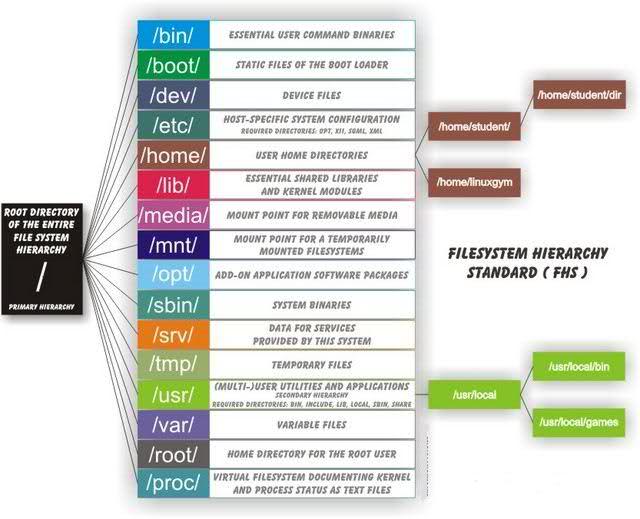
Linux - File Structure & Files



# General Structure

### /bin

All executable binary programs required during booting, repairing, running into single user mode, and other important commands (cat, ls, etc)

### /boot

Holds important files during boot-up process including the linux kernel

### /dev

Contains device files for hardward devices. For example, cpu and cdrom. When accessing a item in dev you are interacting with the indiviual bits and bytes on the hardware, to access actual files use the /media directory.

* /dev/null - pseudo device which does not exist, direct output to here to ignore and delete it

### /etc

Contains application configuration files and the startup, shutdown, start, and stop script for every induvidual program

### /home

Home directory of users. Each user has subdirectory in /home named after their username.

### /lib

Contains kernel modules and shred library images required to boot the system and run commands in the root file system

### /lost+found

Install during creation of linux and is where files which are broken during unexpected shut-downs are stored. Can be useful for recovery.

### /media

Temporary mounting for external hardware devices such as usb drives and external hard drives

### /mnt

Temporary mounting for a mounting-file-system

### /opt

Sometimes called optional, contains third party application software such as Java

### /proc

Pseudo file-system which contains information about running processes with process-id (pid)

### /root

Home directory of the root user

### /run

Only clean solution for the early-runtime-dir problem

### /sbin

Contains binary executable programs required by the sysadmin such as iptables, ifconfig, etc

### /srv

Called 'Service', contains server and service related files

### /sys

Virtual filesystem which stores information on and allows modification to devices connected to the system

### /tmp

Temporary storage available to all which is cleared on next boot

### /usr

Contains binaries, docs, source code, and libraries for second level programs (non-system programs)

### /var

Contains variable files of which the contents are expected to grow and change. For example, log, lock, spool, mail, and temp files.

# Finding Files

On most GUI operating systems there is file search functionality in the file manager. While file managers can also do this on linux, there are a few built in commands which can be used to search for files.

## locate

The locate command searches through the system indexes:

locate <filename>

This is fast, however since the indices are generated by the updatedb cron task, can sometimes be out of date. Update the indices using:

sudo updatedb

## find

find searches down through a directory structure, looking for files which match the set crieteria. For example:

find /var -name access.log

There are many other options which can be used, such as -mtime <days>, which will match files only modifed in the last <days>

## grep

grep -R can be used to recursivly search the content of plain text files in a file structure. For example:

grep -R -I "PermitRootLogin" /etc/\*

## which

the which command can be used to see where in the $PATH a command comes from, for example to find the binary for a command:

which <command>

# Linux Virtual Filesystem

The linux virtual filesystem (VFS) is the layer above the actual filesystem, e.g ext4, so it is placed between the filename and the actual data on the disk.

Each file has a numerical value which related the filename and filesystem, called the inode. It can be seen with the -I option on the ls command, or the stat command.

## Hard links

Since the inode numbers can be referenced by more than filename, hardlinks can be created meaning several filenames can point to the same data on the disk. It is important to note that permissions are kept on the inode level of the file not the filename, so any hardlinks will have the same permissions.

Hard links are created using the link command, for example:

ln /etc/passwd <hardlink-filename>

## System links

System links simply link to another filename and have completely open file permissions, since the permissions of the linked file is what matters. They can be used to control file access futher.

System links are created using the link command with the -s option:

ln -s /etc/passwd <syslink-filename>

## Differences

Hard links:

* Only link to a file, not a directory
* Can't reference a file on a different disk/volume
* Links will reference a file even if it is moved
* Links reference inode/physical locations on the disk

Symbolic (soft) links:

* Can link to directories
* Can reference a file/folder on a different hard disk/volume
* Links remain if the original file is deleted
* Links will NOT reference the file anymore if it is moved
* Links reference abstract filenames/directories and NOT physical locations.
* They have their own inode

## Filesystem Anatomy

<https://developer.ibm.com/tutorials/l-linux-filesystem/>

# File Managers - Local

There are many different linux file managers, which make navigating the file system quicker and easier:

## Midnight Commander

Package and command name: mc

Lightweight and quick, easy gui

## Ranger

Package and command name: ranger

Quick, simple, uses vim like commands.

# File Managers - Remote

There are many different protocols made for sharing files between remote and local machines:

* SMB: Microsoft's file sharing, useful on a local network of Windows machines
* AFP: Apple’s file sharing, useful on a local network of Apple machines
* WebDAV: Sharing over web (http) protocols
* FTP: Traditional Internet sharing protocol
* scp: Simple support for copying files
* rsync: Fast, very efficient file copying
* SFTP: file access and copying over the SSH protocol (Despite the name, the SFTP protocol at a technical level is completely unrelated to traditional FTP)

SFTP is the most common, since it is installed on most machine and simply uses SSH to connect, meaning the ssh credentials can be used for easy configuration.

Popular SFTP GUI applications are:

* Cyberduck
* Filezilla

# File Viewers

There are many different linux file viewers and editors, each with different advantages and learning curves:

## Vim

Decendant of vi, powerful, popular file editor.

Pros:

* Lightweight
* Installed on most systems
* Quick and useful syntax for long editing sessions
* Very customisable
* Strong user base
* Mouseless

Cons:

* Syntax has steep learning curve

## Nano

Quick lightweight file editor

Pros:

* Lightweight
* Installed on some systems
* Very easy to use

Cons:

* Can be slow to edit files
* Not very powerful

## Less

Quick file viewer and editorq, based on more and vi but with many more commands

Pros:

* Installed on most machines
* Uses vi like commands
* Lightweight and very fast

Internal commands (may be preceded by decimal number N to perform custom action):

* SPACE - scrolls page or N lines
* z - scrolls N lines and sets page size
* d - scrolls N lines and default to half screen
* b - scrolls back N lines and defaults to one window
* r - repaint screen (R to discard buffered information)
* F - scroll forward and keep trying to read when end of file is reached, this similar to what tail -f does to monitor file in realtime
* g - go to beginning of file
* G - go to end of file
* /pattern - regexp search forward
* ?pattern - regexp search backwards
* n - next pattern match