Lab 0: LINUX ACCESS AND TOOLS

Objectives:

- Set up access to the ECEN 324 Linux VM.
- Create a sub-directory in your home directory and navigate between directories.
- Copy and rename a file.
- Edit a C program text file using a command line text editor.
- Compile and run a C program using command line tools.
- Submit the file to your instructor using the `submit` command.

The Linux system we will be using is a virtual machine (VM) that runs on a physical system in a BYU-Idaho data center. Using a remote access program on your own computer, you may log in to the remote machine and run commands on it.

There are various remote access networking protocols. Linux systems run `ssh` (secure shell) *servers* and users can use to connect into the system remotely. Linux, MacOS and Windows 10 systems typically have SSH *clients* installed as one of the applications on the system that can connect to the remote server. You can invoke an SSH client on your computer (your local system) to remotely access the remote Linux system. There are also many other applications that provide a GUI and handle the SSH connection for the user once it is set up in the application.

This lab will have you use a *terminal* (aka *console*, aka *command line*, aka *shell*) to start the SSH client and remotely access the Linux VM using command line commands.

Instructions

Please follow these steps for this assignment:

1) Find your username and password. Your username and a temporary password for accessing the Linux system will be sent to you in your **official BYU-Idaho** email (not I-Learn/Canvas email). If your instructor indicates that accounts have been activated/created and you have not received the email, check your junk/spam folders. If you still can't find the email, you might have rules in place on your email client that deleted it, or the 'dog in the cloud' ate it. Contact your instructor.

2) Open a terminal using one of the three methods below, depending on the operating system (OS) that you are using:
   a) **Windows:** Click the Start icon and type in `cmd`. This should find the 'Command Prompt' app.

   b) **MacOS:** Finder -> Applications -> Utilities -> Terminal.app
      or: Launchpad and search for `terminal`

   c) **Linux:** Open the terminal application. (The exact steps vary with the Linux distribution, but if you are running Linux, you probably already know how to do this.)

Note: If you are trying to connect to the Linux VM while on campus using Wi-Fi, you must be connected to the `BYUI` Wi-Fi, not the `BYUI_Visitor` Wi-Fi.

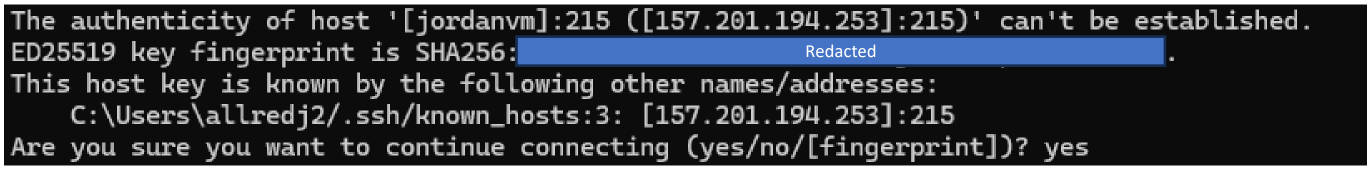
3) Start an SSH connection from the terminal by typing in the command below, replacing `yourUserName` with the username sent to you in the email, and then pressing "Enter."

```
ssh -p 215 yourUserName@157.201.194.253
```

Understanding the above syntax: The `ssh` command will start an SSH client on your local system. The IP address of the remote system you are connecting to is `157.201.194.253`. You are indicating that you want to log in to the remote system using the username you provide before the `@` symbol. The `-p 215` is telling the ssh client to use port 215. The standard ssh port of 22 is not allowed on this VM to help reduce illegal login attempts.

Tip (computer name): If you are on the BYU-Idaho network, you may optionally replace the IP address with the name of the computer, which in this case is `jordanvm` (i.e. user@computer).

