

Session #2 Part 1

The Command Line &

The Linux Filesystem Hierarchy



Summary

1. The Shell
 - 1.1. What is the shell?
 - 1.2. Ways to connect to a shell
2. The Terminal
 - 2.1. What is a terminal emulator?
 - 2.2. Ways to open a terminal
3. The Command Line
 - 3.1. The command line prompt
 - 3.2. The command line syntax
4. Filesystems
 - 4.1. What is a filesystem?
 - 4.2. Windows Directory Structure
5. Linux Filesystem Hierarchy
 - 5.1. The root '/' directory
 - 5.2. Navigating through the filesystem
 - 5.3. The '.' and '..' Links
 - 5.4. Relative and absolute paths

The Shell

What is the shell?

Remembering this diagram from last session, the shell is the first layer that a user can use to interact with the operating system.

It looks something like this:

```
Ubuntu 12.10 upubuntu tty3
upubuntu login: upubuntu
Password:
Last login: Thu Jan 17 12:11:32 PST 2013 on tty2
Welcome to Ubuntu 12.10 (GNU/Linux 3.5.0-17-generic i686)

* Documentation:  https://help.ubuntu.com/

upubuntu@upubuntu:~$ sudo fbcats > image.ppm
[sudo] password for upubuntu:
upubuntu@upubuntu:~$ _
```



This is the standard “TTY” shell that you can use, it is the default one you’ll find on Linux distributions that don’t have a GUI.

Ways to connect to a shell

You can access the TTY shell by pressing CTRL + ALT + any of the function keys ranging from 2 to 7.

Example: CTRL + ALT + F3

It doesn’t really seem like the nicest or most intuitive way to use your system, and what makes it even worse is that you have to exit the GUI every time you need to type a command.

That’s why we have the *glorious* terminal!

Note: If you’re using a **laptop** you may have to press the ‘Fn’ key alongside this combination to make it work.

Note: By default, most Linux distributions use **TTY7 (F7)** or **TTY2 (F2)**.

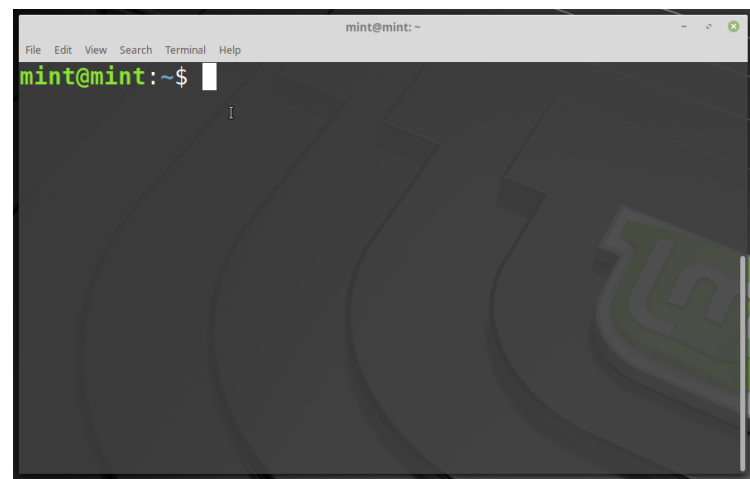
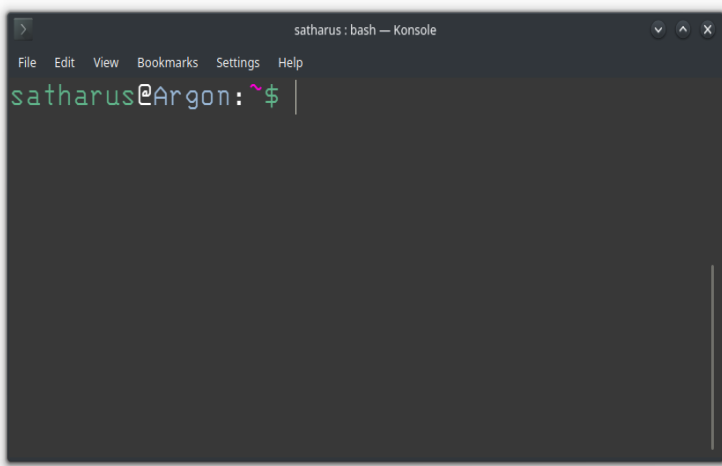
The Terminal Emulator

What is a terminal emulator?

