GDB 4.5 QUICK REFERENCE

HP WDB Version 4.5 for HP-UX (http://www.hp.com/go/wdb/)

Essential Commands

gdb *program* [*core*]	debug program [using *coredump core*]
b [*file*:]*function*	set breakpoint at *function* [in *file*]
run [*arglist*]	start your program [with *arglist*]
bt *backtrace*:	display program stack
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*	begin debugging *program*
gdb *program core*	debug coredump core 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 *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
b [*file*:]*line*	[in file] e.g.: break main.c:37
break [*file*:]*func*	set breakpoint at *func* [in *file*]
break +*offset*	
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*. Use *(ptr_type) address_literal for hardware watchpoint
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 [*file*:]*fun*	delete breakpoints at next instruction
clear	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* [*silent*]	execute GDB command-list
command-list	every time breakpoint n is reached. [silent suppresses default display]
end	end of command-list

Program Stack

backtrace [*n*]	print trace of all frames in stack; or of n frames- innermost if n>0, outermost if n<0
bt [*n*]	
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*
info all-reg [*rn*]	includes floating point

Execution Control

continue[*count*]	continue running; if count specified,
c [*count*]	ignore this breakpoint next count times
step [*count*]	execute until another line
s [*count*]	reached; repeat count times if specified
stepi [*count*]	step by machine instructions rather
si [*count*]	than source lines
next [*count*]	execute next line, including
n [*count*]	any function calls
nexti [*count*]	next machine instruction
ni [*count*]	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)
go *line*	set $pc to a location and stop
go *address*	with a temporary breakpoint
set var=*expr*	evaluate expr without displaying it. Use for altering program variables

Display

print [*/f*][*expr*]	show value of expr [or last
p [*/f*] [*expr*] *f:*	value $] according to format
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
	b individual bytes
	h halfwords (two bytes)
	w words (four bytes)
	g giant words (eight bytes)
	f printing format. Any print format, or
	s null-terminated string
	i machine instructions
disassem [*addr1 [addr2]*]	display memory as machine instructions

Threads

info threads	display information on current threads
thread *n*	switch to the context of thread-id *n*
thread disable *n*	disable thread with thread id n
thread disable all	disable all threads
thread enable n	enable thread with thread id *n*
thread enable all	enable all threads

Expressions

expr	an expression in C, C++, or Modula-2 (including function calls), or:
addr@len	an array of *len* elements beginning at *addr*
file::nm	a variable or function *nm* defined in file
{type}addr	read memory at *addr* as specified type
$	most recent displayed value
$n	nth displayed value
$$	displayed value previous to $
$$n	nth displayed value back from $
$	last address examined with x
$	value at address $
$var	convenience variable; 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*]	
ptype [*expr*]	show data type of expr [or $] without evaluating; ptype gives more detail
ptype *type*	describe type, struct, union, or enum

GDB Scripts

source *script*	read, execute GDB commands from file script
define *cmd* **commandlist**	create new GDB command cmd; execute script defined by command-list
end	end of command-list
document *cmd* **help-text**	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
noprint	be silent for signal
stop	halt execution on signal
nostop	do not halt execution
pass	allow your program to handle signal
nopass	do not allow your program to see signal
info *signals*	show table of signals and GDB action

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

Runtime Checking

heap-check *on/off*| **<*option*>| <*option*>**	
<*num*>	turn on heap checking
info leaks [*leaks.out*]	produce a memory leak report
info heaps [*heap.out*]	produce a heap allocations report

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
history…	groups with the following options:
h…	
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
h save *off/on*	control use of external file for command history
print…	groups with the following
p…	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*	lines after last printed
-*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 nowarranty statement.

[ ] surround optional arguments … show one or more arguments