If you get a message about the authenticity of the host and a prompt asking if you want to continue connecting, you should say yes:

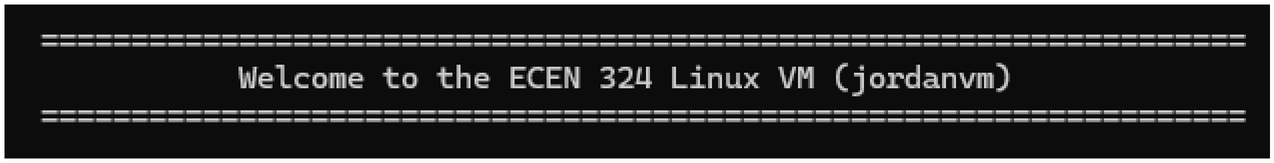


```
The authenticity of host '[jordanvm]:215 ([157.201.194.253]:215)' can't be established.
ED25519 key fingerprint is SHA256:Redacted.
This host key is known by the following other names/addresses:
    C:\Users\allredj2/.ssh/known_hosts:3: [157.201.194.253]:215
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
```

4) You are now accessing the remote Linux system, and it will ask you for your password. Enter the password given to you in the email you received.

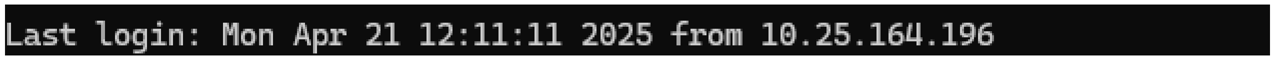
Note: When you type a password on many Linux systems, nothing shows up as you type, not even asterisks (*) or dots. That is normal.

If you have logged in successfully, you should see the following welcome message:



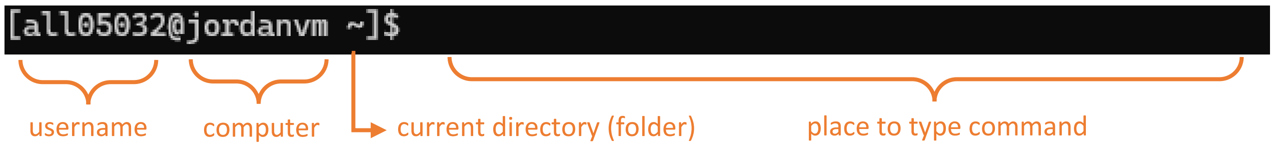
```
 ===================================================================
          Welcome to the ECEN 324 Linux VM (jordanvm)
 ===================================================================
```

The welcome message may be followed by a message about your last login attempt (if any):



```
Last login: Mon Apr 21 12:11:11 2025 from 10.25.164.196
```

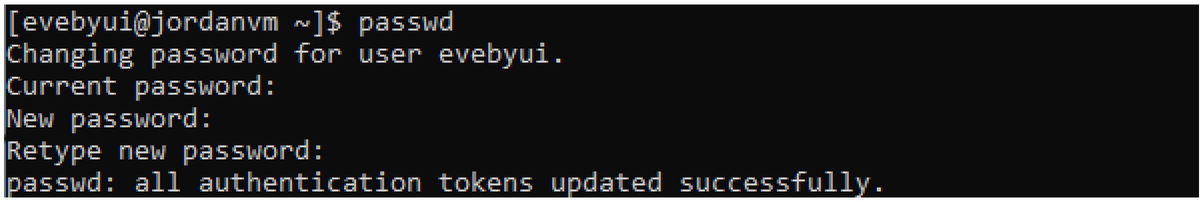
Lastly, you should see a command line prompt, which will look something like the following:



```
[all05032@jordanvm ~]$
```

Note: In Linux, a folder is called a *directory*. The ~ symbol means your *home directory* (aka user directory).

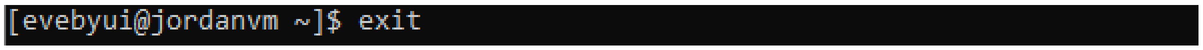
5) As good practice, you should change the password given to you. You may do so using the `passwd` command. It will ask you for the current password (the one you just used to log in with) and then a new password twice.



```
[evebyui@jordanvm ~]$ passwd
Changing password for user evebyui.
Current password:
New password:
Retype new password:
passwd: all authentication tokens updated successfully.
```

Note: If you ever forget your password, you will need to ask the instructor to reset it for you.

