Linux Command Line

Basics for using Linux from Command Line

Linux Partitions and Filesystem

A **partition** is a logical part of the disk, whereas a **filesystem** is a method of storing/finding files on a hard disk (usually in a partition).

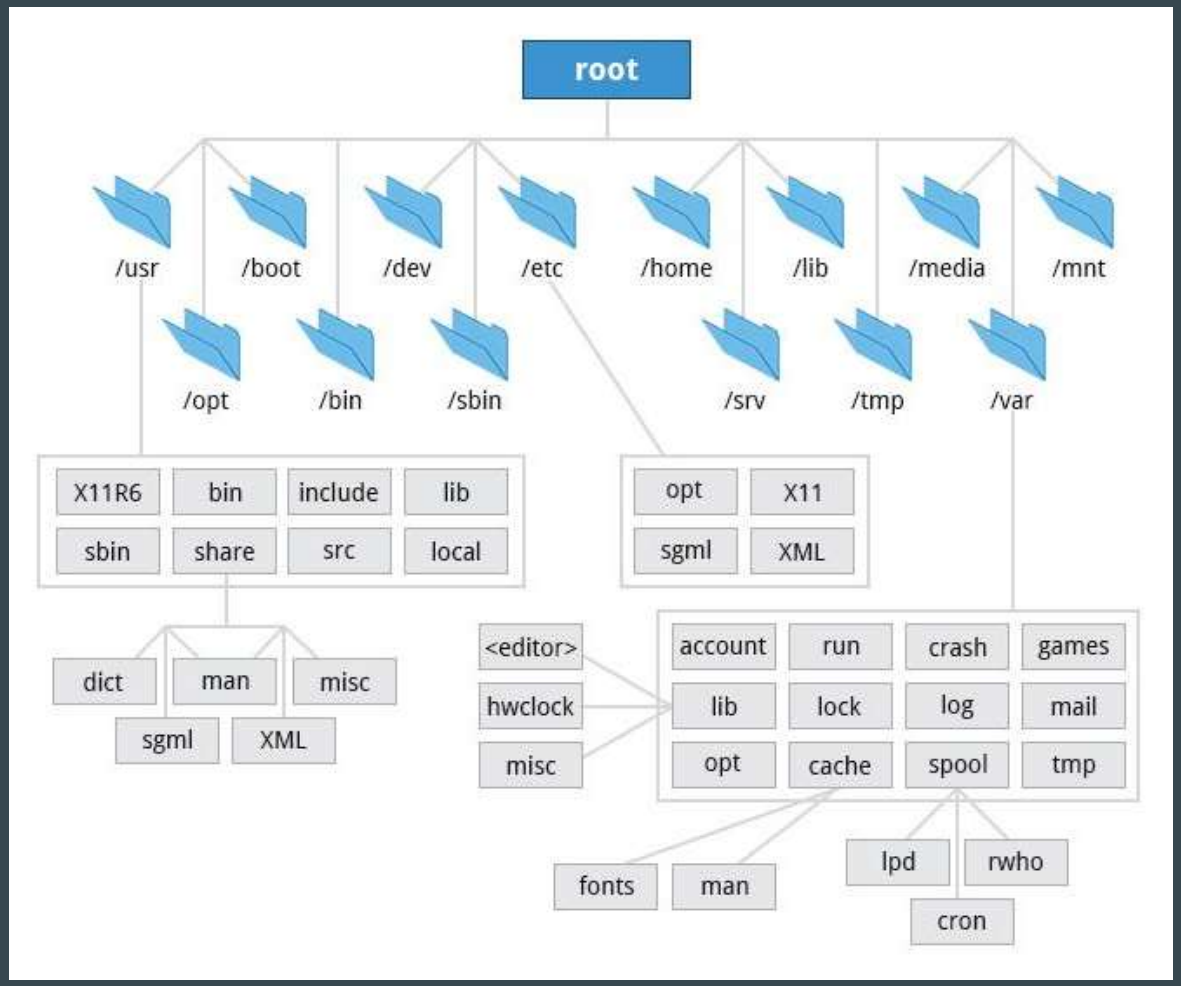
Linux systems store their important files according to a standard layout called the Filesystem Hierarchy Standard, or FHS.

