CSCI 370: Computer Architecture

GDB Reference

Essential Commands

`gdb` *program*	debug *program*
`b` *[file:]function*	set breakpoint at *function* [in *file*]
`r` *[arglist]*	start your program [with *arglist*]
`bt`	backtrace: show stack frames
`p` *expr*	display the value of expression *expr*
`c`	continue running
`n`	next **line** stepping **over** function calls
`s`	next **line** stepping **into** function calls

Starting and Stopping GDB

`gdb` *program*	debug *program*
`gdb --help`	describe command line options
`quit`	exit GDB; also q or EOF (C–d)
`INTERRUPT`	(C-c) terminate current command or send to running process

Executing

`run` *[arglist]*	start your program [with *arglist*]
`run`	start your program with current arglist
`kill`	kill running program
`set args` *[arglist]*	specify argument list for next run
`set args`	specify empty argument list for next run
`show args`	display argument list

Breakpoints

`break` *[file:]line*	set breakpoint at *line* number [in *file*]
`break` *[file:]function*	set breakpoint at *function* [in *file*]
`break +`*offset*	set break at *offset* **lines** from current stop
`break -`*offset*	set break at *offset* **lines** from current stop
`break *`*addr*	set break at address *addr*
`clear`	delete breakpoints at next instruction
`clear` *[file:]line*	delete breakpoints at source *line*
`clear` *[file:]function*	delete breakpoints at entry to *function*
`delete` *[n]*	delete breakpoints [or breakpoint *n*]
`enable` *[n]*	enable breakpoints [or breakpoint *n*]
`disable` *[n]*	disable breakpoints [or breakpoint *n*]

Program Stack

`backtrace` *[n]*	print all frames in stack; when *n* is specified innermost *n* when *n>0,* outermost *n* when *n<0*
`frame` *[n]*	select frame number *n* or frame at address *n*. When *n* isn't specified, 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 in selected frame [for regs *rn*] in selected frame; `all-reg` includes FP registers
`info all-reg` *[rn]*	

Execution Control

`continue` *[count]*	debug *program*
`step` *[count]*	set breakpoint at *function* [in *file*]
`stepi` *[count]*	start your program [with *arglist*]
`next` *[count]*	backtrace: show stack frames
`nexti` *[count]*	display the value of expression *expr*
`until` *[location]*	continue running
`finish`	next **line** stepping **over** function calls
`signal` *s*	next **line** stepping **into** function calls

Display

`disassem` *[location]*	display memory as machine instructions
`print` *[/f] expr*	show value of expression *expr*
`call` *[/f] expr*	like print but does not display `void`
`x` *[/Nuf] expr*	examine memory at address *expr*

Format specifier *f*

x	hexadecimal
d	signed integer
u	unsigned integer
o	octal
t	binary
a	address (abs and rel)
c	character
f	floating-point
s	null-terminated string
i	machine instruction

Expressions

`$reg`	register value
`*(expr)`	dereference expr
`e+e`	addition
`e-e`	subtraction
`e*e`	multiplication
`a[b]`	equivalent to *(a+b)
num	numeric literal

Count specifier *N*

numeric value

Unit size specifier *u*

b	8-bit (byte)
h	16-bit (halfword)
w	32-bit (word)
g	64-bit (giant)

Examples

`print $rax`	prints the value of register %rax
`x/s 0x40018390`	prints memory location as a string
`print *$rbx`	prints (%rbx)
`x/10w $rdi`	prints 10 ints starting at the address specified with %rdi

GDB Dashboard Extension

`dashboard`	redraw the dashboard
`dashboard -layout [args]`	change layout
`help dashboard`	print help/usage