When you change your password, it would be a good idea to exit the SSH connection and reconnect using your new password to make sure nothing went wrong in changing your password. You can exit an SSH connection with the `exit` command.

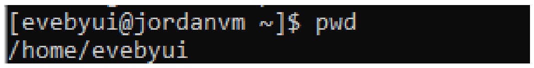


```
[evebyui@jordanvm ~]$ exit
```

If you exit, go ahead and SSH back into the remote Linux machine to continue the lab.

Tip (history): In most terminal windows, you may use the up-arrow key to recall previous commands you executed. Back in your local terminal, pressing the up-arrow key should bring up the ssh command you executed previously so that you don't have to re-type it.

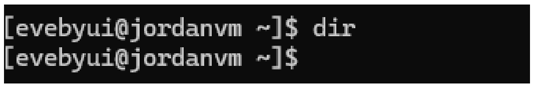
6) You are currently in your home directory. Execute a `pwd` command (**p**rint **w**orking **d**irectory) to see the name of the directory you are currently in.



```
[evebyui@jordanvm ~]$ pwd
/home/evebyui
```

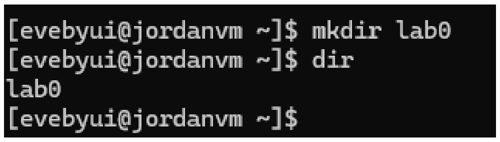
It should be `/home/yourusername`. When you change directories as you navigate around the file system, the `pwd` command can be helpful to remind you where you are.

7) Enter the `dir` command to display the contents of the current **dir**ectory. It should display nothing, since you haven't put anything into your home directory yet.



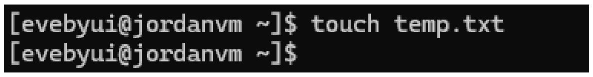
```
[evebyui@jordanvm ~]$ dir
[evebyui@jordanvm ~]$
```

8) You will now make a sub-directory (a folder inside a folder) in your home directory. You can make a new directory with the `mkdir` command (**mak**e **dir**ectory), followed by the name of the directory. Make a directory named "lab0" by typing `mkdir lab0` and pressing Enter. At this point, you are still in your home directory, but within your home directory will now be a directory named lab0. Display the contents of your home directory by doing another `dir` command:



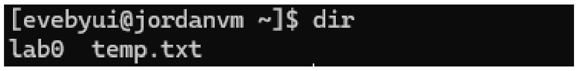
```
[evebyui@jordanvm ~]$ mkdir lab0
[evebyui@jordanvm ~]$ dir
lab0
[evebyui@jordanvm ~]$
```

9) In addition to a subdirectory, let's add a file to your home directory. There are many ways to create a new, empty file. An easy way is by using the `touch` command. Make a temporary file called `temp.txt` by using `touch` followed by the name of the file you want to create:



```
[evebyui@jordanvm ~]$ touch temp.txt
[evebyui@jordanvm ~]$
```

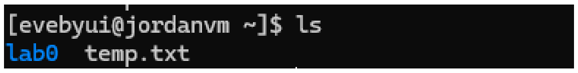
Then, verify that the file has been created by displaying the contents of your home directory again using the `dir` command:



```
[evebyui@jordanvm ~]$ dir
lab0  temp.txt
```

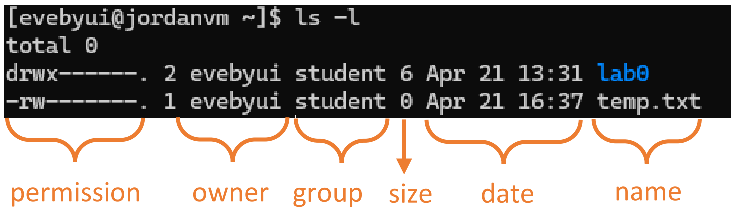
You should now see temp.txt listed along with the lab0 subdirectory you created earlier.

Tip (`ls`): The `dir` command lists anything in the current directory, whether it is a file or a directory, without any indication of whether it is a file or a directory. You can instead use the `ls` command (which stands for **l**i**s**t), which will color-code the contents. Blue items are directories.



