

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

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

Breakpoints and Watchpoints

break [*file:*]*line* set breakpoint at *line* number [in *file*]
eg: **break main.c:37**
b [*file:*]*line*
break [*file:*]*func* set breakpoint at *func* [in *file*]
break +offset set break at *offset* lines from current stop
break -offset
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 *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 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 [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* execute GDB *command-list* every time
[**silent**] breakpoint *n* is reached. [**silent** *command-list* 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** includes floating point

Execution

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

Display

print [*/f*] [*expr*]
p [*/f*] [*expr*]
x
d
u
o
t
a
c
f
call [*/f*] [*expr*]
x [*/Nuf*] [*expr*]
N
u
f
disassem [*ad*]

Automatic
display [*/f*]
display
undisplay *n*
disable disp
enable disp
info display

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; p type gives more detail
p type [<i>expr</i>]	
p type <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
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, 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 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 list
h save off/on	control use of external file for command history
print ...	groups with the following options:
p ...	
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 <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> ; ¹ 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>function</i>	
+ <i>off</i>	
- <i>off</i>	
* <i>address</i>	
list <i>f, l</i>	
info line <i>num</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