Lab Assignment #1(a)

Instructor: Prof. Ioannis Savidis, isavidis@coe.drexel.edu
TA: Shazzad Hossain and Vaibhav Venugopal Rao, {msh89, vv85}@drexel.edu

Drexel University
Electrical and Computer Engineering
September 20-21, 2016
(Tuesday-Wednesday, Week 1)

1 Objective

This handout aims to familiarize you with the LINUX operating system that will be used throughout this course. The major goals are to:

- Provide an overview of LINUX,
- Learn basic Shell concepts,
- Learn basic definitions and usage examples for common LINUX commands.

2 LINUX overview

LINUX is a large, diverse and very rich operating system. LINUX is memory-efficient, easy to maintain and promotes efficient program development. Unfortunately, LINUX is *not* as user friendly as some other popular operating systems! Keeping this in mind, this tutorial is organized so that beginners can start with simple functions and progressive learn about the LINUX system on their own.

Please try the examples at your workstation as you read through this text. The best way to learn about the LINUX system is by personally using it. Soon, you will be an experienced user in a general-purpose multi-user LINUX system!

2.1 LINUX

An operating system, abbreviated as OS, is a collection of programs that coordinates the operation of hardware and software. LINUX is one kind of an OS, which is basically broken down into three components: Scheduler, File System and Shell. You will only interact with the shell, so lets start running some basic LINUX commands now.

2.2 Shell

A LINUX shell, also called “the command line”, provides the traditional user interface for the LINUX operating system and for LINUX-like systems. Users direct the operation of the computer by entering command input as text for a shell to execute. Within the Microsoft Windows suite of operating systems the analogous program is command.com, or cmd.exe for Windows NT-based operating systems.

Since in the LINUX operating system users can select which shell they want to use (which program should execute when they login), many shells have been developed. It is called a “shell” because it hides the details of the underlying operating system behind the shell’s interface. (In contrast with the “kernel”, which refers to the lowest-level or “inner-most” component of an operating system).

3 Basic LINUX Commands

In this section we will discuss the basic commands you are most likely to use in this quarter.

3.1 Log into LINUX

First, you should pop up an XWin32 (or X-term or terminal/console) window, which is found in `startup menus->All Programs->XWin32->X-Win32`. Then in the toolbar (bottom right corner of desktop) there will be an XWin icon that shows up. Right click on the XWin icon and choose `X-Config`. A window will emerge. In `Sessions` tag of the new window, choose `Wizard`, which produces an additional wizard dialog box. You can select any name to fill in `Name` blank, and choose the `ssh` in `Type` option then click `Next`. In the next dialog, type `"xunil.coe.drexel.edu"` in the `Host` field. Then the system will ask you to input your username and password. Input the username and password you were provided. Your user name should be your DrexelOne id (e.g. jl597), and your password should be your DrexelOne password. Click `Next` and choose `Linux` as a `Command`. Click `Finish`, creating a new session. Choose the session you just created, and click `Launch`. You will be logging into the xunil server. A command window should emerge. This is the command prompt. You will be entering commands in the line with $ prompt.

After you login, you should immediately change your password. Type `passwd` to change your passwd, following the instructions the are provided. Remember that you will not see anything on the terminal when you type in your old and new password.

3.2 Work with Directories

First, I will introduce you to some commands which manipulate the directory structure:

3.2.1 List Content of the Working Directory

Type `ls` in prompt line. This command displays the files in the current working directory. For example:

```
$ ls
$
```

Most commands in LINUX can accept arguments. Try `ls -l` (long listing). `-l` is an option type argument, which provides much more information about a file than the simple version. Example:

```
$ ls -l
total 2
-rw-rw-rw- 1 jianchao other 138 Apr 5 19:34 README
drwxr-xr-x 1 jianchao other 512 Apr 2 19:33 routes/
$
```

This listing displays additional information regarding your files. The first entry on the left describes the permissions on the file. Additional information includes: Jianchao is the owner of the file, other is the group, the number following the group is the file size, which is followed by the date, and finally the file name. Do not worry if you do not understand what all of these mean, and proceed to see the name and the size of files.

