

Essential Commands

<code>gdb program [core]</code>	debug <i>program</i> [using coredump <i>core</i>]
<code>b [file:]function</code>	set breakpoint at <i>function</i> [in <i>file</i>]
<code>run [arglist]</code>	start your program [with <i>arglist</i>]
<code>bt</code>	backtrace: display program stack
<code>p expr</code>	display the value of an expression
<code>c</code>	continue running your program
<code>n</code>	next line, stepping over function calls
<code>s</code>	next line, stepping into function calls

Starting GDB

<code>gdb</code>	start GDB, with no debugging files
<code>gdb program</code>	begin debugging <i>program</i>
<code>gdb program core</code>	debug coredump <i>core</i> produced by <i>program</i>
<code>gdb --help</code>	describe command line options

Stopping GDB

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

Getting Help

<code>help</code>	list classes of commands
<code>help class</code>	one-line descriptions for commands in <i>class</i>
<code>help command</code>	describe <i>command</i>

Executing your Program

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

Shell Commands

<code>cd dir</code>	change working directory to <i>dir</i>
<code>pwd</code>	Print working directory
<code>make ...</code>	call “make”
<code>shell cmd</code>	execute arbitrary shell command string

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

Breakpoints and Watchpoints

<code>break [file:]line</code>	set breakpoint at <i>line</i> number [in <i>file</i>] eg: <code>break main.c:37</code>
<code>b [file:]line</code>	
<code>break [file:]func</code>	set breakpoint at <i>func</i> [in <i>file</i>]
<code>break +offset</code>	set break at <i>offset</i> lines from current stop
<code>break -offset</code>	
<code>break *addr</code>	set breakpoint at address <i>addr</i>
<code>break</code>	set breakpoint at next instruction
<code>break ... if expr</code>	break conditionally on nonzero <i>expr</i>
<code>cond n [expr]</code>	new conditional expression on breakpoint <i>n</i> ; make unconditional if no <i>expr</i>
<code>tbreak ...</code>	temporary break; disable when reached
<code>rbreak regex</code>	break on all functions matching <i>regex</i>
<code>watch expr</code>	set a watchpoint for expression <i>expr</i>
<code>catch event</code>	break at <i>event</i> , which may be <code>catch</code> , <code>throw</code> , <code>exec</code> , <code>fork</code> , <code>vfork</code> , <code>load</code> , or <code>unload</code> .
<code>info break</code>	show defined breakpoints
<code>info watch</code>	show defined watchpoints
<code>clear</code>	delete breakpoints at next instruction
<code>clear [file:]fun</code>	delete breakpoints at entry to <i>fun</i> ()
<code>clear [file:]line</code>	delete breakpoints on source line
<code>delete [n]</code>	delete breakpoints [or breakpoint <i>n</i>]
<code>disable [n]</code>	disable breakpoints [or breakpoint <i>n</i>]
<code>enable [n]</code>	enable breakpoints [or breakpoint <i>n</i>]
<code>enable once [n]</code>	enable breakpoints [or breakpoint <i>n</i>]; disable again when reached
<code>enable del [n]</code>	enable breakpoints [or breakpoint <i>n</i>]; delete when reached
<code>ignore n count</code>	ignore breakpoint <i>n</i> , <i>count</i> times
<code>commands n [silent]</code>	execute GDB <i>command-list</i> every time breakpoint <i>n</i> is reached. [silent suppresses default display]
<code>end</code>	end of <i>command-list</i>

Program Stack

<code>backtrace [n]</code>	print trace of all frames in stack; or of <i>n</i> frames—innermost if <i>n</i> >0, outermost if <i>n</i> <0
<code>bt [n]</code>	
<code>frame [n]</code>	select frame number <i>n</i> or frame at address <i>n</i> ; if no <i>n</i> , display current frame
<code>up n</code>	select frame <i>n</i> frames up
<code>down n</code>	select frame <i>n</i> frames down
<code>info frame [addr]</code>	describe selected frame, or frame at <i>addr</i>
<code>info args</code>	arguments of selected frame
<code>info locals</code>	local variables of selected frame
<code>info reg [rn]...</code>	register values [for regs <i>rn</i>] in selected frame; <code>all-reg</code> includes floating point
<code>info all-reg [rn]</code>	

Execution

<code>continue [count]</code>	
<code>c [count]</code>	
<code>step [count]</code>	
<code>s [count]</code>	
<code>stepi [count]</code>	
<code>si [count]</code>	
<code>next [count]</code>	
<code>n [count]</code>	
<code>nexti [count]</code>	
<code>ni [count]</code>	
<code>until [location]</code>	
<code>finish</code>	
<code>return [expr]</code>	
<code>signal num</code>	
<code>jump line</code>	
<code>jump *address</code>	
<code>set var=expr</code>	

Display

<code>print [/f] [expr]</code>	
<code>p [/f] [expr]</code>	
<code>x</code>	
<code>d</code>	
<code>u</code>	
<code>o</code>	
<code>t</code>	
<code>a</code>	
<code>c</code>	
<code>f</code>	
<code>call [/f] [expr]</code>	
<code>x [/Nuf] [expr]</code>	
<code>N</code>	
<code>u</code>	
<code>f</code>	

`disassem [addr]`

Automatic

<code>display [/f]</code>	
<code>display</code>	
<code>undisplay n</code>	
<code>disable disp</code>	
<code>enable disp</code>	
<code>info display</code>	