The terminal emulator is a great way to connect to a shell without having to close your GUI. It is an amazing way to execute commands.

i.e Type commands in a window that is inside your GUI.

And it looks something like this:



Ways to open a terminal

The terminal can be opened using any of the following methods (and many more):

- CTRL + ALT + T (Works on Ubuntu and Linux Mint)
 - SUPER + T (Works on some distributions like elementary OS)
- ALT + F2 then typing in "gnome-terminal" or "konsole" (The name of the terminal emulator that your distribution uses)
- Right click on the desktop and click on "Open a terminal"
- Opening the terminal from the GUI by searching for it in the menu. Search for (Terminal, Console, Konsole, etc..)

Note: The Windows key is called "Super" or "Meta" in Linux.

The Command Line

The Command Line Prompt

When you open the terminal you'll see something like this:



It is prompting you to enter a command, let's break it down:

`osc/root`: The username of the current logged in user.

`@`: Defines that you are connected to the machine that has the name after it

`mint`: The name of the computer running (Name of the host)

`~`: The working directory, the directory that the terminal is working in right now.

Note: The `~` "Tilde" symbol means the **home directory** of the user (`/home/user`) in Linux.

`$`: States that you are logged in as a regular user.

`#`: States that you are logged in as the system administrator (root).

So we can basically summarise it to the following:

`USERNAME@HOSTNAME:WORKING_DIRECTORY($/#)`

The Command Line Syntax

When ordering the computer to do something, i.e giving it a command, you have to take care of the syntax.

Just like programming languages, the Linux shell has specific syntax that you have to use. Just so that it could be understood by the shell.

The syntax goes as follows:

[COMMAND]	[OPTION]	[ARGUMENTS]
ls	-ld, -la, -lA	file or directory
rm	-r, -f, -ir	file or directory

The Command: Intuitively, this is the command that you give to the system, i.e. delete, move, copy, list, etc..

The Option: Modifies the action of the command.

Example: List “ALL” files, delete “recursively”, show the first “40” lines of a file, delete the file “by force”

The Arguments: What you’re going to apply the command to.

i.e. Delete (command) a certain file (argument).

We can say in short that the options modify the command’s effect on the argument.

Let’s take the ls command as an example:

ls – lists the content of a directory(folder).

Running the command ls does the following:

```
osc@mint:~$ ls
Desktop  Downloads  Pictures  Templates
Documents  Music      Public    Videos
```

But that format isn’t really good if you want a detailed view, so we add the **-l** option which makes it list the content but in a “long” form:

```
osc@mint:~$ ls -l
total 0
drwxr-xr-x 2 osc osc 40 Nov  8 12:28 Desktop
drwxr-xr-x 2 osc osc 40 Nov  8 12:28 Documents
drwxr-xr-x 2 osc osc 40 Nov  8 12:28 Downloads
drwxr-xr-x 2 osc osc 40 Nov  8 12:28 Music
drwxr-xr-x 2 osc osc 40 Nov  8 12:28 Pictures
drwxr-xr-x 2 osc osc 40 Nov  8 12:28 Public
drwxr-xr-x 2 osc osc 40 Nov  8 12:28 Templates
drwxr-xr-x 2 osc osc 40 Nov  8 12:28 Videos
```

Much better! Everything is clearer and organised in a list.

How about we take a look at the hidden files too?

