Linux and High Perfomance Computing Workshop

Introduction To Linux

Session Audience:
This session is appropriate for first time Linux users through intermediate Linux users. Some advanced topics will be mentioned though they will not be covered extensively.

Session Goals:
This session will teach basic Linux concepts and introduce common Linux commands. After this session, you should be comfortable using a terminal to accomplish the following: remotely logging into a Linux terminal, directory navigation, directory/file creation and removal, moving files and directories, searching for directories/files, viewing file content, editing files with non-GUI text editors (Vim), viewing and changing file permissions, BASH scripting, compressing and archiving files, setting environment variables, using man pages as a resource to properly use and identify commands.

Manual Pager Utility (Man Pages)

The manual pager utility is an interface to the on-line reference manuals. The manual pager utility, often called the man pages, is very helpful when you need more information about a certain command and its arguments, flags, and options.

Since we would like to learn more about the man command, lets read the manual page about the manual pager utility.

Open a terminal

In the top left corner of the screen click on *Applications*

Navigate down to *Utilities*

In the *Utilities* menu, locate and click the *Terminal* application

Once you are in the terminal, you should see a prompt, followed by the cursor.

Type the following command:

man man

The first “man” in the above command is the actual command.

The second “man” in the above command is the argument to the command.

The command with the argument says that you would like to view the manual page for the manual pager utility.

Press |Enter| to run the command, you should now see the manual page for the manual pager utility.

Scroll through this man page and understand what information is in the manual pager utility.

When you are done, press |Q| to quit.

You may want to use the man pages for other commands as new commands are introduced.

You should now be back at the terminal with a new prompt below the previously entered command.

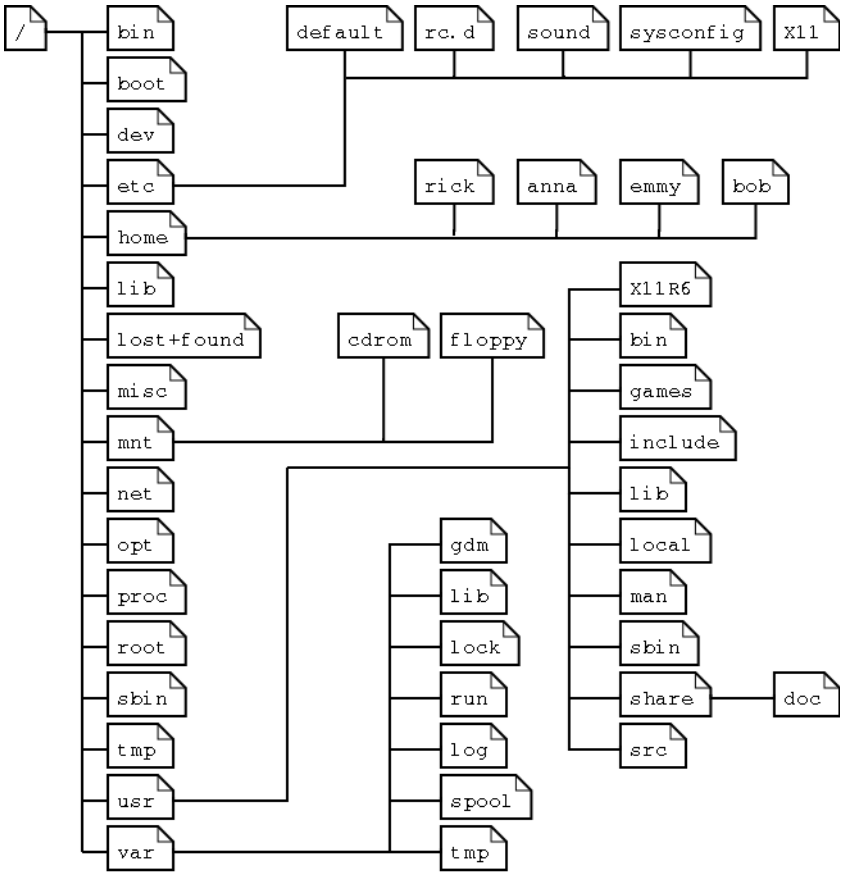
You can clear the terminal at any time by entering the clear command:

clear

Press |Enter| to run the command

The File System

The Linux file system is organized in a tree structure. Generally, the layout will follows the scheme below:



http://www.tldp.org/LDP/intro-linux/html/sect_03_01.html

The directories in the file system can vary from this depending on the operating system, and system administrator.

The root of the filesystem is indicated by the forward slash “/”. This directory contains all other directories and files on the file system.

To accurately reference a directory or file, we can use the full path. This is the path that take you from the root directory to the desired directory or file. As an example, suppose I would like to reference bob's home directory, the path would be /home/bob/

What would be the path for the doc directory? _____

Now that we know how a Linux file system is structured, lets get comfortable navigating the file system.

File System Navigation

Open a terminal if you have not done so already.

Enter and run the command:

pwd

The output for this command is the full filename (the path) of the current working directory. In this case, it should be /home/<username>, unless you have run commands outside of this tutorial.