Expressions

<i>expr</i>	an expression in C, C++, or Modula-2 (including function calls), or:
<i>addr@len</i>	an array of <i>len</i> elements beginning at <i>addr</i>
<i>file::nm</i>	a variable or function <i>nm</i> defined in <i>file</i>
<i>{type}addr</i>	read memory at <i>addr</i> as specified <i>type</i>
<i>\$</i>	most recent displayed value
<i>\$n</i>	<i>n</i> th displayed value
<i>\$\$</i>	displayed value previous to <i>\$</i>
<i>\$\$n</i>	<i>n</i> th displayed value back from <i>\$</i>
<i>\$_</i>	last address examined with <i>x</i>
<i>\$—</i>	value at address <i>\$_</i>
<i>\$var</i>	convenience variable; assign any value
<i>show values [n]</i>	show last 10 values [or surrounding <i>\$n</i>]
<i>show conv</i>	display all convenience variables

Symbol Table

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

GDB Scripts

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

Signals

<i>handle signal act</i>	specify GDB actions for <i>signal</i> :
<i>print</i>	announce signal
<i>noprint</i>	be silent for signal
<i>stop</i>	halt execution on signal
<i>nostop</i>	do not halt execution
<i>pass</i>	allow your program to handle signal
<i>nopass</i>	do not allow your program to see signal
<i>info signals</i>	show table of signals, GDB action for each

Debugging Targets

<i>target type param</i>	connect to target machine, process, or file
<i>help target</i>	display available targets
<i>attach param</i>	connect to another process
<i>detach</i>	release target from GDB control

Controlling GDB

<i>set param value</i>	set one of GDB's internal parameters
<i>show param</i>	display current setting of parameter
Parameters understood by <i>set</i> and <i>show</i> :	
<i>complaint limit</i>	number of messages on unusual symbols
<i>confirm on/off</i>	enable or disable cautionary queries
<i>editing on/off</i>	control <i>readline</i> command-line editing
<i>height lpp</i>	number of lines before pause in display
<i>language lang</i>	Language for GDB expressions (<i>auto</i> , <i>c</i> or <i>modula-2</i>)
<i>listsize n</i>	number of lines shown by <i>list</i>
<i>prompt str</i>	use <i>str</i> as GDB prompt
<i>radix base</i>	octal, decimal, or hex number representation
<i>verbose on/off</i>	control messages when loading symbols
<i>width cpl</i>	number of characters before line folded
<i>write on/off</i>	Allow or forbid patching binary, core files (when reopened with <i>exec</i> or <i>core</i>)
<i>history ...</i>	groups with the following options:
<i>h ...</i>	
<i>h exp off/on</i>	disable/enable <i>readline</i> history expansion
<i>h file filename</i>	file for recording GDB command history
<i>h size size</i>	number of commands kept in history list
<i>h save off/on</i>	control use of external file for command history
<i>print ...</i>	groups with the following options:
<i>p ...</i>	
<i>p address on/off</i>	print memory addresses in stacks, values
<i>p array off/on</i>	compact or attractive format for arrays
<i>p demangl on/off</i>	source (demangled) or internal form for C++ symbols
<i>p asm-dem on/off</i>	demangle C++ symbols in machine-instruction output
<i>p elements limit</i>	number of array elements to display
<i>p object on/off</i>	print C++ derived types for objects
<i>p pretty off/on</i>	struct display: compact or indented
<i>p union on/off</i>	display of union members
<i>p vtbl off/on</i>	display of C++ virtual function tables
<i>show commands</i>	show last 10 commands
<i>show commands n</i>	show 10 commands around number <i>n</i>
<i>show commands +</i>	show next 10 commands

Working Files

<i>file [file]</i>	use <i>file</i> for both symbols and executable; with no arg, discard both
<i>core [file]</i>	read <i>file</i> as coredump; or discard
<i>exec [file]</i>	use <i>file</i> as executable only; or discard
<i>symbol [file]</i>	use symbol table from <i>file</i> ; or discard
<i>load file</i>	dynamically link <i>file</i> and add its symbols
<i>add-sym file addr</i>	read additional symbols from <i>file</i> ; Please contribute to development of dynamically loaded at <i>addr</i>
<i>info files</i>	display working files and targets in use
<i>path dirs</i>	add <i>dirs</i> to front of path searched for executable and symbol files
<i>show path</i>	display executable and symbol file path
<i>info share</i>	list names of shared libraries currently loaded

Source Files

<i>dir names</i>	
<i>dir</i>	
<i>show dir</i>	
<i>list</i>	
<i>list -</i>	
<i>list lines</i>	
<i>[file:]num</i>	
<i>[file:]function</i>	
<i>+off</i>	
<i>-off</i>	
<i>*address</i>	
<i>list f,l</i>	
<i>info line num</i>	
<i>info source</i>	
<i>info sources</i>	
<i>forw regex</i>	
<i>rev regex</i>	
<i>GDB und</i>	
<i>M-x gdb</i>	
<i>C-h m</i>	
<i>M-s</i>	
<i>M-n</i>	
<i>M-i</i>	
<i>C-c C-f</i>	
<i>M-c</i>	
<i>M-u</i>	
<i>M-d</i>	
<i>C-x &</i>	
<i>C-x SPC</i>	
<i>GDB Lice</i>	
<i>show copying</i>	
<i>show warrant</i>	

GDB License

show copying
show warrant

Copyright (C)

The author a

This card may
General Public
Please contribute to development of

GDB itself is f
it under the te
absolutely no v