GDB QUICK REFERENCE

Essential Commands

gdb program [core] debug program [using coredump core]
b [file]:func set breakpoint at function [in file]
run [arglist] start your program [with arglist]
bk trace: display program stack
p expr display the value of an expression
n next line, stepping over function calls
s next line, stepping into function calls

Starting GDB

gdb program begin debugging program
gdb program core debug core dumped program produced by

gdb --help describe command line options

Stopping GDB

quit exit GDB; also q or EOF (eg C-d)
INTERUPT (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> >out 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 var
set env var 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 [in file]
b [file]:[line] eg: break main.c:37
break [file]:func set breakpoint at func [in file]
break +offset set breakpoint at offset lines from current stop
break -offset
break *addr set breakpoint at address addr
break set breakpoint at next instruction
break ... if expr conditionally set breakpoint if expr
cond n [expr] set new conditional expression on breakpoint n; make unconditional if no expr
thbreak temporary break; disable when reached
tbreak regex set break on all functions matching regex
watch expr set a watchpoint for expression expr
watch event set a watchpoint for event expr
info break show defined breakpoints
info watch show defined watchpoints
delete disable breakpoints at next instruction
delete [file]:fun disable breakpoints at entry to func
/delete [file]:line disable breakpoints on source line
delete [n] disable 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 execute GDB command-list every time breakpoint n is reached
command-list suppresses default display
end of command-list

Program Stack

backtrace [n] print trace of all frames in stack; or of n frames—innermost if >0, outermost if <0
frame [n] select frame number n or frame at address n; if n=0, 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 [r]... register values [for reg r] in selected frame; all-reg includes floating point
info all-reg [r]... frame; all-reg includes floating point

Execution Control

continue [count] continue running; if count specified, ignore this breakpoint next count times
step [count] execute until another line reached; repeat count times if specified
si [count] step by machine instructions rather than source lines
next [count] execute next line, including any function calls
nexti [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 pending return value
signal num resume execution with signal s (none if 0)
jump [line] resume execution at specified line number
jump +address evaluate expr without displaying it; use for altering program variables

Display

print [/f] [expr] show value of expr [or last value $] according to format f
p [/f] [expr] hexadecimal
x d signed decimal
u o unsigned decimal
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 addr
N w count of how many units to display
u unit size; one of
b individual bytes
h halfwords (two bytes)
w words (four bytes)
g giant words (eight bytes)
f printing format. Any print format, or
a null-terminated string
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(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

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

1998 Free Software Foundation, Inc.  Permissions on back
Expressions
expr
an expression in C, C++, or Modula-2
(including function calls), or:
address
an array of ten elements beginning at
address
file: num
a variable or function name defined in file
{type} addr
read memory at addr as specified type
$n
nth displayed value
$\$n
nth displayed value previous to $n
$\n
value at address $n
$var
variable name; assign any value
show values [n]
show last 10 values or surrounding $n
show conv
display all convenience variables

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

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

Signals
handle signal act
specify GDB actions for signal:
print announce signal
noprint be silent for signal
stop halt execution on signal
nosp
halt execution on signal
pass allow your program to handle signal
nopass 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
display current setting of parameter
show param
Parameters understood by set and show:
-complain limit
number of messages on unusual symbols
-confirm on/off
enable or disable cautionary queries
-editing on/off
control readline command-line editing
-height n
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
-propr str
use str as GDB prompt
-radix base
octal, decimal, or hex number
-representation
-value at address $n
-conv
convert variable; assign any value

Source Files
dir names
add directory names to front of source
-path
list
show next ten lines of source
show dir
show current source path
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
-t
off
off lines after last printed
-off
off lines previous to last printed
-address
line containing address
list /l
t from line / 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
C-x C-f
step one line (step)
C-x C-n
next line (next)
C-x C-j
step one instruction (stepi)
C-c C-f
finish current stack frame (finish)
C-c C-c
continue (cont)
C-u
up arg frames (up)
C-d
down arg frames (down)
C-x &
copy number from point, insert at end
C-x C-F
(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.

Copyright © 1991, ’92, ’93, ’95 Free Software Foundation, Inc.
Richard M. Stallman
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;
Improvements can be sent to bug-gdb@gnu.org.
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.