If you ever need to know what directory you are in, use the pwd (print working directory) command.

Once we know what directory we are in, it is necessary to see what files or directories are in that directory. For this, we will list the directory contents by using the the ls command.

Enter and run the command:

ls

Enter and run the command:

ls <home_path>

where <home_path> is the path generated by pwd

You should see the same output for the two commands above. The ls command will default to list the contents of the current working directory unless a path argument is passed, like it was in the second ls example. Now list the contents of the root directory "/".

Tab Completion

To enter commands and some arguments faster, and more accurately, use tab completion. If you enter enough characters for a command, or supported argument you can hit tab and the remaining characters will be filled in.

Lets try it with the ls command.

Enter:

ls /h

Press |Tab|

You should now see:

ls /home/

The tab completion was able to fill in the directory "home" as there were not any other directories that started with h under the root directory. If there are multiple options for completing a command or argument, you may press |Tab| 2x to display the options.

Press |Enter|. You should now see the list of contents of the /home/ directory.

So far we have listed the contents of other directories but we have not changed from our current working directory.

To change directories we will use the cd (change directory) command.

Lets begin by changing to the root directory:

```
cd /
```

To verify you are in the root directory, print your current working directory (pwd).

Some Navigation Shortcuts

../	up one directory (parent directory)
~	user home directory
./	current working directory
/	root directory

Now change to the /home/ directory this is not the same as the user home directory:

```
cd /home/
```

Now change directories to your home directory, if you do a ls in /home/ you should see a directory with your username (the one you logged in with), this is your home directory. To change to it, we can use several methods, here are a few:

```
cd
cd ~
cd ./<username>              (relative path)
cd /home/<username>          (absolute path)
```

After running one of these, print your current working directory to ensure you are in your home directory.

Now go back to the /home/ directory. This can be done with a couple of methods:

```
cd /home/                    (absolute path)
cd ../                       (relative path)
```

Print your current working directory and ensure you are now in /home/. Use the other three methods of navigating to your home directory, returning to /home/ each time to get comfortable with relative and absolute paths.

When you have tried the four different methods, return to your home directory.

From you home directory, if you enter cd ../../ what directory will you change to? _____
Try it to check your answer, then return to the home directory.

Creating and Managing Directories

Make sure your current working directory is your home directory.

We now want to create a directory within your home directory. To make a new directory, we will use the mkdir (make directories) command. This command needs the path, relative or absolute, of the directory to be made. If only a new directory name is entered as the argument, the directory will be made in the current working directory (the new directory will be a child of the current working directory).

Lets make a “Forest” directory in our home directory. If we are in our home directory, we can use the following commands:

```
mkdir /home/<username/Forest
mkdir ./Forest
mkdir Forest
```

Now list the contents of your home directory to verify that the Forest Directory was created.

We would like to create two new directories as children of the Forest directory. Lets name these directories “Ungulate”, and “Elk”. To make these directories use the following methods:

```
mkdir /home/<username>/Forest/Ungulate
mkdir ./Forest/Elk
```

Make one more directory in the Forest directory named Moose

Now list the contents of the Forest directory to ensure the Ungulate, Elk, and Moose directories were created.

Change your working directory to the Forest directory.

Moving Directories

List the contents of the Forest directory. Oops, Elk and Moose need to be children of Ungulates. We are now need to move the Elk and Moose directories. To do this, we will use the mv (move) command. This command needs two arguments, the source and destination of the directory.

Move the Elk directory as follows:

```
mv ./Elk ./Ungulate/
```

Now move the Moose directory, use the absolute path for the Moose and Ungulate directories this time.

List the contents of the Forest directory and the Ungulate directory to verify the Elk and Moose directories were successfully moved.

We can also rename directories using mv. Lets rename the Elk directory Bison.

To do this, use the mv command with the Elk directory as the source and the path you would like to have for the new directory. Try this on your own.

Creating and Editing Files (using vi)

Move into the Bison directory.

List the contents to verify the directory is empty.

For this section, we will be using Vim which is the improved version of Vi. This is a non-GUI text editor. There are many non-GUI text editors out there and we encourage everyone to try several so you may pick the one that is most comfortable for you. We are covering the basics of Vim, to improve your Vim skills, ARCC recommends: http://linuxconfig.org/vim-tutorial

non-GUI text editors:

Vim	http://www.vim.org/
Emacs	http://www.gnu.org/software/emacs/tour/
nano	http://www.nano-editor.org/

The three non-GUI editors mentioned above are the most common you will encounter. Tehre are others out there and we encourage everyone to explore their options. If you have a GUI, you may be interested in using a GUI based text editor. There are several of these, though we will not be covering them in this session. If you wish to explore a GUI based editor, click on Applications, Accessories, then look for gedit.

To start Vim for a file named aboutBison enter and run:

```
vim aboutBison
```

In the bottom left of your terminal, you should see text “aboutBison” [NewFile]. This means the file is new, and has not actually be saved or written to the storage yet. To show that the file has not written, lets quit vim by entering **:q** (q for quit) and pressing |Enter|.

