GDB QUICK REFERENCE GDB Version 4

Essential Commands

 $\verb"b" [file:] function"$ p expr run [arglist]gdb program [core] debug program [using coredump core] set breakpoint at function [in file] next line, stepping over function calls next line, stepping into function calls start your program [with arglist] continue running your program display the value of an expression backtrace: display program stack

Starting GDB

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

Stopping GDB

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

Getting Help

 $\mathtt{help}\ command$ $\mathtt{help}\ class$ describe command one-line descriptions for commands in list classes of commands

Executing your Program

run ... < inf > outf start your program with input, output use dev as stdin and stdout for next run kill running program start your program with current argument start your program with arglist redirected list

set args show env var show env show args set args arglist tty devset environment variable var show value of environment variable var show all environment variables specify arglist for next run display argument list specify empty argument list

> $\mathtt{frame}\ [n]$ bt [n]backtrace $\lfloor n \rfloor$

Shell Commands

set env var string

unset env var

remove var from environment

shell cmd execute arbitrary shell command string call "make" Print working directory change working directory to

info catch

info reg [rn]. info locals info args

surround optional arguments ... show one or more arguments

(c)1991, 1992, 1993 Free Software Foundation, Inc.

Permissions on back

$\verb|break|| file: |line|$ Breakpoints and Watchpoints set breakpoint at line number in file break main.c:37

b [file:]line

 $\verb|break|| file:] func$ break -offsetbreak ... if break +offsetbreak conditionally on nonzero expr set a watchpoint for expression exprtemporary break; disable when reached set breakpoint at next instruction set break at offset lines from current stop set breakpoint at func [in file] break on all functions matching regex new conditional expression on breakpoint set breakpoint at address addr n; make unconditional if no expr

cond n expr

tbreak ...

break break *addr

info watch info break show defined watchpoints show defined breakpoints $\mathtt{catch}\ x$ watch expr rbreak regea

clear [file:] line delete breakpoints [or breakpoint n] delete breakpoints on source line delete breakpoints at entry to fun()delete breakpoints at next instruction

 ${\tt clear}\ [file:]$

clear

enable breakpoints [or breakpoint n] disable breakpoints [or breakpoint n] disable again when reached

execute GDB command-list every time ignore breakpoint n, count times breakpoint n is reached. [silent delete when reached

Program Stack

info all-reg [rn info frame $\lfloor addr \rfloor$ print trace of all frames in stack; or of nselect frame n frames up register values [for regs rn] in selected local variables of selected frame arguments of selected frame select frame n frames down frame; all-reg includes floating point n; if no n, display current frame frames—innermost if n>0, outermost if

Execution Control

break at C++ handler for exception x

enable breakpoints [or breakpoint n]; enable breakpoints [or breakpoint n];

> Ħ 0

enable del [n]enable once $\left[n
ight]$ $\verb"enable" \left[n \right]$ $\mathtt{disable}\ [n]$ $\mathtt{delete}\ [n]$

 ${
m end}$ of ${\it command-list}$ suppresses default display

commands n

command-list silent ignore n count

describe selected frame, or frame at addr select frame number n or frame at address exception handlers active in selected frame

n down

continue running; if count specified, ignore execute until another line reached; repeat this breakpoint next count times

step by machine instructions rather than source lines

count times if specified

stepi [count s [count] step | count c [count] continue [count]

si [count]

next machine instruction rather than execute next line, including any function

source line

 $\mathtt{ni}\ [count]$

nexti | count n [count] next [count]

until [location]

resume execution at specified line number resume execution with signal s (none if 0) pop selected stack frame without run until selected stack frame returns run until next instruction (or location) executing [setting return value

 $\mathtt{return}\ [\mathit{expr}]$

finish

set var=expr jump * address $jump\ line$ signal num

evaluate expr without displaying it; use

for altering program variables

 $\ge \left[/ \mathit{Nuf} \right] \; expr$ call [/f] exprdisassem [addr]show value of expr [or last value \$] examine memory at address expr; optional octal count of how many units to display character address, absolute and relative binary unsigned decimal signed decimal printing format. Any print format, or unit size; one of like **print** but does not display **void** floating point hexadecimal format spec follows slash according to format f: s null-terminated string g giant words (eight bytes) w words (four bytes) b individual bytes machine instructions halfwords (two bytes)

