

1 Introduction to Linux

We are just going to go through some of the basics of Linux in this section. As with any new operating system, the more you use it the more proficient you will become. Most flavors of Linux have a wealth of applications that are either free or open source.

Linux developers assume that the users of Linux are a little more sophisticated, hence there are more options to users and things are more open to being changed. On the other hand, it is also easier to make a change that you do not want and may be difficult to reverse. There are very active online communities of Linux users and finding a solution to a particular problem is relatively easy with a few searches.

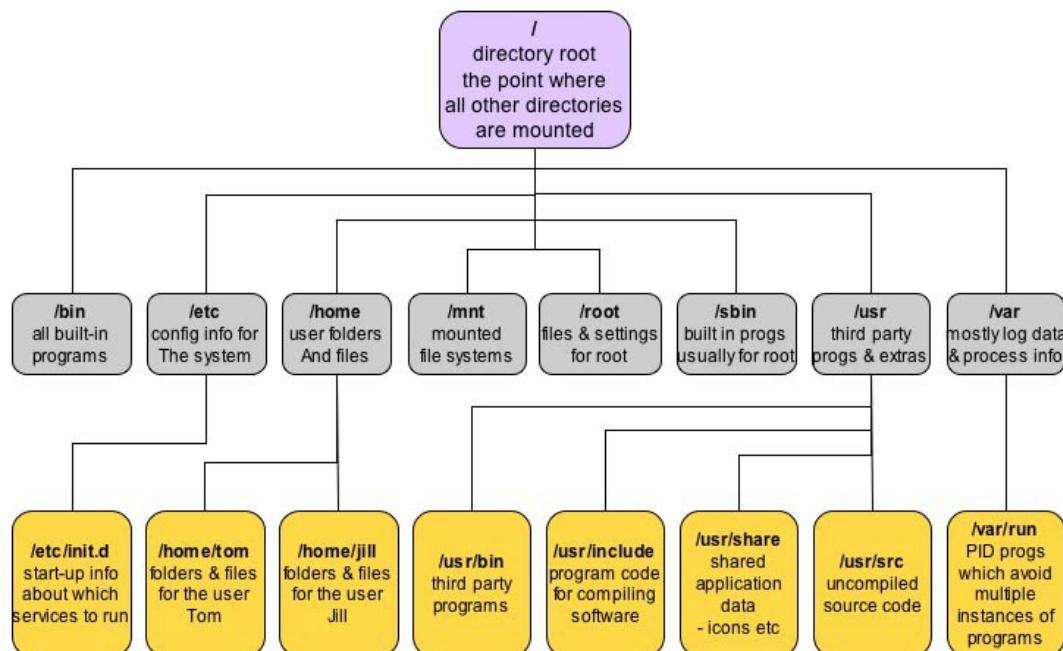
Linux distributions have many graphical user interface programs, including LibreOffice (free word processor, spreadsheet, presentation package, drawing package, and database), GIMP (free graphics editor), file managers, VLC (free media player), Kate and gedit text editors, Firefox and Chrome web browsers, ... that are fairly easy to use and figure out through the menu.

In addition, Linux has a terminal interface (command-line control) that, unlike Windows, is actually useful. We will be exploring some basic Linux commands and discuss the Linux file system.

1.1 File Structure in Linux & Directory Navigation

The file structure in Linux is closer to that of the Mac than it is in Windows.

A Typical Linux File System



- The first thing we will do is change our password, if you have not already done this. In the terminal run the command

```
passwd
```

You will be asked for your current password, then your new password (twice). Note that on some Linux systems you will see `***` as you type and on others you will not.

- In the terminal run the command

```
pwd
```

this stands for print working directory. You will see home and then your user id. Linux was designed like Unix, a multi-user system, so each user on the system has their own space inside the home directory.

- Run the command

```
ls
```

which stands for listing, that is a directory listing.