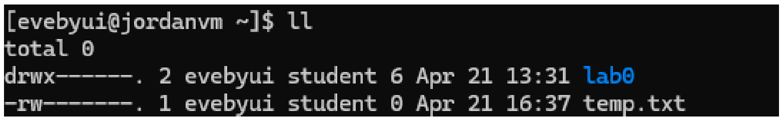
```
[evebyui@jordanvm ~]$ ls
lab0  temp.txt
```

Tip (`ls -l`): The `ls` command by default lists only the file names. You can also show a **l**ong list, with more details with the `ls -l` command. Adding the `-l` flag will show more information about each item in the current directory:



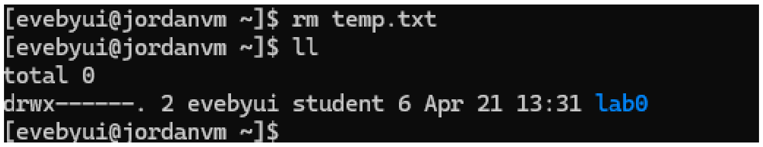
```
[evebyui@jordanvm ~]$ ls -l
total 0
drwx------. 2 evebyui student 6 Apr 21 13:31 lab0
-rw-------. 1 evebyui student 0 Apr 21 16:37 temp.txt
```

Tip (`ll`): The `ls -l` command is common enough that it is often aliased (shortcut) as just `ll`.



```
[evebyui@jordanvm ~]$ ll
total 0
drwx------. 2 evebyui student 6 Apr 21 13:31 lab0
-rw-------. 1 evebyui student 0 Apr 21 16:37 temp.txt
```

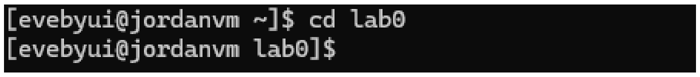
10) Delete the temporary file you just created. You can do that with the `rm` (**rem**ove) command. Type `rm` followed by the name of the file you wish to delete. Then, `ll` to verify the file is gone:



```
[evebyui@jordanvm ~]$ rm temp.txt
[evebyui@jordanvm ~]$ ll
total 0
drwx------. 2 evebyui student 6 Apr 21 13:31 lab0
[evebyui@jordanvm ~]$
```

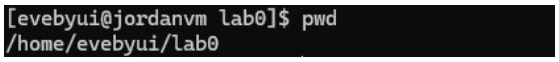
Note: When you reference a file or directory in a command, you can reference it either with the *absolute path* (the full pathname starting at root directory "/") or with the *relative path* (where to find it starting at the current directory). For example, the full path for the file you just deleted was `/home/username/temp.txt`. You could have done `rm /home/username/temp.txt`, but instead you only needed to type `rm temp.txt` because you were already in the `/home/username/` directory.

11) Now, go into the "lab0" directory by using the `cd` (**c**hange **d**irectory) command, followed by the name of the directory you want to change to:



```
[evebyui@jordanvm ~]$ cd lab0
[evebyui@jordanvm lab0]$
```

Do another `pwd` to verify that you are now in the `/home/username/lab0/` directory:



```
[evebyui@jordanvm lab0]$ pwd
/home/evebyui/lab0
```

Tip (parent directory alias): Two periods (..) is an alias/shorthand for the *parent* directory. So you can always cd back into the folder above the current directory with a "`cd ..`" command.

12) Copy a C file named `ecen324_lab0.c` from the `/home/`*`ecen324`*`/lab0/` directory. You will do this with the `cp` (copy) command. After `cp` you will put the file you wish to copy from followed by where you want to put it. Since the file is not in the current directory, you will need to provide the path name of the file along with the filename. The "." after the filename in the example below is an alias/shorthand for your current working directory (the directory name shown by a `pwd` command). Execute the command below to copy the file into the current directory.

```
cp /home/ecen324/lab0/ecen324_lab0.c .
```

Tip (tab key): You may use the tab key for file name completion. When you are typing a file name, you can hit tab for it to autocomplete the name of the file.