The **-a** option lists “all files”, this command can be shortened down to **ls -la** or **ls -al**

Note: Hidden files and directories in Linux start their name with a dot “.”.

```
osc@mint:~$ ls -l -a
total 44
drwxr-xr-x 16 osc osc 520 Nov  8 12:31 .
drwxr-xr-x  1 root root  80 Nov  8 12:27 ..
-rw-r--r--  1 osc osc 310 Nov  8 12:28 .ICEauthority
-rw-r--r--  1 osc osc  49 Nov  8 12:28 .Xauthority
-rw-r--r--  1 osc osc 220 Nov  8 12:27 .bash_logout
-rw-r--r--  1 osc osc 3771 Nov  8 12:27 .bashrc
drwxr-xr-x  6 osc osc 120 Nov  8 12:28 .cache
drwxrwxr-x  3 osc osc  60 Nov  8 12:28 .cinnamon
drwxr-xr-x 12 osc osc 280 Nov  8 12:28 .config
-rw-r--r--  1 osc osc  27 Nov  8 12:28 .dmrc
drwxr-xr-x  2 osc osc  40 Nov  8 12:28 .gconf
drwxr-xr-x  3 osc osc  60 Nov  8 12:28 .gnupg
-rw-r--r--  1 osc osc  22 Nov  8 12:27 .gtkrc-2.0
-rw-r--r--  1 osc osc 516 Nov  8 12:27 .gtkrc-xfce
drwxr-xr-x  3 osc osc  60 Nov  8 12:28 .local
-rw-r--r--  1 osc osc 807 Nov  8 12:27 .profile
-rw-r--r--  1 osc osc   0 Nov  8 12:31 .sudo_as_admin_successful
-rw-r--r--  1 osc osc 9209 Nov  8 12:29 .xsession-errors
drwxr-xr-x  2 osc osc  40 Nov  8 12:28 Desktop
drwxr-xr-x  2 osc osc  40 Nov  8 12:28 Documents
drwxr-xr-x  2 osc osc  40 Nov  8 12:28 Downloads
drwxr-xr-x  2 osc osc  40 Nov  8 12:28 Music
drwxr-xr-x  2 osc osc  40 Nov  8 12:28 Pictures
drwxr-xr-x  2 osc osc  40 Nov  8 12:28 Public
drwxr-xr-x  2 osc osc  40 Nov  8 12:28 Templates
drwxr-xr-x  2 osc osc  40 Nov  8 12:28 Videos
```

Filesystems

What is a filesystem?

A filesystem is the way that the files are stored on a storage device (i.e. Hard Drive, USB Flash Drive, etc..).

Each operating system uses a certain filesystem:



Linux



EXT4, XFS



Windows

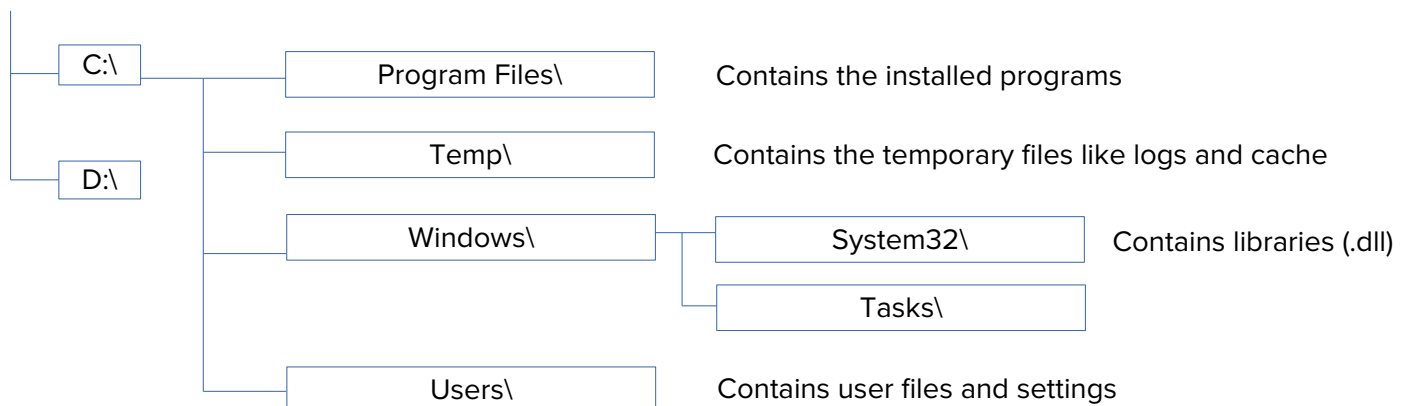


NTFS, FAT32

Note: Linux supports NTFS and FAT32, but Windows doesn't support EXT4 or XFS, that's why you can't see the Linux partitions on Windows.