Most commands have options that allow you to run the command to produce different levels of output or to specialize the running in some way. For example, the listing command has

```
ls -l
```

for a long listing,

```
ls -a
```

for listing all files (even those that are hidden, like `.*` files),

```
ls -al
```

a long listing of all files, and there are more options. For example, `ls -l` on my machine gives me the listing,

```
don@HAL9000 ~ $ ls -l
total 34
drwxr-xr-x 2 don don 4096 Oct  5 12:29 Desktop
drwxr-xr-x 3 don don 4096 Oct  5 13:35 Documents
drwxr-xr-x 2 don don 4096 Oct  5 12:29 Downloads
drwxr-xr-x 2 don don 4096 Oct  5 12:29 Music
drwxr-xr-x 2 don don 4096 Oct  5 12:29 Pictures
drwxr-xr-x 2 don don 4096 Oct  5 12:29 Public
drwxr-xr-x 2 don don 4096 Oct  5 12:29 Templates
drwxr-xr-x 2 don don 4096 Oct  5 12:29 Videos
```

This shows the permission structure, the user and group membership, size, date, time, and file/directory name.

Some flavors of Linux will set up aliases for these and other commands, that make them easier to use. For example, Mint defines the commands `l` (for `ls`), `ll` (for `ls -l`),

and `la` (for `ls -a`). You can define your own aliases by editing the `.bashrc` file on some systems, I would not recommend doing that at this point.

Many Linux commands take wildcards, one of them is the common `*`. So if you run the command `ls *.jpg` the listing will include all of the JPEG image files in the directory.

- There are file explorers in Linux that allow you to navigate the file system just like in Windows Explorer or Finder. In addition, you can open documents, create directories, remove files and directories, everything you would expect from a file system program.

You can also navigate the file system using terminal commands.

- The `cd` command is to change the current directory. Run

```
cd Documents
```

now you are in the Documents directory. Run

```
pwd
```

Also, do a listing, you will probably not see anything since there are no files or subdirectories in this directory. The command `cd ..` will move you up one directory, run,

```
cd ..
```

Also, print the working directory and get a listing. The command `cd ~` will move you back to your home directory.

- Now we will create a subdirectory to store all of our work for this class. The command `mkdir` creates a new directory, make sure you are in your home directory and run

```
mkdir COSC220
```

Do a listing, you should now see a new directory `COSC220`.

- Linux also has an auto-completion feature. Navigate back to your home directory, then type in `cd CO` then hit the Tab key, it should complete the directory name of `COSC220`, then enter will change your directory.
- The terminal window also keeps a history of the commands you type in. So to go back and rerun a command or put it in the command line for editing, use the up and down arrow keys.
- If you are running a program in the terminal and are either inside an infinite loop or the program is not responding then `Control+Z` will abort the execution of the program. To test this, in the terminal run the command `yes`, this will go into an infinite loop of printing “y” to the screen, now type `Control+Z`.
- You can close the terminal like any other window or you can run the `exit` command.

1.2 Permissions

Linux uses a permission structure to control access to files and directories. If you do a long listing of a directory you will see entries like the following.

```
drwxrwxr-x 4 don don 4096 Oct 29 11:22 ASM/
drwxrwxr-x 3 don don 4096 Oct 29 11:26 CPP/
-rwxrwxr-x 1 don don 648 Oct 5 13:40 helloworld*
-rw-r--r-- 1 don don 108 Oct 26 20:36 main.cpp
-rwxrwxr-x 1 don don 9216 Oct 26 20:37 prog*
```

The first 10 characters are the permission structure for the file or directory. The first character is either a d for directory or - for a file, there are other possibilities but it is usually these two. The rest of the permission structure is blocks of three characters rwx. The r stands for read, w for write, and x for execute (the file is a program that can be run). The blocks of three are as follows, the first block is the user block (you), the second block is the group (each user belongs to several groups of users on the system and each file belongs to a particular group), the third block is other (everyone else on the system). For example,

```
drwxrwxr-x
```

is a directory, the user can read files, write files and execute programs in the directory. Any member of the group can read files, write files and execute programs in the directory. All other people on the system can read files and execute programs in the directory but they cannot write to the directory, which includes deleting files, this protects you from other people on the system and their actions.