13) Edit the file using `nano`, `vi`, `vim`, or `emacs` text editors to edit the file you just copied. (The `nano` editor is recommended for Linux beginners.) Type the editor name followed by the file name:

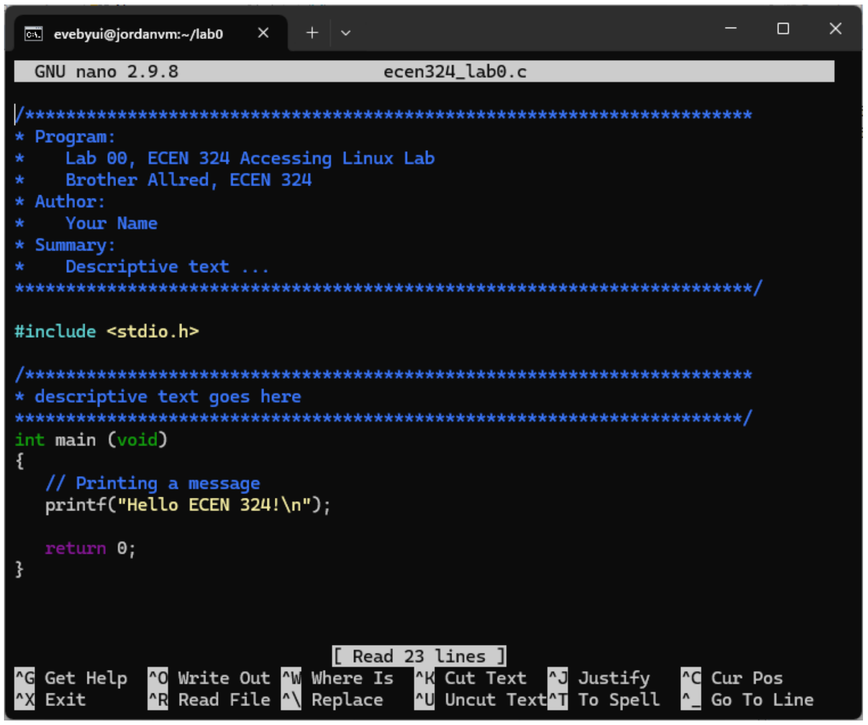
```
nano ecen324_lab0.c
```

Command-line editors are meant to use only the keyboard, not the mouse. You use the arrow keys to navigate in the file.

The screenshot to the right shows the nano editor with contents of the `ecen324_lab0.c` file.