Windows Directory Structure

A directory structure is the way an operating system's files are arranged displayed to the user.



Windows, like every operating system, has a specific directory structure for its NTFS file system.

Each disk is assigned a letter, and you browse your files based on that.

Note: C:\ and D:\ could be 2 separate physical hard drives.

Linux also has a directory structure, called

“Filesystem Hierarchy Standard” or “The Linux Filesystem Hierarchy”.

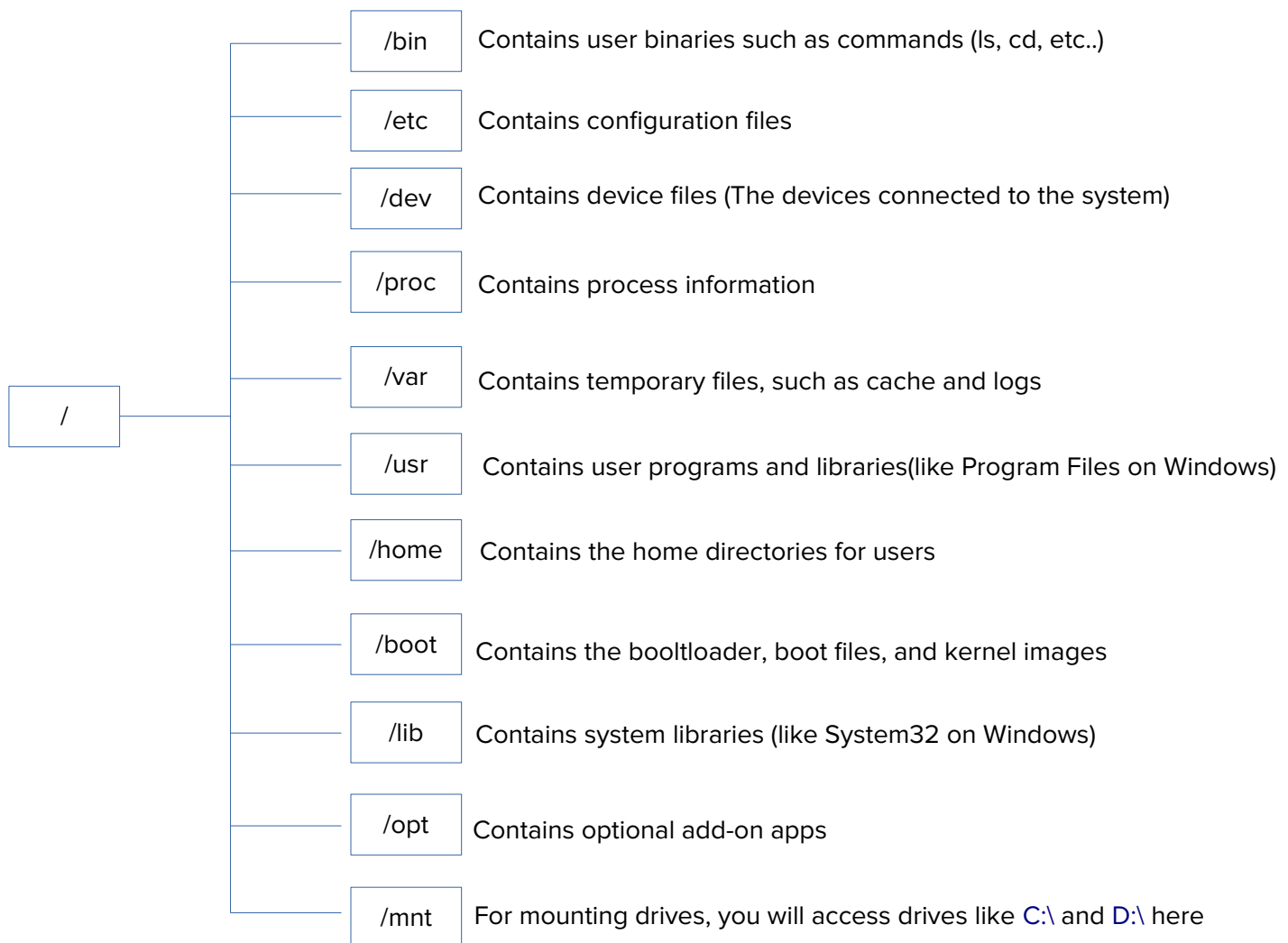
Linux Filesystem Hierarchy

The root '/' directory

The '/' directory or the "root" directory is where everything begins on Linux.

No matter what you want to access, where it is, it will somehow connect to the root directory.

Here's a demonstration of the Linux Filesystem Hierarchy:



From the previous, we can see that:

"Everything in Linux is a file". Even devices and processes, everything is a file under the '/' directory somehow.

Note: C:\ and D:\ here are not accessed as C:\ or D:\, but instead as directories under '/'.

Let's test it out!

```
osc@mint:~$ ls /  
bin    cdrom  etc    lib    media  opt    rofs   run    srv    tmp    var  
boot   dev    home   lib64  mnt    proc   root   sbin   sys    usr
```

If we list the content of the root directory using the `ls` command, we will find the directories in the previous diagram.

Now we know what a filesystem, directory, and file are. Let's talk about how we can access them.

Navigating through the filesystem

You opened a terminal, now what?

The first thing you want to do is to know where the terminal is working:

```
osc@mint:~$ pwd  
/home/osc
```

`pwd`: Print Working Directory, tells you the directory your terminal is working in.

