GDB 5.9 QUICK REFERENCE

HP WDB Version 5.9 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 <*count*>	display program stack (backtrace)
p *expr*	display the value of an expression
c	continue running your program
[n / s]	next line, or step over into function calls

Starting GDB

gdb	start GDB, with no debugging files
gdb *program* [core]	debug *program* [using *coredump core*]
gdb program *pid*	debug existing applications with pid *pid*
gdb --help	describe command line options
gdb -crashdebug	invokes GDB before program aborts

Stopping GDB

[quit / exit]	quit GDB; also q or EOF (eg Ctrl-d)
INTERRUPT	(eg Ctrl-c) terminate current command, or send to running process

Getting Help

help	list classes of commands
help *class*	short descriptions for commands in *class*
help *command*	describe *command*
help java	list Java and JVM debugging commands
java	list Java subcommands

Executing your Program

run [*arglist*]	start your program with *arglist* or with current argument list if *arglist* is not specified
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* or empty list for next run
show args	display argument list
show envvars	show all environment variables
show env *var*	show value of environment variable *var*
set env *var string*	set environment variable *var* to *string*
unset env *var*	remove *var* from environment

Shell Commands

cd dir, pwd, and **make**	supported shell commands in gdb
shell *cmd*	execute arbitrary shell command string

Breakpoints and Watchpoints

break [*file*:]*line* *or* **b [*file*:]*line***	set breakpoint at *line* number [in *file*] e.g.: break main.c:37
break [*file*:]*func*	set breakpoint at *func* [in *file*]
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/line*]	delete breakpoints at the beginning of *func* [in *file*] or on a specific source *line* [in *file*]
clear	delete all breakpoints at the current line
delete [*n*]	delete breakpoints [or breakpoint *n*]
disable [*n*] or **enable [*n*]**	disable/ 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
command-list	execute GDB *command-list*
***command-list* n [*silent*]**	execute GDB *command-list* every time breakpoint *n* is reached. [*silent* suppresses default display]
watch_target *target_expr*	watch a target location
end	end of command-list

Program Stack

info module	identify load modules
backtrace [*n*] or **bt [*n*]**	print trace of all frames in stack; or of *n* frames or where [n] 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 / down] *n*	select frame *n* frames up or down
info frame [*addr*]	describe selected frame, or frame at *addr*
info [args/ locals]	arguments or local variables of selected frame
info [reg /all_reg] [*rn*]…	register values [for *regs rn or all registers*] in the selected frame. Option all_reg includes information for floating point registers too

Viewing the Execution Path Entries

info exec-path [*start_index*] [*end_index*]	lists all the local execution path entries in the current frame
info global-exec-path [*start_index*] [*end_index*]	lists all the global execution path entries for the current thread
exec-path [*up*] [*down*] [*path_index*]	select, print, and navigate through the execution paths

Execution Control

continue [*count*] or **c [*count*]**	continue running; if count specified, ignore this breakpoint next count times
step [*count*] or **s [*count*]**	execute until another line reached; repeat *count* times if specified
stepi [*count*] or **si [*count*]**	step by machine instructions source lines
next [*count*] or **n [*count*]**	execute next line, including any function calls
nexti [*count*] or **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 when executing [setting return value to expr]
signal *s*	resume execution with signal **s** (none if 0)
go [*line/*address*]	set $pc to a location and stop with a temporary breakpoint
set var=*expr*	evaluate *expr* without displaying it. Use for altering program variables

Display

[p / print] [/*f*][*expr*]	show value of *expr* [or last value $] according to format, see **help p**.
x [/*Nuf*] *expr*	examine memory at address *expr*; see **help x**.
disassem [*addr1*|*addr2*]	display memory as machine instructions

Threads

info threads [*n*]	display information on current threads [or a specific thread *n*]
thread *n*	switch to the context of thread *n*
thread disable [*n* |*all*]	disable thread with thread *n* or all
thread enable [n | all]	enable thread with thread n or all
set thread-check {[*on/off*] | [*option*] [*on/off*] | [*option*] [*num*]}	enable detection for the following advanced debugging options
[*recursive-relock*] [*on/off*]	thread attempts to acquire a non-recursive mutex that it currently holds
[*unlock-not-own*] [*on/off*]	thread attempts to unlock an un-acquired mutex/ read-write lock
[*mixed-sched-policy*] [*on/off*]	thread waits on a mutex/read-write lock, held by a thread with a different scheduling policy
[*cv-multiple-mxs*][*on/off*]	different threads non-concurrently wait on the same condition variable with different associated mutexes
[*cv-wait-no-mx*] [*on/off*]	associated mutex of a condition variable is locked and thread calls the pthread_cond_wait() routine
[*thread-exit-own-mutex*] [*on/off*]	thread terminates execution without unlocking the associated mutexes/read-write locks
[*thread-exit-no-join-detach*] [on/off]	thread has terminated execution without joining or detaching the thread
[*stack-util*] [*num*]	thread uses more than the specified % of the stack allocated to the thread
[*num-waiters*] [*num*]	number of threads waiting on a pthread object exceeds [num]
info [*mutex*|*condvar*|*rwlock*] [*n*]	lists all known mutexes, conditional variables or read write locks