Some examples of filesystem types that Linux supports are:

- **`ext3`**, **`ext4`**, **`btrfs`**, **`xfs`** (native Linux filesystems)
- **`vfat`**, **`ntfs`**, **`hfs`** (filesystems from other operating systems)

On a UNIX system, everything is a file; if something is not a file, it is a process

File System


root
/usr
/boot
/dev
/etc
/home
/lib
/media
/mnt
/opt
/bin
/sbin
/srv
/tmp
/var
X11R6
bin
include
lib
sbin
share
src
local
opt
X11
sgml
XML
dict
man
misc
sgml
XML
<editor>
hwclock
misc
account
run
crash
games
lib
lock
log
mail
opt
cache
spool
tmp
fonts
man
lpd
rwho
cron

FileSystem

Home - Each user has a home directory, usually placed under `/home`. The `/root` directory is the root user's home directory.

`/bin`	Contains executable binaries, essential commands used in single-user mode, and essential commands required by all system users, such as: `ps`, `ls` and `cp`
`/usr/bin`	Contains commands that are not essential for the system in single-user mode
`/sbin`	Essential binaries related to system administration, such as `ifconfig` and `shutdown`
`/usr/sbin`	Less essential system administration programs

In some of the most modern Linux systems `/usr/bin` and `/bin` are actually just linked together as are `/usr/sbin` and `/sbin`

File System

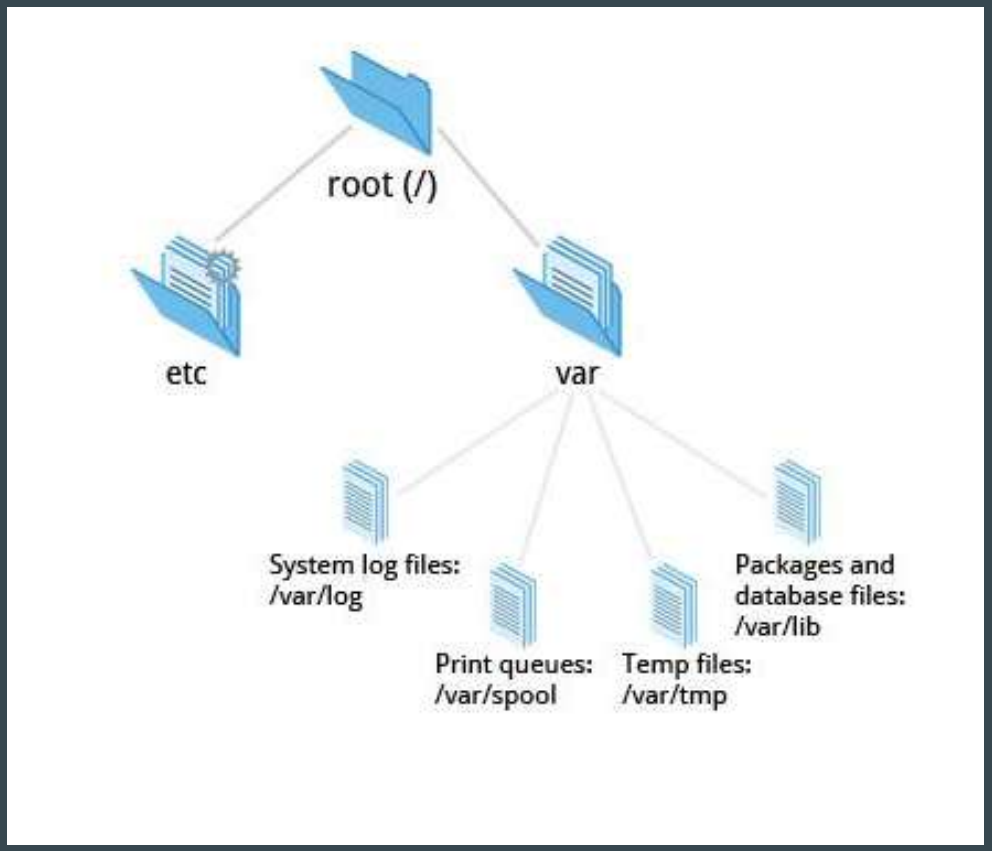
`/var`	Contains files that are expected to change in size and content as the system is running (`var` stands for variable)
`/etc`	Home for system configuration files. It contains no binary programs, although there are some executable scripts.

