Lab 1: Hello, Linux!

In this lab assignment, you will:

1. Learn some basic UNIX/Linux commands
2. Use the built-in Linux manual (man pages) to look up certain functions and commands
3. Try to learn and use an editor to create and write files
4. Compile a "Hello, Linux!" program using the C standard library

---

You will need a Linux machine (or a MacOS, but not Windows) to complete the lab.

Please complete the required exercises below. Note that you need to complete these exercises one by one – ***following the exact order***, as there are dependencies between them. As you work through *each* of these exercises, please record your answers to the exercises in a report. For exercises with typing a command (highlighted in red), please answer the following questions:

1. What does this command do?
2. What is the result you get from typing this command? ***Take a screenshot of the terminal***. Only need to report the first few lines, if the output is long.

When finished, ***submit your report in the PDF form to Canvas***.

Make sure that the name of each person who worked on these exercises is listed in the first answer, and make sure you number each of your responses so it is easy to match your responses with each exercise.

This lab has 2 points in total.

Required Exercises:

1. First we need to have the Linux terminal access. Open the terminal of your laptop or remotely connect to a Linux server.

    As the answer to the first exercise, list the names of the team members on this lab.

2. You can refer to the manual pages to see details and options for each command asked below. Manual pages let you read information about various commands on UNIX/Linux systems; to use them, run

    man *command*

where *command* is the command you are interested in. If you are unfamiliar with manual pages, you may want to try running

    man man

for information on the man command itself. Keep in mind that the manual pages for basic commands vary from system to system (much as the commands themselves do).
Note that you can type ‘q’ to quit the manual pages.

3. Type command:

```
uname -a
```

4. Type command:

```
top
```

5. Type command:

```
cd /bin
```

Note that the /bin directory contains many of the executable commands for the system.

6. Type command (Note that -l is the letter "l", not the numeral "one ".):

```
ls -l
```

7. In “The Unix Time-Sharing System” paper, the authors explain that the following two commands are functionally equivalent (except that you have to remove the temp file afterwards in the second case):

```
ls | head -1
ls > temp; head -1 < temp
```

First answer what do ‘|’, ‘>’, ‘<’ and ‘head’ do?
Then try to type these two lines of commands here. What do you get for each line?

8. Type command:

```
cd
```

Notice that this time we do not give a particular directory. So which directory are we in?
Type command:

```
pwd
```

9. Type command:

```
mkdir tmp_dir_for_lab1
```

Now type:

```
cd tmp_dir_for_lab1
```

10. In “The Unix Time-Sharing System” paper, the authors explain that a user can type two commands together in parenthesis separated by a semicolon, and redirect the output to a file. The file will then contain the concatenation of the two commands. First try this command:

```
(date ; ls) > temp1
```

What does ‘;’ do?
Now type command:

```
cat temp1
```

Next try this command instead:

```
(date ; ls) > temp1 &
```

What does ‘&’ do?
Now try this command:

```
(date & ls) > temp2
```

Now type:

```
cat temp2
```

Answer why temp1 and temp2 are different?

11. Now we will try to use an editor to write a command file. There are two widely used world wide: vi (vim) and emacs. See this wiki page, if you are not familiar about them:

https://en.wikipedia.org/wiki/Editor_war

You will need to learn how to use one of them. The command below uses vi as an example. Type command:

```
vi run.sh
```

Now type letter 'i', then copy and paste the following lines into the terminal:

```
#!/bin/sh

echo y
sleep 10
echo n
```

To exit vi, first type the *escape* key (usually marked 'esc' on the keyboard). Then type ':wq' and type *enter* key. These together creates a file 'run.sh' with the above contents inside.

12. Type command:

```
chmod a+rx run.sh
```

Now type the following two lines of commands (may need to be patient to get them finished):

```
(./run.sh ; ./run.sh) > temp3
(./run.sh & ./run.sh) > temp4
```

Now type command:

```
diff temp3 temp4
```

What does 'diff' do? Why temp3 and temp4 are different?

13. Type command:

```
echo $PATH
```

14. Now we will write a simple c program and practice how to compile it. Create a file with name 'hello.c' and content as following:

```
#include <stdio.h>

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

    fprintf(stdout,"Hello, Linux!\n");

    return 0;
}
```

Answer what '#include <stdio.h>', 'fprintf'and 'stdout' are for? We can 'man fprintf' for manual as well.

You can compile your program with the command:

```
gcc -o hello hello.c
```

Run your program by typing "./hello". If your program runs correctly, then take a screenshot of your terminal output as the answer to this exercise.

15. Type command:

```
cp temp2 temp5
```

16. Type command:

```
rm temp*
```

In case you'd like to start all over again or remove all the files generated from this lab, then you can do the following:

```
cd ..
rm -r tmp_dir_for_lab1
```

17. Finally, find and execute a new command that is not used above. Take a screenshot of your terminal output and explain what this command does.