CSE 351 Section 3

GDB Advanced Features
HW 1 Questions

Some parts are a little confusing, what questions do you have?
Last Time

- Covered GDB basics
  - Compiling for GDB with the \(-g\) flag
  - Using `break` to set breakpoints
  - Printing variable values with `print`
  - Examining blocks of memory with `x`
- These are great ways to debug a C program
- We will learn more today for use in Lab 2
Bomb Lab

Basic introduction

To get the lab into your workspace

```
tar xvf /projects/instr/12au/cse351/bombs/$USER/lab2-bomb.tar
```

Demo of most of step 1!
GDB set args

helps to set the args to defuser.txt within gdb so that when you run, it passes the filename

To pass an argument of "defuser.txt", use

set args defuser.txt
**GDB disassemble**

When stepping through code, use `disas` to see the disassembly near your current line.

Remember to use `where` if you forget where you are in the code.
GDB info reg

Lists all registers and their current value

If you need less output, try:

print $<reg name>

Ex: print $eax
GDB stepi, nexti

stepi
Run the next assembly command, jumping if necessary

nexti
Run the next assembly command, skipping over function calls
GDB display

Makes a list of variables/expressions to output each time the debugger pauses.

For example, to track the value of $eax:

```plaintext
display $eax
or
display /x $eax
```
Other tool: objdump

-t: symbol table
-d: disassemble

demo:
objdump -t bomb
(like a map of function and other locations)

objdump -d bomb
objdump -d bomb > filename.txt
(all the assembly code, split into chunks)
Other tool: `strings`

display all strings stored in the bomb:

`strings -t x bomb`
C vs. Assembly

If you would like some practice looking at C and assembly, compile this and run it in gdb!

Demo program

http://goo.gl/75vdM
wget http://www.cs.washington.edu/education/courses/cse351/12au/section-slides/asm_example.c

Compile - don't forget the -g flag