Lecture 1 - 5/31/2022

Introduction

These are the notes I take during class.

- Wherever I use the angle brackets `<example>`, they denote that you replace this chunk of text with something else **without** angle brackets `:)`

Before class starts, one must be able to log on to Grace following these instructions and then use this site to set up their environment. I did this below.

Note: if you are returning to this class, you must reset your Grace Environment.

Using grace

- "Eclipse died" - Nelson
- Command Line Interfaces (CLIs), are used everywhere in development, so it is good to know.
- Grace is a linux computer located at the address `grace.umd.edu`.
    - Remote computers such as grace are also called "hosts".

To connect to grace, we must ssh to it. This can be done using an ssh client. For Windows, this class uses MobaXterm. For MacOS, this class uses XQuarts. Instructions to download these sofwares can be found here.

In class, Nelson did not specify any ssh client for linux users. He also said linux users should know what they're doing. I believe to connect to grace from a linux client, you just do

```
ssh grace.umd.edu
```

from the command line.

Using Grace Remotely

If you are not connected to eduroam at UMD, you will need to access campus resources such as Grace using a VPN. UMD uses GlobalProtectVPN. The installation instructions for this can be found at terpware.

Linux Commands

- Create directory by using `mkdir <example>` where `<example>` is the name of the folder you want to make.
    - `mkdir` stands for "make directory"
- Go into a folder by using `cd <folder>` where `<folder>` is the name of the folder you want to go into
- Go back to the home folder by using `cd`
- List everything in a folder using `ls`
    - `ls -F` puts a `/` next to everything that is a folder
- copy: `cp -r <directory/of/file/or/folder/fileOrFolderName> <newDirectory>`
    - `-r` means recursive, so it will copy everything

- remove directory: `rmdir <dirName>` where `<dirName>` is the name of the directory you wish to remove
- remove file: `rm <fileName>` where `<fileName>` is the name of the file you wish to remove
- `mv`: renames a file or moves a file/directory to a directory.
    - for example, assuming `d1` is a directory and `m1` is a file:
        - `mv f1 f2` -> renames `f1` to `f2`
        - `mv f1 d1` -> moves `f1` to the directory `d1`
        - `mv d1 d2` -> renames `d1` to `d2`

Directories

- Diretories are like folders
- In the host `grace.umd.edu`, there are several directories, one of which is your account which was set up using the link in Nelson's email.
- In your account folder, there is a `216` folder in which you can put your projects for this class.

Multiple Windows

- you can connect to Grace multiple times with an ssh client, so each connection is a new window you can use to interact with the host.

Setting up the environment

This command below will create the correct alias for submitting things, and links to class resources.

```
/afs/glue/class/summer12022/cmsc/216/0101/public/bin/setup_216
```

Restart the environment by using `exit` and then reconnecting using the same ssh methods.

Now, if you execute `ls`, you should see that there is a `216` directory and a `216public` directory.

- `216` directory - This directory is located in your home directory. You need to complete your assignments and do any work related to the course in this directory.
- `216public` directory - You will see in your home directory a directory called `216public`. This is a directory we used to place material (e.g., project files, descriptions, etc.) you need for the course. Feel free to explore its contents. Note that you only have read permission for files available in 216public; if you want to edit any files, you need to copy our files to your 216 directory. For example, to copy class examples present in the directory `Week1`, you will execute the following command:

    ```
    cp -r ~/216public/lecture_examples/Week1 ~/216
    ```

Next, we have to setup our `gcc` alias. I'll copy paste this from the above class resource.

Setting `gcc` Alias

In this course you need to compile code using gcc and a set of options. Instead of having to type those options every time, linux allows you to create an alias. The steps to create an alias are:

1. Go to your home directory in grace. Remember that executing cd will take you to your home directory.
2. In your home directory open (or create if it does not exist) a file called `.aliases` (the filename has a period at the beginning.)
3. Add the following line to the `.aliases` file and make sure you add a blank line after it.
    - ```
      alias gcc "gcc -ansi -Wall -g -O0 -Wwrite-strings -Wshadow -pedantic-errors -fstack-protector-all -Wextra"
      ```
4. Log out and then log on again.
5. You can verify the alias has been set if you execute alias gcc and you see the new options.
6. If you have problems you can also cut and paste the command from the file gcc_aliases_info.txt that you will find in the info directory of the grace public directory.

Creating Files Using Editors

When you open a file with an editor, you type that editor's name followed by the file you wish to open. For example, to edit a file named `p1.c` with `nano`, you can enter the command: `nano p1.c`. If `p1.c` does not exist, it will be created automatically.

Nano

- `nano` is a file editor in Grace.
- open `nano` using `nano <filename>`, where `<filename>` is the name of the file you wish to edit. If the file doesn't exist, nano will create the file in the current directory.
- all commands in `nano` is shown at the bottom of the window.
    - the symbol `^` represents `ctrl`. For example, the command "`^X`" means "`ctrl+X`" to exit nano.

Coding in nano

We can code in nano by just typing the code there. The example code is this:

```
// stdio is "standard input/output"
#include <stdio.h>

int main () {
    // i swear to got I do not like Pepsi because Coca-Cola > Pepsi
    // but this is just the example Nelson gives
     printf("I like Pepsi\n");

     return 0;
}
```

Compiling code
```

- the tool `gcc` compiles the code. Use it by using `gcc <codeName>` where `<codeName>` is the name of the file which contains your code.
- `gcc` will output to a file called `a.out`. This file is the exacutable that `gcc` compiled. You can run this executable by typing `a.out`.

Compile and execute `p1.c` by running

```
gcc p1.c
```

Introduction Summary

Here is a summary of how the class began.

1. Learn how to use either MobaXTerm, XQuarts, or another ssh client to ssh into `grace.umd.edu`. The instrucitons for this are here
2. Follow this introduciton to linux here.
3. Take a look at these basic linux comands by Nelson

C

This example `C` program called `p2.c` shows some basic commands of a `C` program.

```c
# include <stdio.h>

int main (void) {
    int x, y;
    float m = 3.75;
    char letter = 'a';

    x = 10;
    printf("The value to print is %d and %f and %c\n", x, m, letter);

    i = 1;
    while ( i <= 5) {
        printf("Johnny and Amber\n")

        i++;
    }

    if (i == 6) {
        done = 1;
    } else {
        done = 0;
    }

    if (done) {
        printf("yes, we are done\n");
    } else {
        printf("NOOOOOOOOOOOO we are done\n");
```

```
    }

    return 0;
}
```

- `if` statements are like Java
- `while` loops are like in Java
- `do-while` loops are like in Java
- there is no `boolean` type. In `C`, `0` is false, anything else is true.
- different compilers initialize uninitialilzed variables to different random variables.