Now that you know where you are in the system, you should see the content of the directory using the `ls` command.

```
osc@mint:~$ ls  
Desktop    Downloads  Pictures  Templates  
Documents  Music      Public    Videos
```

What if we wanted to enter the Pictures directory?

```
osc@mint:~$ cd Pictures  
osc@mint:~/Pictures$ pwd  
/home/osc/Pictures  
osc@mint:~/Pictures$
```

`cd`: Change Directory, changes the working directory to the specified argument.

Now the working directory is Pictures, notice how the text before \$ also changed to ~/Pictures which is the same as /home/osc/Pictures, which is the working directory.

What if I wanted to go back to the home directory?

There are 4 ways:

`cd ~`: This basically means `cd /home/osc` since `~` means the home directory of the current user.

`cd /home/osc`: Tells the shell to change the working directory to `/home/osc`.

`cd`: Running the `cd` command without an argument takes you to the home directory by default.

`cd ..`: `..` refers to the parent directory, continue reading:

The `'.'` and `'..'` Links

Each directory has 2 hidden files (links) in it, `'.'` and `'..'`.

The `'.'` link refers to the directory itself.

The `'..'` link refers to the directory before it (parent directory).

Example: If the working directory is `/home/osc/Pictures/`, then:

`'.'` = `/home/osc/Pictures/` and `'..'` = `/home/osc/` which is the directory before it.

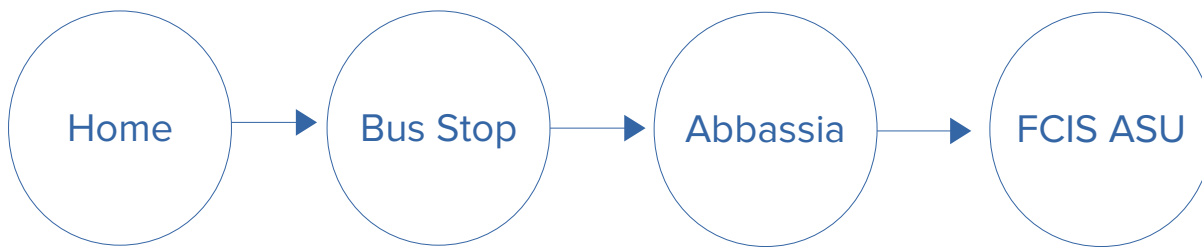
To verify:

```
osc@mint:~/Pictures$ ls -la
total 0
drwxr-xr-x  2 osc osc  60 Nov  8 16:04 .
drwxr-xr-x 16 osc osc 520 Nov  8 12:31 ..
-rw-r--r--  1 osc osc   0 Nov  8 16:04 Photo.jpg
osc@mint:~/Pictures$ ls .
Photo.jpg
osc@mint:~/Pictures$ cd ..
osc@mint:~$ pwd
/home/osc
```

