GDB QUICK REFERENCE

**Essential Commands**

- `gdb program [core]`    start program with core dump
- `b [file:]func`          set breakpoint at function [in file]
- `run [arglist]`          start your program with arglist
- `bt`                    show backtrace
- `p expr`                display value of expression
- `c`                     continue running your program
- `n`                     step into next line
- `s`                     step over next function

**Starting GDB**

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

**Stopping GDB**

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

**Getting Help**

- `help`                  list classes of commands
- `help class`            one-line descriptions for commands in class
- `help command`          describe command

**Executing your Program**

- `run arglist`           start your program with arglist
- `run`                   start your program with current argument list
- `run ...<inf>/outf`     start your program with input, output redirected
- `kill`                  kill running program
- `tty dev`               use dev as stdin and stdout for next run
- `set args arglist`      specify arglist for next run
- `set args`              specify empty argument list
- `show args`             display argument list
- `show env`              show all environment variables
- `show env var`          show value of environment variable
- `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

[ ] surround optional arguments  . . . show one or more arguments

**Breakpoints and Watchpoints**

- `break [file:] line`    set breakpoint at line number [in file]
- `break [file:] line`    eg: `break main.c:37`
- `break [file:] func`    set breakpoint at function [in file]
- `break +offset`         set break at offset lines from current stop
- `break -offset`         set break at previous offset
- `break *addr`           set breakpoint at address addr
- `break ... if expr`     conditional breakpoint if expr
- `break ...`             temporary break; disable when reached
- `disable [n]`           disable breakpoints [or breakpoint n]
- `enable [n]`            enable breakpoints [or breakpoint n]
- `enable once [n]`       disable again if breakpoint n; again when reached
- `enable del [n]`        enable breakpoints [or breakpoint n]; delete when reached

**Execution Control**

- `continue [count]`      continue running; if count specified, ignore this breakpoint next count times
- `c [count]`             continue running; count times if specified
- `step [count]`          execute until another line reached; repeat count times if specified
- `n [count]`             step by machine instructions rather than source line
- `s [count]`             step over machine instructions
- `next [count]`          execute next line, including any function calls
- `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]`         return selected stack frame without executing [setting return value]
- `signal num`            resume execution with signal (none if 0)
- `jump line`            resume execution at specified line number or address
- `jump addr expression`  evaluate expr without displaying value; use for altering program variables

**Display**

- `print [file:] expr`    show value of expr [or last value $]
- `p [file:] expr`        according to format $f$
- `x`                     hex
- `d`                     decimal
- `u`                     unsigned decimal
- `o`                     octal
- `t`                     binary
- `a`                     address, absolute and relative
- `c`                     character
- `f`                     floating point
- `call [file:] expr`     like `print` but does not display `void`
- `x [Nuf] expr`          examine memory at address expr; optional format spec follows `sh`:
  - `N` count of how many units to display unit size, one of:
    - `b` individual bytes
    - `h` halfwords (two bytes)
    - `w` words (four bytes)
    - `g` groups of words (eight bytes)
    - `f` format printing format, Any `print` format, or
    - `i` machine instructions
- `disassemble [addr]`    display memory as machine instructions

**Automatic Display**

- `display [file:] 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
Expressions

expr
an expression in C, C++, or Modula-2
(including function calls), or:
add[ten]
an array of ten elements beginning at addr
file: nm
a variable or function nm defined in file
type addr
read memory at addr as specified type
$n
s'th displayed value
$s
s'th displayed value
$n
s'th displayed value
$n
s'th displayed value or previously s
$--
value at address $.
$var
convenience variable; assign any value

show values [$]
show last 10 values (or surrounding 8$)
show conv
display all convenience variables

Symbol Table

info address s
show where symbol s is stored
info func [reg]
show names, types of defined functions
(all, or matching reg)
info var [reg]
show names, types of global variables
(all, or matching reg)
what is [expr]
show data type of [expr or $] without
examining; ptype gives more detail
ptype [expr]
describe type, struct, union, or enum

GDB Scripts

source script
read, execute GDB commands from file script
define cmd
create new GDB command cmd; execute
script defined by cmd
end command-list
end of command-list
document cmd
create online documentation for new GDB
command cmd
end end of help-text

Signals

handle signal act
specify GDB actions for signal
print
announce signal
no print
be silent for signal
stop
halt execution on signal
nostop
do not halt execution
pass
allow your program to see signal
no pass
do not allow your program to see signal
info signals
show table of signals, GDB action for each

Debugging Targets

target type param
connect to target machine, process, or file
help target
display available targets
attach param
connect to another process
detach
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:
complain km
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
modula2)
listsize n
number of lines shown by list
prompt str
use str as GDB prompt
radix base
octal, decimal, or hex number
representation
verbatim 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...
groups with the following options:
print...
groups with the following options:
p....
p address on/off
print memory addresses in stacks, values
p array on/off
compact or attractive format for arrays
p demangle 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 on/off
struct display: compact or indented
p union on/off
display union members
p vtable on/off
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
to target; or discard
add-sym file addr
dynamically loaded at addr
display working files and targets in use
info files
display executable and symbol file path
path dirs
display directories and symbol file path
show path
display executable and symbol file path

Source Files

dir names
add directory names to front of source
path
dir
clear source path
show dir
current source path
list
display next ten lines of source
list -
display previous ten lines
list lines
display source surrounding lines, specified
as
[fi:] num
line number in named file
[fi:] 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 file / 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
rew 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-x
step one line (step)
M-x n
next line (next)
M-x i
step one instruction (step)
C-c C-f
finish current stack frame (finish)
M-x c
continue (cont)
M-x u
up arg frames (up)
M-x d
down arg frames (down)
C-x &
copy number from point, insert at end
C-x SP
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.

Roland Pesch (pesch@cygnus.com)

The author assumes no responsibility for any errors on this card.
This card may be freely distributed under the terms of the GNU
General Public License.
Please contribute to development of this card by annotating it.
GDB itself is free software; you are welcome to distribute copies of
it under the terms of the GNU General Public License. There is
absolutely no warranty for GDB.