

CSCI 2330 GDB Reference Sheet

Start

`gdb myprog` Launch myprog in gdb

Run and Stop

`help` Get information about gdb
`quit` Exit gdb
`run` Run program
`run 1 2 3` Run with command-line arguments 1 2 3
`run < in.txt` Run with input redirected from in.txt
`kill` Stop the program
`Control-D` Exit gdb
`Control-C` Stop the currently running gdb command
`make` Run make to rebuild without leaving gdb

Breakpoints

`break sum` Set breakpoint at entry to function sum
`break 20` Set breakpoint at line 20 in current file
`break prog.c:20` Set breakpoint at line 20 in prog.c
`break *0x80483c3` Set breakpoint at address 0x80483c3
`delete 1` Delete breakpoint #1
`disable 1` Disable breakpoint #1
`enable 1` Enable breakpoint #1
`delete` Delete all breakpoints
`clear sum` Clear breakpoints at entry to function sum

Execute

`step` Execute one C line
`next` Execute one C line
(treats functions as one line)
`stepi` Execute one instruction
`stepi 4` Execute four instructions
`nexti` Execute one instruction
(treats function as one instruction)
`continue` Execute until next breakpoint
`until 3` Execute until breakpoint #3
`finish` Execute until current function returns
`call sum(1, 2)` Call sum(1, 2) and print return value

Context

`backtrace / where` Print current address & stack backtrace
`info program` Print current status of the program
`info functions` Print functions in program
`info stack` Print backtrace of the stack
`info frame` Print info about current stack frame
`info registers` Print registers and their contents
`info breakpoints` Print status of breakpoints

Examine Code

`disas` Disassemble current function
`disas sum` Disassemble function sum
`disas 0x80483b7` Disassemble function around 0x80483b7
`disas 0x80483b7 0x80483c7` Disassemble within address range
`print /x $rip` Print program counter in hex
`print /d $rip` Print program counter in decimal
`print /t $rip` Print program counter in binary

Examine Data

`print /d $rax` Print contents of %rax in decimal
`print /x $rax` Print contents of %rax in hex
`print /t $rax` Print contents of %rax in binary
`print 0x100` Print decimal representation of 0x100
`print /x 555` Print hex representation of 555
`print /x ($rsp+8)` Print (contents of %rsp) + 8 in hex
`print *(int*) 0xbffff890` Print integer at address 0xbffff890
`print *(int*) ($rsp+8)` Print integer at address %rsp + 8
`print (char*) 0xbffff890` Print string at address 0xbffff890

`x/w 0xbffff890` Examine 4-byte word at address 0xbffff890
`x/w $rsp` Examine 4-byte word at address \$rsp
`x/wd $rsp` Examine 4-byte word at address \$rsp
in decimal
`x/2w $rsp` Examine two 4-byte words at address \$rsp
`x/2wd $rsp` Examine two 4-byte words at address \$rsp
in decimal
`x/g $rsp` Examine 8-byte word at address \$rsp
`x/s 0xbffff890` Examine string stored at 0xbffff890
`x/20b sum` Examine first 20 opcode bytes of func sum
`x/10i sum` Examine first 10 instructions of func sum
`display /FMT EXPR` Print expression EXPR using format FMT
each time execution stops

Formats: x/[NUM][SIZE][FORMAT]

If not given, uses sensible default or last-used format

NUM = number of objects to display

SIZE = size of each object

b = 1 byte

h = 2 bytes ("half word")

w = 4 bytes ("word")

g = 8 bytes ("giant/quad word")

FORMAT = format for displaying each object

a = address (pointer)

d = decimal

x = hexadecimal

o = octal

t = binary