CSCI 2330 GDB Reference Sheet

Start and Layout

`gdb myprog`	Launch myprog in gdb (from shell)
`layout src`	Show source code while debugging
`layout asm`	Show assembly code while debugging

Run and Stop

`help [h]`	Get information about gdb
`quit [q]`	Exit gdb
`run [r]`	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 [k]`	Stop program
`Control-D`	Exit gdb
`Control-C`	Stop the currently running gdb command
`make`	Run make to rebuild without leaving gdb

Breakpoints

`break [b]`	Set breakpoint at current location
`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 [d]`	Delete all breakpoints
`delete 1`	Delete breakpoint #1 (from "info break")
`disable 1`	Disable breakpoint #1
`enable 1`	Enable breakpoint #1
`clear sum`	Clear breakpoints at entry to function sum

Execute

`step [s]`	Execute one C line
`next [n]`	Execute one C line (treats functions as one line)
`stepi [si]`	Execute one ASM instruction
`stepi 4`	Execute four ASM instructions
`nexti [ni]`	Execute one ASM instruction (treats function as one instruction)
`continue [c]`	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 [bt]`	Print current address & stack backtrace
`info [i]`	Print info about program state (see below)
`info program`	Print current status of the program
`info break`	Print status of breakpoints
`info frame`	Print info about current stack frame
`info register`	Print registers and their contents

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 [p]`	Print expression (last value by default)
`print foo`	Print value of foo
`print /x foo+5`	Print value of (foo+5) in hex
`print /d 0xAB`	Print 0xAB in decimal
`print /d $rax`	Print contents of register %rax in decimal
`print /x $rax`	Print contents of register %rax in hex
`x/FMT ADDRESS`	Examine memory at ADDRESS using format FMT
`x/g 0xbffff890`	Examine 8-byte word at address 0xbffff890
`x/g $rsp`	Examine 8-byte word at address $rsp
`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/s 0xbffff890`	Examine string stored at 0xbffff890
`x/6bc $rsp`	Examine six bytes at address $rsp as chars
`x/10i sum`	Examine first 10 instructions of func sum
`x/20b sum`	Examine first 20 opcode bytes of func sum
`display /FMT EXPR`	Print expression EXPR using format FMT each time execution stops
`display`	Show current auto-display expressions
`undisplay NUM`	Remove expression NUM from auto-display

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
  - d = decimal
  - x = hexadecimal
  - t = binary
  - a = address (pointer)
  - c = character
  - s = string