CSE 3100 Systems Programming
Lab #0

Welcome to Lab 0! At the start of each lab, you will receive a document like this detailing the tasks you are to complete. Read it carefully. Try your best to finish labs during the lab session.

This lab is intended to be interactive. Feel free to discuss the assignment with other students. However, please recall that all of your code should be your own! If you have trouble, feel free to raise your hand and ask your TA for guidance!

After lab 0, you should be able to access your VM, use basic commands in a shell (e.g. to copy/remove/rename files), edit text files like C source files, compile and run simple C code, and submit your work.

1 Setting Up Your VM

Before you can access your virtual machine, there is a short setup to complete.

1. **Ensure you are connected to university Wi-Fi and that you have a mobile device for 2FA**. If you are not connected to university Wi-Fi, you will need to use the UConn AnyConnect VPN available here. (This applies to when you are using the virtual machine for the duration of the semester to work on assignments. If you are not connected to university Wi-Fi, you need to be connected to the VPN.)
2. Open a terminal on your local machine and SSH (Secure Shell Protocol; a way of connecting remotely to the terminal of another machine in a network.) into your VM at the address **[NETID]-vm.cse.uconn.edu/**. If you are on Linux, you can run

   ```
   ssh [NETID]@[NETID]-vm.cse.uconn.edu
   ```

   to connect. If you are having issues with SSH or are unsure how to do so on Mac/Windows, ask your TA.
3. Enter your NetId password when prompted. You will not be able to see what you are typing as it is hidden to not reveal your password. Press enter when finished.
4. When prompted for 2FA, enter either 1 or 2 and press enter to select your preferred 2FA method.
5. You are (hopefully) now connected to your VM by SSH! Now, every command you type is run by your VM and relayed back to you in the terminal you have open! To quit the connection, you can run the command `exit`.
6. Run the command `/opt/scripts/codefix.sh`, which sets up your VSCode environment for browser usage.

You should now have access to your VM in browser at the address `[NETID]-vm.cse.uconn.edu`! You can just enter this into your browser of choice and begin working. Two *very* important things to remember for the VM:

- **BACK UP YOUR WORK! VMs are NOT backed up. Losing significant work to data loss on the VM is not an excuse. If you are done working on a file in your VM, make sure you download it to your local machine just in case.**
- **VMs undergo maintenance from 5:30AM to 7:30AM. There is a possibility that machines are down during this time, so do not plan on having access during this block of time.**

2 Shell

In the shell (most likely bash), try the following commands (and explore a bit!):

`pwd`	`Show the current directory (where you are).`
`ls`	`List the files and directories in a directory.`
`ls -l`	`The -l flag tells ls to show more details.`
`mkdir adir`	`Create a directory. Try different names.`
`ls -l adir`	`Lists the files in the directory ’adir’.`
`cd adir`	`Move into the directory ’adir’.`
`ls -l ..`	`List the files in the parent directory.`
`cd`	`Go back to your home directory.`
`cd adir`	`Go to ’adir’ again.`
`echo "hi!"`	`Print ’hi!’ to the standard output (stdout).`
`echo "hi!" > file`	`Save ’hi!’ to a new file called ’file’`
`cat file`	`Print the contents of a file to stdout.`
`grep "hi" file`	`Search the file for the string ’hi’.`
`less file`	`View the file interactively (q to quit).`
`rm file`	`Permanently deletes the file.`
`cat`	`Read from standard input (stdin) and dump to stdout. Ctrl-D to exit.`
`history`	`Show history of commands.`
`man ls`	`Show the manual for the ’ls’ command.`
`man cat`	`Show the manual for the ’cat’ command.`
`man scanf`	`Show the manual for the ’scanf’ function.`
`exit`	`Exit from the shell.`

3 Text editors and your first C program!

To work on a project, you need to go to the corresponding directory in the virtual terminal. Normally Coding Rooms goes to the correct directory when you start a virtual terminal. Under the project directory, you can modify existing files and/or create new files (.c files, .h files, readme’s, make files, etc.).

You can use the web editor, or an editor in the virtual terminal, to edit text files, which include your C source code. If you decide to learn an editor in the terminal, there are three primary options: `vim`, `emacs`, and `nano`. It is highly recommended to learn one of them, in addition to the web editor.

Use your favorite editor to open `lab0.c`. If you use the editor in the Coding Rooms IDE, you can click on the Files icon (top left) and you can use use the buttons. Otherwise, here are different ways to do it in a terminal. The following three editors are available in many systems.

```
vi lab0.c
emacs lab0.c
nano lab0.c
```

Add the following C code in `lab0.c`. If you are using the editor in the Coding Rooms IDE, changes should save automatically. Otherwise, remember to save the file after editing.

```
#include <stdio.h>

int main(void)
{
     printf("Hello, World!\n");
     return 0;
}
```

This is your first C program! Before you can run it, you must compile it. You can do so by typing the following command in shell/console, or by clicking the 'Run' button in the Coding Rooms IDE. Note that you can use console/shell at the bottom of the Coding Rooms IDE. If you would like, you can also use the tips in **??** to open a visual desktop, where you can open multiple terminal windows.

```
cc lab0.c
```

This will compile your C code in `lab0.c` and create an executable called `a.out`. If the process fails, read the error messages, fix the error in your C file (using the text editor), and try again. If there is no error, you will see `a.out` in your current directory.

You can optionally specify the name of the executable that is generated by the compiler.

```
cc -o lab0 lab0.c
```

To run your program, type

```
./lab0
```

4 Something new

Revise `lab0.c`. Use a `while` loop to calculate the sum of positive even integers below 200 and print the result to the standard output. This is very similar to the sum example we have studied. Think about how much you should increase the loop control variable to get to the next even number.

Your program should output the following text:

```
Hello, World!
9900
```

5 Submission

If you think your code works, you should submit it on **gradescope**. You can download the files you have worked on in the file directory interface to the left of the IDE. There is a link to gradescope in the HuskyCT, you should already be added to the course with your UConn email.

Please remember to submit your work before the deadline for each assignment and exam.

Enjoy!

Exit from an editor in terminal

There are many tutorials on nano, vi/vim, and emacs. Feel free to explore and share with other students. In case you are stuck in an editor, here are the commands you can exit from the program.

nano

nano is very user friendly. You can see the help at the bottom of the screen.

```
nano a.txt          Start nano to edit file a.txt
Ctrl-x              Exit from nano. See the prompt at the bottom of the screen
```

There are many tutorials on the Internet, for example, here.

emacs

Commands in emacs are control characters or prefixed a set of characters.

```
emacs a.txt         Start emacs to edit file a.txt
Ctrl-x Ctrl-s       Save file
Ctrl-x Ctrl-c       Exit from emacs
```

Here and here are examples of cheat sheets and tutorials on emacs.

vi/vim

vi has three modes. Normally, you are in the normal mode when vi just starts. Press `i` to enter the insert mode, where you can insert/type text. Press `Esc` to exit from the insert mode and get back to the normal mode. If you get lost, press `Esc` many times to get back to the normal mode.

```
vi a.txt            Start vi/vim to edit file a.txt
:wq                 Save the file and then exit from vi. Work in the normal mode
:q!                 Exit from vi without saving file. Work in the normal mode
```

Here is an example cheat sheet.