CSE 351

Introduction & Course Tools

Introduction

- I graduated from UW with a Bachelor’s degree in computer engineering
- Currently am a 5th-year Masters student
- This is my 7th quarter as a TA for 351
- I run marathons, race triathlons, etc

Why take 351?

- Aside from it being a CSE requirement…
- The labs are fun
- You learn how computers work!
- Introduction to the C language, as well as Intel x86_64 assembly

Working environment

- You have three options
  - Install the CSE Home VM (Recommended)
  - If you have a CS account, you can use the lab machines (or remote into attu)
  - You can use your own personal computer running a Linux distribution (i.e. Ubuntu)

Course Tools

- Text editor
- GNU Compiler Collection (GCC)
- GNU Project Debugger (GDB)
- You can find all of these installed on the CSE Home VM

Text editor

- This is a personal preference
- Try several, choose the one you like
- Command-line
  - Nano
  - Vim
  - Emacs
- Graphical
  - Gedit
  - Emacs

GCC

- This is a command-line utility that compiles your C files
- To create an executable program in C, there are two phases:
  - Compiling
  - Linking
- Compile: `gcc -Wall -std=gnu99 -c main.c`
  - This produces an object file called `main.o`
- Link: `gcc main.o -o test`
  - This produces an executable program called `test`

GCC

- For this class, you will only be writing simple programs, so you can easily combine the compiling & linking phases
- Compile & Link: `gcc -Wall -std=gnu99 main.c -o test`
- This accomplishes the same thing as before in just one command

Hello World

```
#include <stdio.h>

int main(int argc, char *argv[]) {
   printf(“Hello World!\n”);
}
```

Try it on your own

- If you have a laptop with you, download the following file: HelloWorld.c
- Navigate to the directory where it is located, compile it, and run it

Try it on your own

- Navigating to the directory:
  - The command `cd` can help
- Looking inside each directory:
  - Run the `ls` command
- Compiling the program:
  - `gcc HelloWorld.c -o hello`
- Running the program:
  - `./hello`

About `printf()`

- Used for printing to the console
- You can’t just concatenate strings with variables like you can in Java
- Insert placeholders to print out variables
  - The placeholder depends on the type of the variable
  - “%d”, signed int
  - “%u”, unsigned int
  - “%f”, float
  - “%s”, string
  - “%x”, hexadecimal int
  - “%p”, pointer

`Printf()` Examples

- `printf(“I am %d years old”, 20)`
  - Prints “I am 20 years old”
- `printf(“My name is %s”, “Steve”)`
  - Prints “My name is Steve”
- `printf(“%d in hex is %x”, 2827, 2827)`
  - Prints “2827 in hex is 0xb0b”

Another example

- Download the file: calculator.c
- Again, navigate to the file, compile it, and run it
  - Example usage: `./calculator 4 5 +`

Linux man pages

- When you don’t know how to use a particular shell command, you have several options
- One option is this site: http://google.com
- Another option is using the `man` command:
  - `man 3 printf`
  - This will give a detailed description of `printf()`

Lab 0 introduction

- If you haven’t already downloaded it, go ahead and download Lab 0
- Open the arrays.c file in an editor and we will go through it as time permits