This can be a little confusing at first, so practice with yourself and maybe try drawing it on a piece of paper to visualise how things really work.

Relative and absolute paths

Let's simplify this by taking a guy called "Jack" as an example, Jack goes to FCIS ASU every day, this is the path he takes daily:



Jack's route to college daily is Home->Bus Stop->Abbassia->FCIS ASU.

If he met someone at Abbassia and asked him: "Where are you going?" Jack's response will be "FCIS ASU" only, because that's the next step.

If someone asked Jack "What's your full route to college?" Jack's response would be " Home->Bus Stop->Abbassia->FCIS ASU".

Note that his route from Abbassia is shorter because it is relative to Abbassia.

The same thing applies in Linux for directories and files.

Absolute Path: The total path leading to the directory.

Relative Path: The path relative to the working directory.

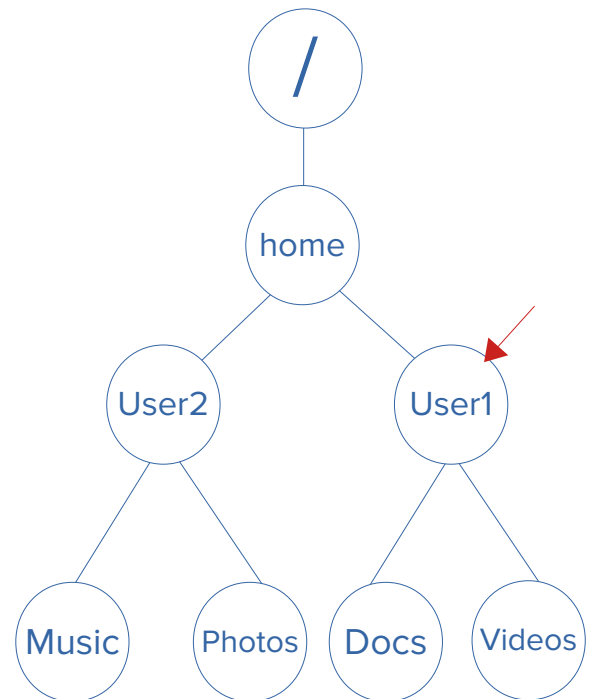
Example:

In the diagram,

Let the working directory be /home/User1

The relative path for “Videos” would be: Videos

The absolute path would be: /home/User1/Videos



Test yourself:

In the same diagram, let the working directory be /home/User1 and the user you're logged in as called User1.

- Which directory does '.' refer to?
- Which directory does '..' refer to?
- What would be the working directory if you run `cd ..`?
- What would be the working directory if you run `cd .`?
- What would be the working directory if you run `cd Videos`?
- What would happen if you run `cd ../User2`?
- What would happen if you run `cd`?
- What would happen if you run `cd User2`?
- What would happen if you run `cd /home/User2`?



Solution:

- The directory itself (User1).
- The parent directory (the directory before it: home).
- The working directory would be home.
- The shell will change the directory to the current working directory so nothing will change.
- The shell will change the working directory to /home/User1/Videos.
- The shell will change the working directory to /home/User2
(This is the relative path)
- The shell will change the working directory to /home/User1 as you are logged in as User1, so nothing will change because the working directory is already /home/User1
- Error, there isn't a directory called "User2" under the directory "User1".
- The shell will change the working directory to /home/User2
(This is the absolute path)