CSE 351

Introduction & Course Tools

Meet Your TA

TA Name

- Interesting information examples:
  - Where you are from
  - Year in school
  - Hobbies
  - Unique talents

Introductions

- Pick an interesting (but quick) ice breaker to get students to introduce themselves or a classmate.

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 x86_64 assembly

Working Environment

- Install the CSE Home VM (best option for EE)

  or

- Use the CSE Lab computers running Linux

  or

- Remote access into Attu (only CSE Students)

NO OTHER ENVIRONMENTS ARE SUPPORTED

Text Editors

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

Bash vs SSH

Bash: The ***command line interface***

SSH: The ***connection*** between you and the server

Using Bash

- Changing directories
  - cd <path>

    (Relative path or absolute path beginning with “/”)
  - cd ..

    (Go up one directory)

- Editing files
  - ‘vi <file path>’ or ‘emacs <file path>’

    (Relative path or absolute path beginning with “/”)

Accessing Attu

- Connect via SSH to use Bash
  - Windows: Use PuTTY
  - Mac: Open ‘terminal’, and run:

    ‘ssh –l<CSE USERNAME> attu.cs.washington.edu’

- Transferring Files
  - Use FileZilla/Cyberduck -- *FTP Clients* -- to drag & drop between server and your computer
  - Bash alternative: “scp” (google or ask us how to use)

Accessing the VM

- Install the VM in VMWare
  - Ask for help if you need it
- To access Bash
  - Spin up the VM, open console / terminal
- To transfer files
  - Drag and drop from your computer to the VM window and vice versa

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=c99 -c main.c`**
  - This produces an object file: **`main.o`**
- Link: **`gcc main.o -o test`**
  - This produces an executable program file: **`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=c99 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 HelloWorld.c from the course website
- 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”, “Alfian”)

- Prints “My name is Alfian”

printf(“%d in hex is %x”, 2827, 2827)

- Prints “2827 in hex is 0xb0b”

Another Example

- Download calculator.c from the course website
- 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 or C function, 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()`**