Automatic Display

display memory as machine instructions

 ${\tt disable\ disp}\ n$ info display enable disp nundisplay ndisplay

display [/f] expr show value of expr each time program disable display for expression(s) number nnumbered list of display expressions enable display for expression(s) number nremove number(s) n from list of display all enabled expressions on list stops [according to format fautomatically displayed expressions

Expressions	
expr	an expression in C, C++, or Modula-2
	(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 $type$
€9	most recent displayed value
\$ n	nth displayed value
\$\$	displayed value previous to \$
*	nth displayed value back from \$
₽.	last address examined with \mathbf{x}
\$	value at address \$_
\$var	convenience variable; assign any value
show values $\lceil n ceil$	show last 10 values or surrounding \$n
[]	

Symbol Table

show conv

	info	TITT
	func	TIII O additess
	[regex]	Ö
_	sh	211

 ${ t info var} \ [{\it regex}]$

ptype type ptype [expr

display all convenience variables

show names, types of global variables (all, show where symbol s is stored or matching regex) (all, or matching regex) low names, types of defined functions

whatis $\lfloor expr \rfloor$

show data type of expr [or \$] without evaluating; ptype gives more detail

describe type, struct, union, or enum

GDB Scripts

source script

script

read, execute GDB commands from file

document cmd define cmd command-list create new GDB command cmd; execute script defined by command-list

create online documentation for new GDB end of help-text end of command-list command cmd

help-text

Signals

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

Debugging Targets

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

Controlling GDB

set param value

set one of GDB's internal parameters

Parameters understood by set and show: height lppediting on/off confirm on/off complaint limit language lang number of messages on unusual symbols control readline command-line editing number of lines before pause in display enable or disable cautionary queries display current setting of parameter Language for GDB expressions (auto, c or

listsize noctal, decimal, or hex number use str as GDB prompt number of lines shown by list modula-2)

+off

width cplwrite on/off verbose on/off radix base ${ t prompt} \ str$ control messages when loading symbols Allow or forbid patching binary, core files number of characters before line folded representation

h file filename h exp off/oncontrol use of external file for command file for recording GDB command history number of commands kept in history list groups with the following options: disable/enable readline history expansion

h size size

history ...

(when reopened with exec or core)

h save off/onprint ... groups with the following options:

p address on/off print memory addresses in stacks, values p demangl on/off source (demangled) or internal form for p array off/on compact or attractive format for arrays C++ symbols

p asm-dem on/off demangle C++ symbols in machineinstruction output

p union on/off p vtbl off/on p object on/offp elements limit number of array elements to display pretty off/onprint C++ derived types for objects display of union members struct display: compact or indented display of C++ virtual function tables

Ъ

 ${\tt show}$ commands nshow commands show commands + show next 10 commands show 10 commands around number nshow last 10 commands

Working Files

file [file]

use file for both symbols and executable;

symbol [file] exec [file]core [file] add-sym file addr load fileread additional symbols from file, dynamically link file and add its symbols use symbol table from file; or discard use file as executable only; or discard read file as coredump; or discard dynamically loaded at addr with no arg, discard both

display executable and symbol file path add dirs to front of path searched for display working files and targets in use list names of shared libraries currently executable and symbol files

show path path dirs info files

info share

Source Files

dir dir names show dir show current source path clear source path add directory names to front of source

list list lines list -[file:]numline number in named file display source surrounding lines, specified show next ten lines of source show previous ten lines

rev regex forw regex info sources info source info line num list f, l-off*address[file:] functionbeginning of function in named file show name of current source file show starting, ending addresses of search preceding source lines for regex search following source lines for regex list all source files in use from line f to line lline containing address off lines previous to last printed off lines after last printed compiled code for source line num

GDB under **GNU** Emacs

M-d C-h m C-x & M-c M-j M-s M-n C-c C-f M-nC-x SPC M-x gdb step one instruction (stepi) down arg frames (down) up arg frames (up) continue (cont) finish current stack frame (finish) next line (next) step one line (step) describe GDB mode (in source file) set break at point copy number from point, insert at end run GDB under Emacs

GDB License

show warranty show copying There is NO WARRANTY for GDB. Display GNU General Public License Display full no-warranty statement

Copyright (c)1991, 1992, 1993 Free Software Foundation, Inc. Roland H. Pesch

General Public License This card may be freely distributed under the terms of the GNU

The author assumes no responsibility for any errors on this card

GDB itself is free software; you are welcome to distribute copies of Please contribute to development of this card by annotating it.

absolutely no warranty for GDB.

it under the terms of the GNU General Public License.