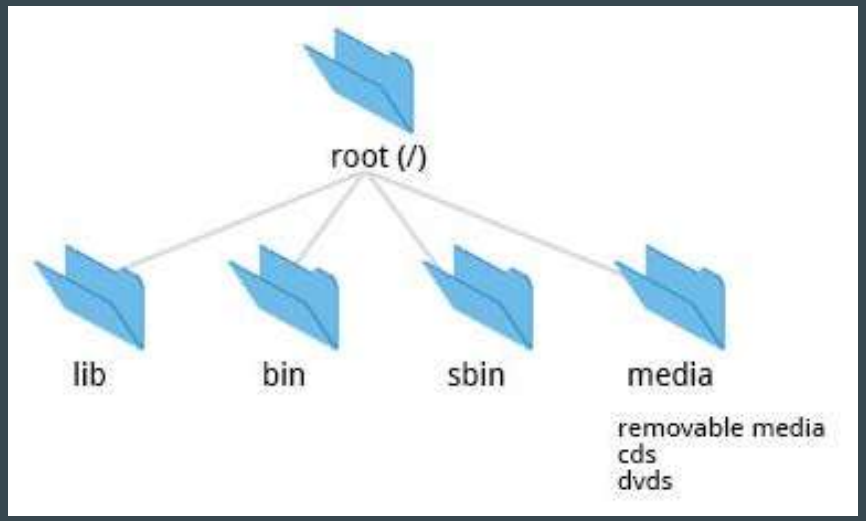
The `/var` directory may be put in its own filesystem so that growth of the files can be accommodated and the file sizes do not fatally affect the system. Network services directories such as `/var/ftp` (the FTP service) and `/var/www` (the HTTP web service) are also found under `/var`.

Files like `passwd`,`shadow` and `group` for managing user accounts are found in the `/etc` directory. System run level scripts are found in subdirectories of `/etc`.

File System

`/lib`	Contains libraries (common code shared by applications and needed for them to run) for the essential programs in `/bin` and `/sbin`. Kernel modules (kernel code, often device drivers, that can be loaded and unloaded without re-starting the system) are located in `/lib/modules/<kernel-version-number>`
`/media`	The `/media` directory is typically located where removable media, such as CDs, DVDs and USB drives are mounted. Unless configuration prohibits it, Linux automatically mounts the removable media in the `/media` directory when they are detected.


root (/)
etc
var
System log files:
/var/log
Print queues:
/var/spool
Temp files:
/var/tmp
Packages and
database files:
/var/lib


root (/)
lib
bin
sbin
media
removable media
cds
dvds

FileSystem

`/opt`	Optional application software packages
`/sys`	Virtual pseudo-filesystem giving information about the system and the hardware. Can be used to alter system parameters and for debugging purposes.
`/srv`	Site-specific data served up by the system. Seldom used.
`/tmp`	Temporary files; on some distributions erased across a reboot and/or may actually be a ramdisk in memory
`/usr`	Multi-user applications, utilities and data

FileSystem - SubDirectories under `/usr`

`/usr/include`	Header files used to compile applications
`/usr/lib`	Libraries for programs in /usr/bin and /usr/sbin
`/usr/lib64`	64-bit libraries for 64-bit programs in /usr/bin and /usr/sbin
`/usr/sbin`	Non-essential system binaries, such as system daemons
`/usr/share`	Shared data used by applications, generally architecture-independent
`/usr/src`	Source code, usually for the Linux kernel
`/usr/X11R6`	X Window configuration files; generally obsolete
`/usr/local`	Data and programs specific to the local machine. Subdirectories include bin, sbin, lib, share, include, etc.
`/usr/bin`	This is the primary directory of executable commands on the system

Linux Shell

The Shell is a program that accepts commands (user input) and provides them to OS

