CMSC111 Computer Science II
Lab 2 – Getting to know Linux

Introduction
The purpose of this lab is to introduce you to the command line interface in Linux.

Getting started

In our labs
If you are in one of our labs, you just have to use your CS credentials to log into the machine in front of you. You can reset your CS account password reset on
https://web.engr.ship.edu/for-students/password-reset-request/

In Windows/Mac
Getting to our servers from your Windows or Mac machine just take a few steps.

How to set up 2FA:
https://web.engr.ship.edu/for-students/essential-software/mfa-for-ssh-servers/

You need a program to connect to our server (sloop.cs.ship.edu or clipper.cs.ship.edu). Linux or Mac users can use Terminal and connect to the server as follows:
ssh userid@sloop.cs.ship.edu
It will ask you for your password and then you will be in.

Windows users can use Windows PowerShell and connect to the server as follows:
ssh userid@sloop.cs.ship.edu
It will ask you for your password and then you will be in.

Directories & Files
In Linux, the things you called “folders” in Windows are called “directories” and you are always “in” a directory (which is called your “working directory”). That means that, unless you specify otherwise, linux will expect the files that you reference are in that directory.

Working directory
When you first log into the system, you start in your home directory – in other words, your home directory is your working directory. You can see what directory you are in by entering the **pwd** (print working directory) command. When you first log in, pwd should return something that looks like:

/home/userid

That is the path that is your home directory, but there isn’t any reason that you need to memorize that. Fortunately, you can always use ~ to reference your home directory.

You can create directories with the **mkdir** command. Let's suppose that you want to create a directory to hold all of the stuff for this class. You can create that directory (within your current working directory) by entering

mkdir s25

Once you have created a directory, you can make it be your working directory (which is often referred to a "going into it" using the **cd** (change directory) command. Move into the directory you just created by entering

cd s25

Use the **pwd** command to see that your working directory has been changed.

You can always get back to your home directory by typing

cd

to move back to your home directory and verify where you are with the **pwd** command.

You can nest directories like folders can be inside folders in Windows. Move into your s25 directory and use the mkdir command to create subfolders named cmsc111, labs, and lab2. Then move into the lab2 folder.

At this point, pwd should return something like

/userid/s25/cmsc111/labs/lab2

This is giving you the "path" to the directory. The first / is the topmost directory in the system and each subsequent section of this path is a subdirectory.

There are two more special folders that you need to know about: . means the current working directory and .. means the directory containing the current working directory (sometimes called the parent directory). That means, that entering

cd ..

will move you to the parent directory and doing that again will take you back to your home directory.

In review, we have learned about these special directory aliases:

Alias	Meaning
~	your home directory
.	the current working directory
..	the directory containing the current working directory

ls and man commands

You can see all of the files in your current working directory with the **ls** command. **ls** entered by itself will give you a list of the visible files and directories in the working directory, but there are also options that you can put on that command. For example, it is common to enter **ls –al** to list the files in the working directory. The "-al" are options on the command: the "a" means "all" (show all of the files) and the "l" means "long listing format" which gives a lot more information about each file (file permissions, size, date modified).

Move to your home directory and enter

ls –al

to see what files are in that directory. You may see other things, you should definitely see the cmsc111 directory you created earlier.

At this point, you can see that the general structure of linux commands is

<command> <options following a dash> <other arguments>

When you want to what options and arguments are valid for a command, you can enter **man <command>** to see those details.

Enter **man ls** to see the details about the ls command. Note that hitting the q key will get you out of the man entry.

Look at the output from that ls –al and the man entry for ls more carefully. See if you can figure out

- what the d in the first column means
- what the next 9 characters mean
- what are the units on the size of the file
- is the time stamp modification or creation

Also, compare the files/directories listed in the ls and ls –al commands to see what the "all" in the "a" option really means.