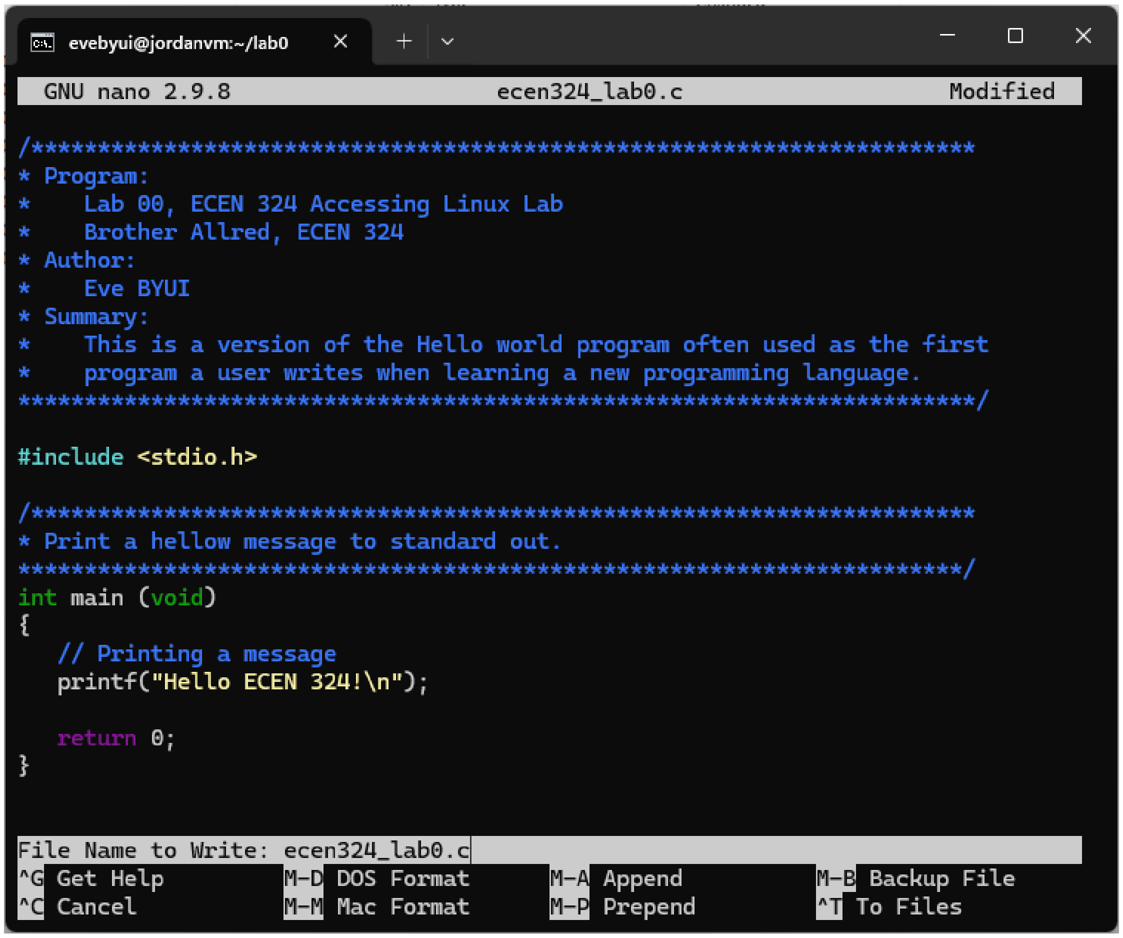
At the bottom are command options. The "^" symbol in the command options represents the control (ctrl) keyboard key.

Warning: Adding an extra line at the top or a space before the comment characters (*) will mess up the `submit` tool for C code assignments. In short, don't change the first four lines!



```
  GNU nano 2.9.8                          ecen324_lab0.c

/***************************************************************************
* Program:
*    Lab 00, ECEN 324 Accessing Linux Lab
*    Brother Allred, ECEN 324
* Author:
*    Your Name
* Summary:
*    Descriptive text ...
****************************************************************************/

#include <stdio.h>

/***************************************************************************
* descriptive text goes here
***************************************************************************/
int main (void)
{
   // Printing a message
   printf("Hello ECEN 324!\n");

   return 0;
}

                                [ Read 23 lines ]
^G Get Help   ^O Write Out  ^W Where Is   ^K Cut Text   ^J Justify    ^C Cur Pos
^X Exit       ^R Read File  ^\ Replace    ^U Uncut Text ^T To Spell   ^_ Go To Line
```

Edit the file with your name and edit the author and "descriptive text …" comments. To save your file after editing it, do a `ctrl-o` (write out) command, which will give you:



```
  GNU nano 2.9.8                 ecen324_lab0.c                       Modified

/*************************************************************************
* Program:
*    Lab 00, ECEN 324 Accessing Linux Lab
*    Brother Allred, ECEN 324
* Author:
*    Eve BYUI
* Summary:
*    This is a version of the Hello world program often used as the first
*    program a user writes when learning a new programming language.
*************************************************************************/

#include <stdio.h>

/*************************************************************************
* Print a hellow message to standard out.
*************************************************************************/
int main (void)
{
   // Printing a message
   printf("Hello ECEN 324!\n");

   return 0;
}

File Name to Write: ecen324_lab0.c
^G Get Help          M-D DOS Format       M-A Append           M-B Backup File
^C Cancel            M-M Mac Format       M-P Prepend          ^T To Files
```

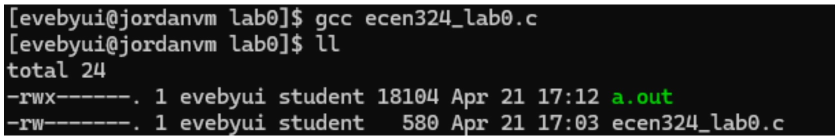
It is asking for the name of the file you want to save it as. The default is the current name. Just hit enter to write to the filename as displayed. After saving, you can exit `nano` with `ctrl-x`.