Types of files

Directories	files that are lists of other files
Special files	the mechanism used for input and output. Most special files are in /dev
Links	make a file or directory visible in multiple parts of the system
Sockets	inter-process networking protected by the file system's access control
Named pipes	form a way for processes to communicate with each other

Symbol	Meaning
`-`	Regular file
`d`	Directory
`l`	Link
`c`	Special file
`s`	Socket
`p`	Named pipe
`b`	Block device

Key Combinations in Bash

`Ctrl+A`	Move cursor to the beginning of the command line
`Ctrl+E`	Move cursor to the end of the command line
`Ctrl+C`	End a running program and return the prompt
`Ctrl+D`	Log out of the current shell session
`Ctrl+L`	Clear terminal
`Ctrl+R`	Search command history
`ArrowUp` `ArrowDown`	Browse History
`Tab`	Command or Filename Completion
`Tab Tab`	Shows Command or Filename completion possibilities

Basic Navigation

Command	Result
`cd [location]`	Move around with Change Directory to specified [location]
`pwd`	Displays the present working directory
`cd ~` or `cd`	Change to your home directory (short-cut name is ~ (tilde))
`cd ..`	Change to parent directory
`cd -`	Change to previous directory (- (minus))

`ls` - Contents of the current directory (or any other location if specified)

```
ls [options] [location]
```

Absolute & Relative Paths

Absolute Path: Specify a location (of file or directory) in relation to root directory

An absolute path begins with the root directory and follows the tree, branch by branch, until it reaches the desired directory or file. Absolute paths always start with `/`

Relative Path: Specify a location (of file or directory) in relation to Current Location

A relative pathname starts from the present working directory. Relative paths never start with `.`

Most of the time it is convenient to use relative paths, which require less typing.

Take advantage of the shortcuts provided by: `.` (present directory), `..` (parent directory) and `~` (your home directory).

Files

Everything is a file!!!

Linux is Case Sensitive

To accommodate spaces in names, use

Quotes - `cd 'My Pictures'`

or Escape Characters - `cd My\ Pictures`

Hidden Files and Directories start with a period .

`ls -a` Lists the contents of directory, including hidden files

Files...

Create directory

```
mkdir [Options] <directory>
```

Delete Directory

```
rmdir [Options] <directory>
```

A Directory must be empty before it can be deleted

Create Blank File

```
touch [Options] <filename>
```

Files...

Move File / Directory

```
mv [options] <source> <destination>
```

Delete File / non-empty directory

```
rm [options] <file>
```

Delete non-empty directories, use recursive option '-r'

```
rm -r <directory>
```

To create an empty file or change the access timestamp on existing file

```
touch <filename>
```

Viewing Files

Command	Usage
`cat`	Used for viewing files that are not very long; it does not provide any scroll-back
`tac`	Used to look at a file backwards, starting with the last line.
`less`	Used to view larger files because it is a paging program; it pauses at each screenful of text, provides scroll-back capabilities, and lets you search and navigate within the file
`tail`	Used to print the last 10 lines of a file by default. You can change the number of lines by doing -n 15 or just -15 if you wanted to look at the last 15 lines instead of the default
`head`	The opposite of tail; by default it prints the first 10 lines of a file.

less shortcuts

Command	Action
Page Up or b	Scroll back one page
Page Down or space	Scroll forward one page
G	Go to the end of the text file
1G	Go to the beginning of the text file
/characters	Search forward in the text file for an occurrence of the specified characters
n	Repeat the previous search
h	Display a complete list less commands and options
q	Quit

File Permissions

Three distinct actions that can be performed on files

- `r` - read, `w` - write & `x` - execute

3 Sets of Access

- **Owner** - person who owns the file
- **Group** - every file belongs to a group
- **Others** - anyone other than Owner and Group

Use long listing option to view permissions `ls -l`

File Permissions

Change permissions

```
chmod [permissions] [path]
```

Examples

```
chmod g+x run.sh    chmod u-w run.sh    chmod o+wx run.sh
```

[permissions] is made up of 3 components

- Changing permission for: user/owner [`u`/`o`], group [`g`], others [`o`], all [`a`]
- Granting [`+`] or Revoking [`-`] permission
- Which permission: read [`r`], write [`w`], execute [`x`]