List the contents of the Bison directory and you should see that no file has been created.

Press the |up_arrow| key until you see the command “vim aboutBison” that you entered earlier. Hitting the up_arrow and down_arrow keys allow you to navigate through previously entered commands, the latest command you entered will be the first listed. Press |Enter| to once again open Vim with the new file aboutBison.

Now we will write the file to save it in the Bison directory. To do this, enter **:w** (w for write) and press |Enter|. You should now have text in the bottom left corner of your terminal the aboutBison has been written.

To quit out of vim we will enter **:q** and press |Enter|.

List the contents of the directory again and you should see that we created the file aboutBison.

To open the file with vim, we will simply issue the same “vim aboutBison” command. You can open this file from anywhere by providing a relative or absolute path to the file.

Open the about bison file.

So far, we have only been using commands (**w**, **q**) as we were in normal mode and pressing |:| switches to command mode. To enter normal mode from any other mode, press |Esc|. Now lets insert text, to do this we need to go to insert mode. To go into insert mode from normal mode press |i|. After you press |i|, you should see that the bottom left corner indicates you are in insert mode.

Type the following:

Name: Bison
Color: Brown
Size: Huge

Using the arrow keys, navigate to the end of the “Name:” line and hit |Enter| to add a new line. Type the line “Food: Grass”. Your file should look like:

Name: Bison
Food: Grass
Color: Brown
Size: Huge

Feel free to add additional text to get comfortable.

Now lets quit out by going back to normal mode (|Esc|), then entering **:q**. You should see a warning that there has not been a write since the last change. If we would like to quit and not save any changes since our last write, you could enter **:q!**. Since you want to write these changes, we will write and quit by entering **:wq**.

Copying files and Directories

We would like to have an aboutMoose file in the Moose directory. To do this, we will use the cp (copy) command. Similar to the move command, the copy command needs a source and destination as arguments. To copy the aboutBison file and rename it to aboutMoose, we will use the following command:

cp ./aboutBison ../Moose/aboutMoose

List the contents of both the Moose and Bison directories to verify the files aboutMoose and aboutBison exist and are in the correct directories. Use Vim to edit the aboutMoose file so it contains correct information, Size: Large.

Change your working directory to the Ungulate directory.

Now we want to copy an entire directory and its contents. We will use a recursive copy by including the -r -R or –recursive option in the cp command.

Copy the Moose directory to a new directory called Deer by one of the following:

```
cp -r ./Moose/ ./Deer/
cp -R ./Moose/ ./Deer/
cp --recursive ./Moose/ ./Deer/          (note the two dashes)
```

Verify that the directory and its contents were successfully copied. Rename the aboutMoose file in the Deer directory to aboutDeer and edit the file so it contains the correct information, Size: Medium.

Viewing File Content

Now we would like to read the content of this file. One method of doing this is to reopen the file in a text editor, this is often more time consuming if you do not have intentions of editing the file.To do so, we will use the cat command. Just a reminder, if you would like more information on the cat command you may want to man cat.

Run the following command:

```
cat aboutBison
```

You should see the text from the aboutBison file.

Another useful program is less. You can use less to read larger (longer) files. Less will allow you to scroll forward and backward through the file output. Less does not have to read the entire input file before starting, so with large input files it starts up faster than text editors like vim.

Add a lot of lines to one of your about files (the content of these lines is not important) and use less to view the file.

```
less about*                    (* is often a wildcard in Linux)
```

Archiving and Compressing Files

In order to make it easy to store and distribute many files, we will archive them using tar. This will combine the multiple files into one archive which can then be restored by using tar.

Navigate to your home directory.

We will archive the entire Forest directory by running:

```
tar -cvf forestArchive.tar ./Forest/
```

The -cvf are options. The c means create an archive, v means verbose (output what tar is doing through the process), and f specifies the archive file.

We could have used the multicharacter options to accomplish the same task:

```
tar --create --verbose –file=forestArchive.tar ./Forest/
```

Removing Directories and Files

Before we untar the files, lets delete the the files in the Forest directory.

Files can be deleted with the rm (remove) command. Delete the aboutBison file:

```
rm <aboutBison>
```

where <aboutBison> is a relative or absolute path to the aboutBison file.

We can will also remove the directories with rm. If the directory is empty we can use the same method as we did for files (or rmdir command). If the directory has files in it, we will need to recursively remove the directory. Make sure you have the archive forestArchive.tar then try to remove the Forest directory and all files or directories it contains (Hint you will need to use the -r, -R, or --recursive option).

Verify that the Forest directory is now removed.

Now we will untar the forestArchive.tar archive using one of the following methods:

```
tar  -xf forestArchive.tar
tar --extract --file=forestArchive.tar
tar --get --file=forestArchive.tar
```

Verify that the Forest directory and all of its files/directories have been restored.

We can also compress the files we archive by adding the -z or --gzip options. Tar and archive the Forest directory again. Compare the file size (Hint try interpreting the output of ls -l).