Editors

You need to pick an editor that you like. The obvious choices are: vi, pico, and nano. pico and nano are simple equivalents of notepad – completely language unaware. While vi(m) will require some extra time to learn, it is a very powerful editor. Here are some tutorials to help you play with vi:

vim: http://www.openvim.com/tutorial.html (except that we can use the arrow keys for movement)

To edit, lab2.c, you type

pico lab2.c or nano lab2.c or vi lab2.c

File commands

You can use commands to manipulate your files in linux. Use pwd and cd to put yourself in your Lab2 directory. Here are the important file commands (things in <> must be replaced with the name of the files you are manipulating). Do each of the examples in this table and use the ls command to see its effects.

Command	Meaning	Example
touch <f1>	create a file with a given name	touch foo
cp <f1> <f2>	copy a file from one place to another	cp foo fee
mv <f1> <f2>	move a file from one place to another	mv foo fuu
rm <f1>	remove a file	rm fuu

Remember the special directory aliases we covered above and figure out what these commands would do:

```
cp foo ..
mv fee ..
cp foo ~
touch ../tmp
mv ../tmp .
```

The only thing left is to remove directories. There are two ways to do that:

rm –r <directory> will recursively remove everything in a directory (that directory, all of its files, and all of the directories underneath it.

rmdir <directory> will remove a directory, but only if it is empty.

You need to be careful with -f option since it won't double-check with you if you really want to delete/remove files/directories.

Your First C Program

Using the editor you selected, edit a file named hello.c and make it contain this code:

```
#include  <stdio.h>

int main( )
{
    printf("Welcome to CS2!\n");

    return 0;
}
```

* C is case-sensitive.

Linux contains many tricks to make repeating commands easier. You can use up and down arrows to scroll through the commands you have entered recently. Also, you can use the ! (which we pronounce "bang") to get the most recent command that started with something. For example, !gcc would give you the most recent command that started with gcc

Compile this program using this command:

gcc hello.c

That will create a default executable file named a.out

that you can run using:

./a.out

Remember that the "." means the current directory.

If you would like to specify the name of the executable file the compiler creates, you can compile with this command:

gcc –o hello hello.c

which would name the executable file "hello" (which you can change to be whatever you like).

Standard C Format in CMSC111

```
/* CMSC111 Computer Science II
    Lab 2 Getting to Know Linux: lab2a.c
    Programmer: Your Name
    Professor: Dr. Lee
   File Created: Jan 29, 2025
   File Updated: Jan 29, 2025
*/

#include <stdio.h>

#define N 100

int main()
{
   int i;
   i = 5;                                          // You can also declare and assign value as int i = 5;
   printf("Yes, this is your fantastic playground!!!\n");
   printf("What is N? %d\n", N);                   // print out the constant N on the screen
   printf("What is i? %d\n", i);                   // print out the value of integer i on the screen
   printf("What is N * i? %d\n", N*i);             // print out the value of N*i on the screen

   return 0;

}
```

When you have command line inputs:

int main(int argc, char *argv[]) OR

int main(int argc, char **argv)

integer type argc returns the number of arguments

character string argv[] has each argument in the position starting 0, 1, 2, ...

argv[0], argv[1], argv[2], etc.

```
/* CMSC111 Computer Science II
    Lab 2 Getting to Know Linux: lab2b.c
    Programmer: Your Name
    Professor: Dr. Lee
   File Created: Jan 29, 2025
   File Updated: Jan 29, 2025
*/

#include <stdio.h>
int main(int argc, char *argv[])
{
   int i;
   printf("Input?  ");
   scanf("%d", &i);                          // reads an integer; stores into i
   printf("Value in i? %d \n", i);

   printf("argc? %d\n", argc);               // print out the value of argc
   printf("argv[0]? %s\n", argv[0]);         // print out the value in argv[0]
   printf("argv[1]? %s\n", argv[1]);         // print out the value in argc[1]
   printf("argv[2]? %s\n", argv[2]);         // print out the value in argc[2]
   return 0;

}
```

How to Compile

gcc -o lab2b lab2b.c

How to Run

./lab2b What Would Be