Introduction to Linux

Introduction to Linux and the command line

Introduction to Linux

This tutorial teaches Linux basics to get new users on their feet, getting started with the terminal, the Linux command line, and executing commands. If you are new to Linux, you will want to familiarize yourself with the terminal, as it is the standard way to interact with a Linux computer. Using the command line may seem like a daunting task but it is actually very easy if you start with the basics, and build your skills from there.

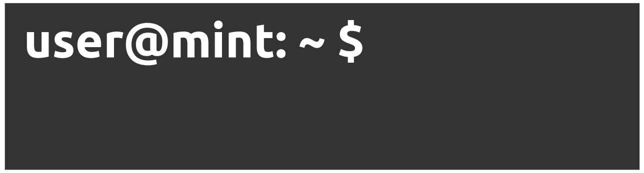
Instructions that you should follow are written in red.

Introduction to Linux: **The Shell**

In a Linux system, the *shell* is a command-line interface that interprets a user's commands and script files, and tells the computer's operating system what to do with them. This tutorial was written using the *Bourne-Again shell*, usually referred to as ***bash***, which is the default shell for most Linux distributions, including the one on the Raspberry Pi. To access the shell on the Raspberry Pi, we use what is called a Terminal.

Double-click the Terminal icon on the start menu to launch a terminal window.

You will see the default command prompt, something like:



```
user@mint: ~ $
```

Introduction to Linux: **The Prompt**

```
user@mint: ~ $
```

```
jsmith@mint:/var/log $
```

Here is a breakdown of the composition of the command prompt:

`user`: The *username* of the current user

`mint`: The *hostname* of the computer

`~`: The *current directory*. In bash, the ~, or tilde, is a special character that represents the path of the current user's *home directory*; in this case, it represents `/home/user`

`$`: The prompt symbol. This denotes the end of the command prompt, after which the user's keyboard input will appear

Above-right is an example of what the command prompt might look like, if logged in as user `jsmith` and in the `/var/log` directory:

Introduction to Linux: **Commands**

Commands can be issued at the command prompt by specifying the name of an executable file. There are many standard Linux commands and utilities that are installed with the OS, that allow you navigate the file system, install software packages, and configure the system and applications.

We will run through a few examples that will cover the basics of executing commands.

TIP: Everything on Linux is case sensitive, including filenames, directories and programs.
While it is **confusing** and **bad practice**, you *are* allowed to have both a file named File1.pdf and file1.pdf in the same directory.

Introduction to Linux: **Commands**

To execute a command without any arguments or options, simply type in the name of the command and hit Enter. If you run a command like this, it will exhibit its default behavior, which varies from command to command.

For example, if you run the `cd` (**c**hange **d**irectory) command without any arguments, you will be returned to your current user's home directory. The `ls` command will print a **l**i**s**ting of the current directory's files and directories.

Try running the `ls` command to list the files and directories

Introduction to Linux: **Arguments**

Many commands accept arguments, or parameters, which can affect the behavior of a command. For example, the most common way to use the cd command is to pass it a single argument that specifies which directory to change to. Let’s change to the /usr/bin directory, where many standard commands are installed.

Issue this command:

```
cd   /usr/bin
```

The cd component is the command, and the first argument /usr/bin follows the command. Note how your command prompt's current path has updated.

Try running the ls command to see the files that are in your new current directory. There are many files here.

Introduction to Linux: **Options**

Most commands accept *options*, also known as *flags* or *switches*, that modify the behavior of the command. As they are special arguments, options follow a command, and are indicated by a hyphen and followed by one or more *options*, which are represented by individual upper- or lower-case letters. Additionally, some options start with --, followed by a single, multi-character (usually a descriptive word) option.

For a basic example of how options work, let's look again at the `ls` command. Execute the following command:

```
ls  -l  /usr/bin
```

Here, `ls` is the name of the command, the `-l` option tells ls to give a *"long listing"* of files, providing more information than the standard `ls` command. The argument `/usr/bin` tells ls that you don't want a list of the current directory, but of some other directory, in this case `/usr/bin`.

Introduction to Linux: **Options**

Multiple flags can be used in one command, for example

```
ls -l --all ~/
```

← *(that's a 'dash-dash' before "all")*

gives a long list (`-l`) including **a**ll (`--all`) files in the directory, including hidden files in your home directory (~/)

Let's go back to our home directory by executing the `cd` command without any arguments:

```
cd
```

Check-point:

Now that you have learned about the basics of the Linux terminal (and a few commands), you should have a good foundation for expanding your knowledge of Linux commands.

Introduction to Linux: **Navigation**

When you first open the Terminal window, you were placed in your home directory, that is `/home/user`. You have full control over the contents of this directory; it's yours. We've discussed already how you move around, by using the `cd` command and how to see what files are in the directory using the `ls` command. At some point, you might get lost and forget where you are: in this case, the `pwd` command (*print working directory*) command will help you find your place and tell you where you are. Let's `cd` to our Desktop:

```
cd  desktop
```

You should get a message saying that there is no such file or directory. Recall that Linux files are case sensitive, and if you run ls, you'll see that the Desktop folder starts with a capital D. Run instead:

```
cd  Desktop
```

Introduction to Linux: **Navigation**

Let's say we now forget what directory we're currently in. Executing the `pwd` command

```
pwd
```

shows us the output `/home/user/Desktop`, reminding us where we are. The special directory **..** (period-period) always refers to the directory one level above where we currently are. Run the command

```
cd  ..
```

and then

```
pwd
```

and you'll see the output `/home/user`, meaning the `cd ..` command moved us up one level from `/home/user/Desktop` to `/home/user`.