3.2.2 Display Current Working Directory

Type `pwd`, this command displays your current working directory.

LINUX has a file system where files are stored on storage devices such as disks, and file directories are organized in a logical and structured fashion. Directories could have files, programs, and subdirectories in them, like in all usual file systems. Every user has his/her own home directory, where no one but the user can write to. In your current working directory, you could create other directories, and store or delete files. Note that our space is limited, so be neat in allocating space for storing your work.

3.2.3 Change Working Directory

The command used to change the working directory. Type `pwd` to see your working directory. Now type `cd directory`. Use `pwd` again to see the new working directory.

In general, `cd directoryname` is used to change to another directory and cd .. to pop out of a directory.

Type `cd ..` (note the space between cd and ..) We are back in our home directory, check typing `pwd`.

3.2.4 Make Directory

The `mkdir` command is used to create a directory in your current working.

```
$ mkdir yourdir
$ ls
```

A directory named “yourdir” is created in your current working directory, in the hierarchical structure.

3.2.5 Remove Directory

This command is used to remove directories from the system.

```
$ rmdir yourdir
$
```

The directory is now removed (If it were not removed already using rm -r command). “rmdir” is preferably used with -rf arguments, this deletes the subdirectories and files within.

NOTE: Check -i option for `rm` and `rmdir`.

3.3 Working with Files

The commands that are used with files, like file copy, remove, and so on were discussed in the previous subsection. The subsection below describe the operation of the text editor vi. You can also choose an alternative text editor, such as `emacs` or `gedit`.

3.3.1 Create a File using `VI`(A Text Editor in LINUX)

First, we use `vi`(A text editor in LINUX) to create a txt file, the procedure is:

(1).Create a File using vi

If you write the command

```
$ vi my_file
$
```

you will see the screen with a column of tildes. The vi editor is now in the so called command mode.

The two basic commands are:

- i–Insert text to the left of cursor
- a–Insert text to the right of cursor

Since you are at the beginning of an empty file it does not matter which of these you type. Write the text: `I am a graduate student. My name is` *`name`*`.`

(2).Cursor Movements Commands

You need to be in the command mode. If you do not know what the actual operating mode is, press `esc`. The `esc` keystroke always turns the editor into the command mode. Then you can move along the screen, and can keystroke the following commands:

- h–Insert text to the right of cursor
- j–Cursor is moved one line down
- k–Cursor is moved one line up
- l–Cursor is moved one space to the right

(3).Deleting Text

If you are in the command mode then

- x–Delete one character at the cursor
- dd–Delete one line where the cursor is placed

(4).File Saving

You must be in command mode. You can then use several tricks to save the file:

- :x–Write file to the disk and finish
- ZZ–Write file to the disk and finish
- :w–Write file to the disk and continue

(5).Replace Mode

The replace mode is very useful. The replace mode permits an overwrite of existing text.

- :r–Replace one character over the cursor
- :R–Overwrite text until the next action (e.g.keystroke of `esc`)

(6).Quit `vi` After editing, you may want to quit `vi`. The commands for quitting are

- :q!–quit vi, do *not* write file to disk
- ZZ–quit vi, write file to disk

3.3.2 Display File

If you write the command `more` and a name of a file, then the file will be displayed. For example:

```
$ more filename
$
```

User can control the output:

- press space...the next screen is displayed;
- press enter...the next row is displayed;
- press q.......the command is finished.

3.3.3 Copy File

The copy file command is used to make a copy of a file. You can copy the file you just made in your directory. Type

```
$ cp filename.txt anothername
$
```

The first argument is the name of the original file for copying (source file), and the second argument is the new location to place the copy the save (destination file). Type `ls` to see the copy file. Full path names could be used as source and destination files.

