Lab Assignment #1

Due date: Sunday, Jan 17, 11:59pm.
Purpose: demonstrate a number of unix commands, edit and run your first C program

Getting started

Log in to a lab machine with your CSE account. Your CSE username is the same as your MSU NetID. If this is your first time logging in, your password is your student PID (A12345678). When you log in for the first time, you are required to change your password. If you've never done do, please follow the steps described at http://www.cse.msu.edu/?Pg=113&Col=2 to complete the activation process.

Some Unix Commands

Open a terminal window on your screen, do Applications/Accessories/Terminal. Type the following commands and observe the output.
First type:

```
script session_lab1.txt
```

The script command will record all input/output of this session in the given file, session_lab1.txt.

```
pwd
```

The pwd command prints the current directory. You should get something like: /user/your_username

```
ls
```

The ls command lists the current directory. You should see a listing of the files in your directory. Now try it with the –a option.

```
ls –a
```

Is there a difference from the previous output?

```
touch myFirstFile
```

The touch command "touches" a file, setting the time and date of the file to the current time and date. If the file does not exist, it creates an empty one. Now type the command ls again with more details:

```
ls –l
```

You should see the file you just created. Touch myFirstFile and list again and note the difference in timestamp.

```
touch myFirstFile
ls –l
```

The rm command removes a file. Type the following command to remove that file you just created:

```
rm MyFirstFile
```

Do a directory listing to be sure this worked.
The mkdir command creates a new directory. Type this command:

```
mkdir cse220
```

Now do a listing to be sure it created the directory.
Create another directory called activities

The cd command changes the current directory. Change into the activities directory:

```
cd activities
```

Print the current working directory by typing pwd. Verify you are in /user/your_username/activities
Create a blank file in the current directory by touching mySecondFile
Try to change to the directory summerActivities:

```
cd summerActivities
```

This should fail since summerActivities does not exist.
Now change into the special directory ..

```
cd ..
```

This represents the parent of the current directory. Now print the current working directory. You should be in /user/your_username
Now change into the special directory .

```
cd .
```

This print the current directory. You should remain in the same directory. '.'is a special directory representing the current directory, so changing into '.' keeps you in the same directory

The rmdir command removes a directory. Try to remove the directory activities:

```
rmdir activities
```

The rmdir command will only remove an empty directory. If you have content in the directory it will fail with a message like "failed to remove stuff: Directory not empty". In that case, you can remove the directory using this command:

```
rm –r
```

Verify activities has been removed.
Try typing the following command:

```
cd ~/cse220
```

After each command type pwd to be sure you know what it does.
You have now typed a complete path. We separate the directory names with a "/" character. This command takes you to the directory cse220 which is a child of the home directory. You should be able to type this again and it should work, leaving you in the same directory.

```
cd ../cse220
```

This moves you to the directory cse220 as a child of the parent directory. This will be the same directory.

```
cd ../cse220/.././cse220
```

Remember that '..' means parent directory and '.' means the current directory. Do you know what this command is doing?

If you are not already, move into the cse220 directory. Create a directory called lab01. Change into directory lab01. Type the following:

```
ls /user/cse220/labs
```

This lists the content of directory labs. You should see a file HelloWorld.c. Copy it into your lab01 directory:

```
cp /user/cse220/labs/HelloWorld.c .
```

The cp command copies a file from labs into the current directory. List the content of lab01 to make sure you have successfully copied this file.

Rename this file to MyHelloWorld.c

```
mv HelloWorld.c MyHelloWorld.c
```

Again, do a directory listing to confirm. View the content of this file:
To display the first 5 lines, type:

```
head -5 MyHelloWorld.c
```

To display the last 7 lines, type:

```
tail -7 MyHelloWorld.c
```

Now list the whole content:

```
cat MyHelloWorld.c
```

End the session recording by typing the following: (All the commands between script and exit and their output will be saved into the output file).

```
exit
```

Copy the file /user/cse220/labs/Infinite.c into your lab01 directory.
Compile it and run it:

```
gcc –o infinite Infinite.c
./infinite
```

Remember gcc compiles the program, the -o switch tells it to create an output file named "infinite" instead of "a.out". The second command executes the program.
You should notice that your program runs endlessly and you won't be able to use the terminal anymore.
Opens another terminal window and list the most CPU intensive processes:

```
top
```

You will likely see infinite right there. Press q to quit top. Type this command in the working terminal window:

```
ps u
```

You should see a list of running processes, each with a process id (PID). Using the PID for the infinite process, stop your infinite program from running:

```
kill infinitePID
```

Now do a ps again and infinite should be gone. Look at top again and it should not show that process anymore.

Editing your C program

Now type the following commands to compile and run this program:

```
gcc –o sayHello MyHelloWorld.c
./sayHello
```

Note the ./ means the file to execute is in the current directory. It won't work without it (try it).
At this time, start the program gedit, either from a menu or from the command line using this command:

```
gedit &
```

Open the file MyHelloWorld.c. This is the C program you ran earlier. It should look like this:

```
#include <stdio.h>

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

```
}
```

Change the program to print "Hello from *your name*\n" instead of “Hello World!!!\n”.
Save your changes, compile with executable sayHello and run.

Handin

Submit through the handin the following files: session_lab1.txt, MyHelloWorld.c , sayHello.

The “handin” system has options to allow you to review your files online and to download them. You Should always verify that you submitted the correct files and they were received by the handin system. You can submit files as many times as you like for a particular assignment. Handin will only keep the last version of each file. Remember to submit your files prior to the deadline as you won’t be able to use handin if the deadline has passed.