GDB QUICK REFERENCE

GDB Version 4.0—Cygnus Support 1991

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`	starts GDB, with no debugging files
`gdb` *program*	begin debugging *program*
`gdb` *program core*	debug coredump *core* produced by *program*

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 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 environment`	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* `b` [*file*`:`]*line*	set breakpoint at *line* number [in *file*] eg: `break main.c:37`
`break` [*file*`:`]*function*	set breakpoint at *function* [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*
`catch` *x*	break at C++ handler for exception *x*
`info break`	show defined breakpoints
`info watch`	show defined watchpoints
`clear`	delete breakpoints at next instruction
`clear` [*file*`:`]*fun*	delete breakpoints at entry to *fun*()
`clear` [*file*`:`]*line*	delete breakpoints on source line
`delete` [*n*]	delete breakpoints *n*; [or all breakpoints]
`disable` [*n*]	disable breakpoints *n* [or all]
`enable` [*n*]	enable breakpoints *n* [or all]
`enable once` [*n*]	enable breakpoints; disable again when reached
`enable del` [*n*]	enable breakpoints; delete when reached
`ignore` *n count*	ignore breakpoint *n*, *count* times
`commands` *n* *command list*	execute GDB *command list* every time breakpoint *n* is reached
`end`	end of *command list*

Program Stack

`backtrace` [*n*] `bt` [*n*]	print trace of all frames in stack; or of *n* frames—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` *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 reg *rn*] in selected frame
`info catch`	exception handlers active in selected frame

Execution Control

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

Display

`print` [`/`*f*] *expr* `p` [`/`*f*] *expr*	show value of *expr* according to format *f*:
`x`	hexadecimal
`d`	signed decimal
`u`	unsigned decimal
`o`	octal
`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
`disassemble` [*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(s) *n* from list of automatically displayed expressions
`disable display` *n*	disable display for expression(s) number *n*
`enable display` *n*	enable display for expression(s) number *n*
`info display`	numbered list of display expressions

Expressions

expr	an expression in C or C++ (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*	*n*th displayed value
$$	displayed value previous to $
$$*n*	*n*th 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 convenience	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* without evaluating;
ptype *expr*	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*	new GDB command *cmd*, executes script
command list	defined by *command list*
end	end of *command list*
document *cmd*	new online documentation for GDB
help text	command *cmd*
end	end of *help text*

Signals

handle *signal act*	specify GDB actions when *signal* occurs:
print	announce when signal occurs
noprint	be silent when signal occurs
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, 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 expr*	set one of GDB's internal parameters
show *param*	display current setting of a GDB parameter

Parameters understood by set and show:

complaints *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
prompt *str*	use *str* as GDB prompt
radix *base*	octal, decimal, or hex number representation
verbose *on/off*	control messages when loading symbol table
width *cpl*	number of characters before line folded
history ...	(h) groups the following options:
h exp *off/on*	disable or enable readline history expansion
h file *filename*	file for recording GDB command history
h size *size*	number of commands kept in history list
h save *off/on*	control use of external file for command history
print ...	(p) groups the following options:
p address *on/off*	print memory addresses in stacks, values
p array *off/on*	compact or attractive format for arrays
p demangle *on/off*	source or internal form for C++ symbols
p asm-dem *on/off*	demangle C++ symbols in machine-instruction output
p elements *limit*	number of elements to display from an array
p object *on/off*	print C++ derived types for objects
p pretty *off/on*	struct display: compact or indented
p union *on/off*	enable or disable 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 *name*	use *file* for symbols and executable
core *name*	read *file* as coredump
exec *name*	use *file* as executable only
symbol *name*	use only symbol table from *file*
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
share [*regex*]	add symbol information for shared libraries matching *regex*, or all shared libraries
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 centered around *lines*, specified as one of:
[*file*:]*num*	line number [in named file]
[*file*:]*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 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

Roland Pesch (pesch@cygnus.com), May 1991—Revision: 1.6