

Essential Commands

<code>gdb program [core]</code>	debug <i>program</i> [using <code>coredump core</code>]
<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 <code>coredump core</code> 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>]
<code>b [file:]line</code>	eg: <code>break main.c:37</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] command-list</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>stepl [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] [e]</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] [exp]</code>	
<code>x [f/Nuf] [exp]</code>	
<code>N</code>	
<code>u</code>	
<code>f</code>	

`disassem [ad]`

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</i> } <i>addr</i>	read memory at <i>addr</i> as specified <i>type</i>
\$	most recent displayed value
\$ <i>n</i>	<i>n</i> th displayed value
\$\$	displayed value previous to \$
\$\$ <i>n</i>	<i>n</i> th displayed value back from \$
\$_	last address examined with x
\$_	value at address \$_
\$var	convenience variable; assign any value
show values [<i>n</i>]	show last 10 values [or surrounding \$ <i>n</i>]
show conv	display all convenience variables

Symbol Table

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

GDB Scripts

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

Signals

handle <i>signal act</i>	specify GDB actions for <i>signal</i> :
print	announce signal
noprnt	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, GDB action for each

Debugging Targets

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

Controlling GDB

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

Working Files

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

dir <i>names</i>	
dir	
show dir	
list	
list -	
list <i>lines</i>	
[<i>file:</i>] <i>num</i>	
[<i>file:</i>] <i>functi</i>	
+ <i>off</i>	
- <i>off</i>	
* <i>address</i>	
list <i>f, l</i>	
info line <i>nu</i>	
info source	
info sources	
forw <i>regex</i>	
rev <i>regex</i>	

GDB und

M-x gdb
C-h m
M-s
M-n
M-i
C-c C-f
M-c
M-u
M-d
C-x &
C-x SPC

GDB Lice

show copying
show warrant

Copyright (C)

The author a

This card may
General Public
development of

GDB itself is f
it under the te
absolutely no v