3.3.4 Rename and/or Move the File

This is used to move a file between directories and/or rename a file/directory.
To rename, type:

```
$ mv filename1 filename2
$
```

Check the results. Now type

```
$  mv filename1 ./directoryname/filename2
$
```

Switch to `sampledir` and see the file renamed under `sampledir`. You could leave any_name place empty, and just carry the file into the directory, leaving the name the same. Now rename the sampledir as enoughdir by typing:

```
$ mv sampledir enoughdir
$
```

3.3.5 Remove File

The command removes the file from the system.

```
$rm file
$
```

If you use a wildcard, for example,

```
$rm h*c
$
```

you will remove all files beginning with `h` and ending with `c` which are in the current working directory. If you write

```
$rm *
$
```

you will erase all files from your working directory. If you write

```
$rm -i*
$
```

will also remove all files but, the system will ask for permission before removing each file. The command

```
$rm -r yourfile
$
```

removes the directory *yourfile* even if this directory is not empty.

3.3.6 Change Permission

This command is used to change the permissions over a file that you own. Type `ls -l` to see the permissions of the files stored in your current directory. The leftmost column indicates whether the file is a file(-), directory(d) or a link(l). "b" and "c" can also appear in this position, if the file is a special file used to control a hardware device. This is not our point of interest. The remaining nine characters in the first strings describe the permissions or mode of the file.

A file can have three sets of permissions: user, group, and other, which are abbreviated, respectively, u,g, and o. Each of the three groups has three parts: read access, write access and execute access. Read means the subject (u,g or o) can read the file, write means the subject can edit the file, and execute means execute the file as a command.

For a directory, the meaning of read, write, and execute are slightly different: read access means that the subject is allowed to look at the contents of the directory (with ls for example), write access means that the user can create a file in the directory, and execute means the user can go through the directory searching through the subdirectories.

The LINUX system provides the chmod command to change the permissions of a file that you own. The syntax for chmod is user class (u,g, or o), followed by the given action (- or +), and finally by the permission to change (r,w,or x).

Now create a file and a directory, using the cat and mkdir commands and see the default permission status.

```
$ chmod u+x <created_file_name>
$ chmod o-rx <created_directory_name>
```

The chmod can also take a numeric argument that describes the user class and permission to change as a sequence of bits. Try:

```
$ chmod 466 <created_file_name>
$
```

3.3.7 Running Processes Information

This is used to give information on the running processes. Type:

```
$ ps -al
$
```

Two command arguments are; -a means all, and -l denotes long listing. PID denotes the Process ID number.

If a process is stuck, for instance the netscape browser, you can identify that application from the ps list, and stop that execution by using the kill command.

3.3.8 Stop Execution

Usage: kill [PID of the running application] Do not try to experiment with this command right now, but remember that there is such a command to stop running programs.

3.3.9 Find File

The command

```
$find /usr -name filename
$
```

finds a file `filename` in subdirectores of directory usr.

3.4 Other Important Commands

There are additional commands that are also important, like get help, change password and so on.

3.4.1 Get help for LINUX

This command is the help command for the LINUX environment. For example, type:

```
$ man ls
$
```

The `man` command formats and displays a page (actually lots of pages) on the usage of the ls command and its arguments. Try to determine what the `ls -al` command does. The `man` command provides an extensively useful online tutorial for LINUX. If you are interested, you can take your time to go through some popular commands, but be careful of how much time you spend. Remember you can even type `man man` for information on the usage of the `man` command.

3.4.2 Password

After logging in, you can change your password by typing the `passwd` command. You will be required to enter your old password and enter the new password twice.

3.4.3 Leaving the System

It depends on the kernel. Usually ctrl-d exits the kernel, but in a lot of kernels you must use a special command.

```
$ logout
$
```

or

```
$ exit
$
```

When you complete your work, don't forget to logout.