File Permissions

Shorthand

#	R	W	X
0	0	0	0
1	0	0	1
2	0	1	0
3	0	1	1
4	1	0	0
5	1	0	1
6	1	1	0
7	1	1	1

Hard & Soft (Symbolic) Links

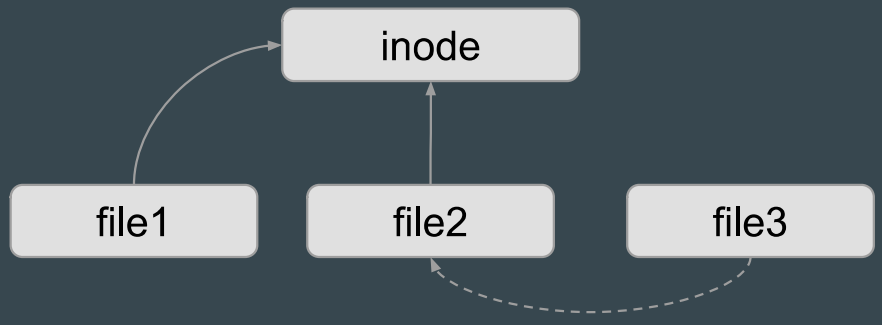
ln can be used to create hard links and (with the -s option) soft links, also known as symbolic links or symlinks.

A hard link for file1, called file2, is created with the command:

```
$ ln file1 file2
```

Symbolic (or Soft) links are created with the -s option

```
$ ln -s file1 file3
```



Standard File Streams

By default there are three standard file streams (or descriptors) always open for use:

- Standard input (`stdin`) default is keyboard
- Standard output (`stdout`) printed on terminal
- Standard error (`stderr`) printed on terminal

All open files are represented internally by file descriptors. These are represented by numbers starting at zero. `stdin` is file descriptor `0`, `stdout` is file descriptor `1`, and `stderr` is file descriptor `2`. If other files are opened in addition to these three, they will start at file descriptor 3 and increase from there.

I/O Redirection

To redirect standard output to a file (contents overridden each time)

```
$ ls > my_output.txt
```

To append to a file

```
$ ls >> my_output.txt
```

Standard Input can be redirected too

```
$ sort < my_data.csv
```

I/O Redirection

Redirect `stderr` to separate file

```
$ command_name 2> error-filename
```

Redirect anything written to `stderr` in the same place as `stdout`

```
$ command_name > out_all_filename 2>$1
```

Pipelines - The output of one command can be fed as input to another command

```
$ ls -l | less
```

Wildcards

Wildcard	Result
`?`	Matches any single character
`*`	Matches any string of characters
`[set]`	Matches any character in the set of characters, for example [adf] will match any occurrence of "a", "d", or "f"
`[!set]`	Matches any character not in the set of characters

Finding Files

```
find <location> <comparison-criteria> <search-term>
```

Recurses down the filesystem tree from any particular directory (or set of directories) and locates files that match specified conditions. The default pathname is always the present working directory.

To search for a file named my_notes.txt under my_docs directory

```
$ find ./my_docs -name "my_notes.txt"

./my_docs/my_notes.txt
```

Comparing Files

`diff` is used to compare files and directories.

```
diff <filename1> <filename2>
```

Option	Usage
`-c`	Provides a listing of differences that include 3 lines of context before and after the lines differing in content
`-r`	Used to recursively compare subdirectories as well as the current directory
`-i`	Ignore the case of letters
`-w`	Ignore differences in spaces and tabs (whitespace)

File Utility

In Linux, a file's extension often does not categorize it the way it might in other operating systems. One can not assume that a file named file.txt is a text file and not an executable program.

The real nature of a file can be ascertained by using the `file` utility. For the file names given as arguments, it examines the contents and certain characteristics to determine whether the files are plain text, shared libraries, executable programs, scripts, or something else.

Filters

`sort`	Sorts standard input then outputs the sorted result on standard output
`grep`	Examines each line of data it receives from standard input and outputs every line that contains a specified pattern of characters.
`tail`	Outputs the last few lines of its input. Useful for things like getting the most recent entries from a log file.
`sed`	Stream Editor
`awk`	An entire programming language designed for constructing filters.