Expressions

expr	and expression in C, C++, or Modula-2 (including function calls)
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
$	expression used in most recent command
$*n*	nth displayed value
$$	displayed value previous to $
$$*n*	nth displayed value back from $

$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/var] [regex]	show names, types of defined functions or types of global variables (all, or matching regex**)**
info var [regex]	show names, types of global variables (all, or matching regex)
[whatis / ptype] [expr]	show data type of expr [or $] without evaluating; ptype gives more detail
ptype type	describe type, struct, union, or enum
which *symbol*	prints the scope, file and line details of ***symbol***

GDB Input Scripts

source script	read, execute GDB commands from script
define [cmd]	create new GDB command *cmd*;
[commandlist]	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* <*args*>	specify GDB actions for signal:
print *or* **noprint**	announce signal or be silent for signal
stop *or* **nostop**	halt / do not halt execution on signal
pass *or* **nopass**	pass/ no pass of signals to program
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
set mapshared [on/off]	set the shared library loading mode in GDB

Controlling GDB

set param *value*	set one of GDB's internal parameters
show param	display current setting of 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
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[*options*] *or* **h [*options*]**	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
h save [off/on]	save /do not save command history in a file
print[*options*] *or* **p[*options*]**	groups with the following options:
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 of union members
p vtbl [on/off]	display of C++ virtual function tables
show commands [n/+]	show last 10 commands, show 10 commands around number [n], show next 10 commands [+]

Runtime Heap Checking

info corruption	Lists the potential in-block corruptions in all the freed blocks
heap-check [*option*] [*on/off*]	set heap checking options
info leaks [*leaks.out*]	produce a memory leak report
info heap [*heap.out*]	produce a heap allocations report
info heap-interval <*filename*>	create heap growth report
info heap process	high level memory usage of a process
info heap arena	high level memory usage for all arenas
info heap arena [0 |1|2|..] blocks stacks	block level and overall memory usage with stack trace where applicable.
info dangling	Display all dangling pointers and blocks which are potential sources of memory corruption
set heap-check interval <*nn*>	set incremental heap profiling
set heap-check repeat <*nn*>	set repeat cycles for incremental heap profile
set heap-check reset	reset incremental heap growth data
set heap-check header-size < no of bytes >	Set 'Header' guard for each block of the allocated memory
set heap-check footer-size < no of bytes >	Set 'Footer' guard for each block of the allocated memory

Working Files

file [*file*]	use *file* for both symbols and executable
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 search path for executable or symbol files
show paths	display executable and symbol file path
info share	lists names of shared libraries currently loaded

Core file Commands

core-file FILE	FILE as core dump to examine memory registers
packcore	create tar file for executable and core file
unpackcore	unpack tar file created with packcore
getcore	examine core file
dumpcore	generate a core file without modifying the process state
info rtti <address>	display run time type information for C++ polymorphic object

Inline Debugging

set inline debug [options]	set inline debugging preferences
[on|off]	enable inline debugging without breakpoint feature or disable inline debugging
[inline_bp_all]	enables inline debugging with the breakpoints feature for all instances of an inline function
[inline_bp_individual]	enables inline debugging with breakpoints feature for individual instances of an inline function

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 /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 or previous 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 *or* **info sources**	list the current source file or all source files in use
forw *regex* *or* **rev *regex***	search following or preceding source lines for *regex.*

GNU GDB Logging Commands

set logging file	set the current log file
set logging [on|off]	set logging on or off
set logging overwrite [on|log]	allow overwrite or append to the log file
set logging redirect [on|off]	set logging output mode

Debugging Macros

show macro [*macro-name*]	display the macro definition, source file name, and the line number**.**
expand macro [*macro-name*]	expands the macro and substitutes any parameters in the macro