ... pop selected stack frame
- `executing [context [return_value]]` ... resume execution without displaying it; use for showing program variables

**Display**

- `print [file]` ... show value of expr or list value $
- `p [file] [expr]` ... according to format $f$
- `x [file] [expr]` ... hex
danced
- `d [file] [expr]` ... signed decimal
- `u [file] [expr]` ... unsigned decimal
- `o [file] [expr]` ... octal
- `t [file] [expr]` ... terminate
- `a [file] [expr]` ... address, absolute and relative
c
- `c [file] [expr]` ... character
- `f [file] [expr]` ... floating point
- `call [file]` ... use print but does not display void
- `call [file]` ... examine memory at address expr; optional format spec followed by
- `x [file] [expr]` ... count of how many until to display
- `n` ... unit size one of
- `t` ... integer type, long or short
- `f` ... floating point, double or long double
- `a` ... string variable
- `i` ... machine instructions
- `diagnose [adb]` ... display memory as machine instructions

**Automatic Display**

- `display [file] [expr]` ... show value of expr each time program
- `scope [according to format $f$]` ... display at enabled expressions on list
- `undisplay n` ... remove number(s) n from list of
- `disable disp n` ... automatically displayed expressions
- `enable disp [number]` ... disable display for expression(s) number n
- `info display` ... enable display for expression(s) number n
- `numbered list of display expressions`... numbered list of display expressions
Expressions

expr
address
file:nam
{type} add
$[n]
$++
$=  value
show values [n]
show Command

table
Symbol Table
info address
info func [ref]
info var [ref]
what is
ptype [type]
GDB Scripts
source script
define end command-list
end
document end help-list
Signals
handle signal act
print
reprint
stop
next
debug
info signals
Debugging Targets

target
help target
attach
detach
Controlling GDB

set param value
show param
Parameters
complain limits
confirm on/off
 editing on/off
height top
language lang
listsize
prompt str
track base
verbose on/off
width opt
write on/off
history ...
print ...
show commands n
show commands +
Working Files

file [file]
core [file]
exec [file]
symbol [file]
load file
add-sym file addr
info files
info path
info share
Source Files

dir names
dir
show dir
list
list -
list lines
list files

GDB under GNU Emacs

gdb

GDB License

copying
warranty

This License applies to any file which contains a notice placed
by the copyright holder saying it may be distributed under the
terms of this License. There is absolutely no warranty for GDB.