Another example,

```
-rw-r--r--
```

This is a file that the user can read and write to (i.e. edit). All other users on the system can read the file but cannot alter its contents. You can change the permissions of the files you own through the file managers or using the command `chmod` in the terminal.

- Navigate to the COSC220 directory, create a subdirectory called Lab01, and navigate into the Lab01 directory. We could create a file using a text editor or file manager, but we can create one from the terminal as well. Run,

```
touch test001.txt
```

Now get a long listing. What permissions are on the file? The `chmod` command syntax is of the form

```
chmod ugo+rwx <filename>
```

The u is for the user, g for the group and o for other. The r is read, w is write, and x is execute. The + between them means that we are giving these permissions to the users

and a - would be that we are taking away the permissions from the users. You can use any combination of ugo and any combination of rwx. Run the following commands and get a long listing after each to see how the permissions changed.

```
chmod ugo+rwx test001.txt
chmod go-x test001.txt
chmod g-rwx test001.txt
chmod ugo-rwx test001.txt
chmod u+rwx test001.txt
```

1.3 man Pages

Ever since the creation of Unix, all commands had help screens that documented in detail the usage and options of each command. These help screens are called man pages (short for manual pages). In the terminal run the command

```
man ls
```

This will list the man pages for the ls command. You can use the arrow and paging keys to scroll through the documentation and when you are done hit the q key to quit.

1.4 Other Commands

Here are some more Linux commands that are used inside the terminal, some are useful and some are just for fun. You can learn more about the commands by looking at the man pages for each of them. We will add to this list as the semester goes on.

- time — displays the execution time of a program.
- cp — Copy files or directory.
- rm — removes files.
- rmdir — removes directories.
- mv — moves or renames files.
- lscpu — returns information about your CPU. Since we are running Linux through a virtual machine you will not get the same information as you would if Linux were installed directly.
- cal — shows a calendar for the current month. You can also use `cal <year>` to get the yearly calendar for that year.
- fortune — try the command `fortune`.

- `cowsay` — try the command `cowsay I love COSC 220`.

Since we have our file `test001.txt` lets do a few of these commands with it.

- Run the command `cp test001.txt test002.txt` then get a directory listing. This will make a copy of `test001.txt` stored as `test002.txt`.
- Run the command `mv test001.txt test003.txt` then get a directory listing. This will move (really rename) `test001.txt` as `test002.txt`.
- Create a new directory named `testdir`.
- Run the command `cp * testdir` then `cd` into `testdir` and get a directory listing. You should see both files in the new directory.
- Go back one directory with `cd ..`.
- Get a directory listing or do `pwd` to make sure that you are in the correct directory.
- Run the command `rmdir testdir` you should get an error that the directory is not empty and it did not remove the directory. This is a good catch since you should be careful about removing entire directories.
- Run the command `rm -r testdir` and get a directory listing. This time the directory will be removed. Use with caution! One thing to note is that if you use a GUI file manager deletions will be saved on your trash and you can get them back but if you use the terminal commands like we did here the files and directories are not saved to trash.

1.5 Command-line Compiling

Most IDE's, like Eclipse, Visual Studio, Code::Blocks, ..., have the ability to compile and run a program by using a menu option or a toolbar button. When you select these options what is really happening under the hood is that a command-line command is being run on your system. In fact, if you watch the compiling output window in Code::Blocks you will see the actual compiling commands being executed.

Linux has many such IDE's available and some IDE's like Eclipse and Code::Blocks are cross platform and will run on several operating systems. Since one of the goals of this course is to become more familiar with Linux and command-line operations we will be compiling programs using the compiling commands in the terminal. Later we will be using Makefiles to do more complicated compiling.