Introduction to Linux: **Creating directories**

You will want to, at some point, create directories (folders) to organize your files. The `mkdir` command does this (**make dir**ectory). Execute the command

```
mkdir myfiles
```

If you now run `ls`, you'll see a new directory called `myfiles`. Change to this directory

```
cd myfiles
```

Introduction to Linux: **Creating files**

Now let's create a file called data1.txt. We will use a text editor called `nano`. Execute the command

```
nano  animals.txt
```

to open nano and tell it you want to open (or create if it doesn't exist) the file animals.txt. Type the following lines into the text editor:

```
monkey
donkey
turtle
iguana
```

When you're finished, press `^X` (that's the Ctrl key and the x key at the same time) to exit, then the y key to save changes, and finally the Enter key to confirm.

Now run `ls` and you'll see animals.txt listed.

Introduction to Linux: **Creating files**

Use the above steps to create a second file called fruits.txt, listing four fruits:

```
banana
apple
orange
kiwi
```

If you run `ls`, you should see both files. If you want to quickly view the contents of a file, use the more command. Run the following command:

```
more  fruits.txt
```

For long files, `more` will only show one page of data at a time.

Introduction to Linux: **Viewing files**

For short files, you can also use the `cat` command (*concatenate*). It is similar to `more`, in that it reads the file, but unlike `more`, it displays the entire file at once, even for large files.

True to its name, it can also be used to concatenate (or merge) the contents of multiple files. Run the command

```
cat  fruits.txt  animals.txt
```

This will print the contents of the fruits file and then the animals file, one after another. Try the command `paste fruits.txt animals.txt` and see how it differs from `cat`

Introduction to Linux: **Output redirection**

It is possible to write output to a file, instead of showing it on the screen. Run the command

```
cat  fruits.txt  animals.txt > all.txt
```

This is the same as before, but the `> all.txt` tells Linux to dump the output to the all.txt file. all.txt will be created if it doesn't exist yet, or overwritten if it does.

Introduction to Linux: **Copying files**

Files can be **copied** using the `cp` command. Run

```
cp  fruits.txt  food.txt
```

This will create a copy of fruits.txt and name it food.txt. `ls` will now show all files.

You can rename files using the `mv` (**move**) command. This moves the file from one location to another. Run the command

```
mv  food.txt  fruits.txt
```

If you now run `ls`, the food.txt file will be gone, as food.txt was moved ("renamed") to fruits.txt.

Introduction to Linux: **Deleting files**

! **This brings about an important point:** there is usually no warning when overwriting existing files. In the previous example the contents of food.txt overwrites the contents of fruits.txt **without warning**. Linux assumes that you, as the user, knows what you are doing. *Make sure you understand what a command will do before you execute it.*

In order to delete files, you can use the `rm` (**rem**ove) command. Let's first make a copy of animals.txt. Run

```
cp   animals.txt   animalsbackup.txt
```

and then let's remove the animals.txt file by running

```
rm   animals.txt
```

Introduction to Linux: **Moving files**

! Again, it’s important to notice that there is **no warning** before a file is removed. Likewise, there is not a “Recycle Bin” where deleted files are kept. Removing a file essentially destroys it.

Now let’s rename our backup back to the original file. Run

```
mv  animalsbackup.txt  animals.txt
```

Check-point:

You should now be able to create files, edit files, view files, move/rename copy files and navigate through Linux directories.

Introduction to Linux: **Super-user**

As a basic Linux user, you only have write access to /home/user. If you want to install programs or change system files, you must do so as the super user. To run a command as the super user, preface it with `sudo`. For example, run

```
sudo apt-get update
```

to update the lists of available software and then

```
sudo apt-get install gnuplot-x11
```

to install gnuplot (a plotting software).

Cleanup

We're now finished with these examples. Let's clean up our working directory. Run the command

```
rm  *
```

The * *expands* to represent all files in the current directory.

Now go back to your home directory by running

```
cd ..
```

Check-point:

You should now be able to create files, edit files, view files, move/rename copy files and navigate through Linux directories.

Introduction to Linux: **Gnuplot**

Gnuplot is a command-line program to plot functions and data files. It's easy to use and easy to produce nice looking plots. Let's download some data which we will plot:

```
mkdir balloon_data
cd balloon_data
wget http://uoitphysics.ca/balloon/data/pressure.txt
wget http://uoitphysics.ca/balloon/data/altitude.txt
wget http://uoitphysics.ca/balloon/data/space_data.csv
```

Gnuplot

Let's create a gnuplot script:

```
nano balloon_temperature.gp
```

The remainder of this tutorial will be completed live.