Note: if you change the filename by saving it with a different name, you will need to use your new filename in the gcc and submit commands shown below.

14) Compile your program with `gcc` (the **G**NU **C** **c**ompiler). Type gcc followed by the name of the C code file. If there are compile errors, fix them.

```
gcc ecen324_lab0.c
```

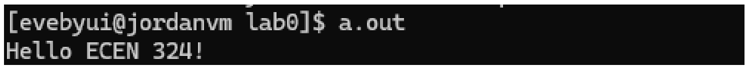
That will compile your code and create an executable machine code binary program file. If you don't specify a name for your executable program, it will be called `a.out`. Verify the `a.out` executable program exists with an `ll` command. Executables are shown in green.



```
[evebyui@jordanvm lab0]$ gcc ecen324_lab0.c
[evebyui@jordanvm lab0]$ ll
total 24
-rwx------. 1 evebyui student 18104 Apr 21 17:12 a.out
-rw-------. 1 evebyui student   580 Apr 21 17:03 ecen324_lab0.c
```

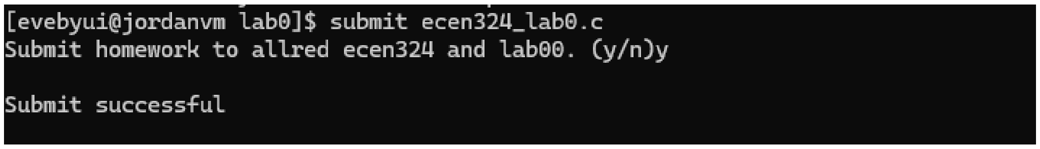
Tip (naming executables): You can alternatively specify a different name for the executable file you want gcc to output by using the `-o` flag followed by the desired name when you compile.

15) Run the program to make sure it works by typing the name of the program:



```
[evebyui@jordanvm lab0]$ a.out
Hello ECEN 324!
```

16) If it successfully prints "Hello ECEN 324!", has no compilation errors, and you have updated all relevant comments, then submit the C code using the `submit` command followed by the name of the C code file you want to submit. Verify the instructor name, the course name, and the assignment name, and then enter 'y' to submit:



```
[evebyui@jordanvm lab0]$ submit ecen324_lab0.c
Submit homework to allred ecen324 and lab00. (y/n)y

Submit successful
```

Warning: Make sure you submit the C code text file and not the executable binary program file.

Warning: When using the submit command you must be 'cd-ed' into the directory where the file you are submitting exists.

Additional Information

After completing this lab assignment, you may desire to learn about applications that are more user friendly when working on remote systems.

MobaXterm: For Windows, I highly recommend MobaXterm, which I can demo in class.

PuTTY: I believe that everyone should become familiar with the PuTTY tool. It is worth taking time to figure out. It also allows for X11 forwarding (see message below).

VS code extension: VS code has extensions that allow you to use the editor and other capabilities of VS code while working with files that reside on a remote system. To use VS Code with our Linux system please install this extension:

https://marketplace.visualstudio.com/items?itemName=Natizyskunk.sftp

(Find it my searching for SFTP in extensions. There are set up instructions in the I-Learn modules.)

Warning: Please don't use the "remote server extension" in VS code. It creates quota issues.

X11 forwarding: You will often see a "-X" option used on ssh commands. This enables the X11 protocol (gui interface) to be tunneled through an ssh connection. To use this, you need to be running an X server on your local system or using an application that has an X server built in to it, such as MobaXterm. On a macOS system, you may install XQuartz. On Windows, you may install Xming. Linux systems come with an X server already installed. The following document may be helpful: Linux Cloud Remote Access

The following video may be interesting to you: Emacs, PuTTY, Bash Tips and Tricks (19 minutes).