1. Navigate to the Lab01 directory and create the file `hello.cpp` using the `touch` command.
2. Open the file in a text editor.

- (a) Kate: Graphical user interface, easy to use and can embed a terminal window at the bottom. Either run the command `kate` from the terminal or find Kate in the menu system.
 - (b) nano: Text based interface, easy to use. Run the command `nano` from the terminal.
 - (c) gedit: Graphical user interface, easy to use. Either run the command `gedit` from the terminal or find gedit in the menu system.
 - (d) emacs: Text based look with a steep learning curve. Run the command `emacs` from the terminal.
 - (e) vim: Text based look with a steep learning curve. Run the command `vim` from the terminal.
3. Create the standard Hello World program in C++. Not that you should need the code for this but it is below.

```
1 #include <iostream>
2
3 using namespace std;
4
5 int main()
6 {
7     cout << "Hello World!" << endl;
8     return 0;
9 }
```

4. Open the terminal, if it is not already open, and make sure that the directory is the Lab01 directory, if not then navigate to it. Compile the program by running the command,

```
g++ hello.cpp
```

Now get a long listing and notice what has changed. You should see a new file `a.out` and it should have executable permissions on it. Run the program using the command,

```
./a.out
```

Note that `a.out` is the default name of the executable file. Also, unlike Windows, the extension of the filename does not have to be `exe` for an executable file. To change the name of the output file you use the `-o` flag for the compiling command. Run

```
g++ hello.cpp -o helloworld
```

Now get a listing and you should see the new `helloworld` file that is executable. Run it the same way, with

```
./helloworld
```

2 Submit Your Work

Put your name and date at the top of the program as a comment. You will upload the files for each lab in a single zip file. In the terminal, navigate to the COSC220 directory. Run the command,

```
zip -r Lab01.zip Lab01
```

This will zip the entire contents of the Lab01 directory into the zip file Lab01.zip. Using a browser, go to the MyClasses page for this assignment and upload the Lab01.zip file for your submission.

Some file managers allow you to right click a directory or selection of files and select “compress” or “archive” to create a zip file. From the command-line,

1. To zip a single directory into a single archive called “myzipfile.zip”,

```
zip -r myzipfile.zip DirectoryName
```

2. To zip multiple files into a zip file called “myzipfile.zip”:

```
zip myzipfile.zip file1 file2 file3 file4
```

3. You can also use wildcards in these commands, for example,

```
zip myzipfile.zip *.cpp
```

Will zip up all of the files with the cpp extension and nothing else.

3 Installing a Linux Virtual Machine

You will probably not want to use the campus computer labs for all your programming exercises in this class, hence you will want to install Linux on your personal computer. There are a number of ways to do this but the easiest is to create a Linux virtual machine. For instructions on doing this please see the link “Linux VM: Installing Linux Mint as a Virtual Machine” under “Class Resources” on the MyClasses page for the course. You are welcome to install a different distro of Linux if you would like. I chose Mint since I thought it would be the easiest one for you to install and use. If you go with a different distro I would recommend either Ubuntu or Debian.

Running virtual machines is a common tool in software development. In most circumstances when you are creating software you want to test and deploy it to multiple operating systems, at the very least Windows, Mac, and Linux. Much of this testing and creation of setup programs can be done easily through a virtual machine without the need to have multiple computers with separate operating systems. Personally, I use Linux as my primary operating system and when I deploy software to a Windows, I use a Windows VM to test the software and create a setup program. You can also run several VMs at one time and hence test network applications on a single machine. Another use that I have done in the past is to set up a Linux VM as a web server and develop php based websites on the host side as a

client-server setup. This made development much faster as I did not need the physical server that the web site was ultimately going to be deployed. This allowed me to test and debug my code on a private machine before the site went public.