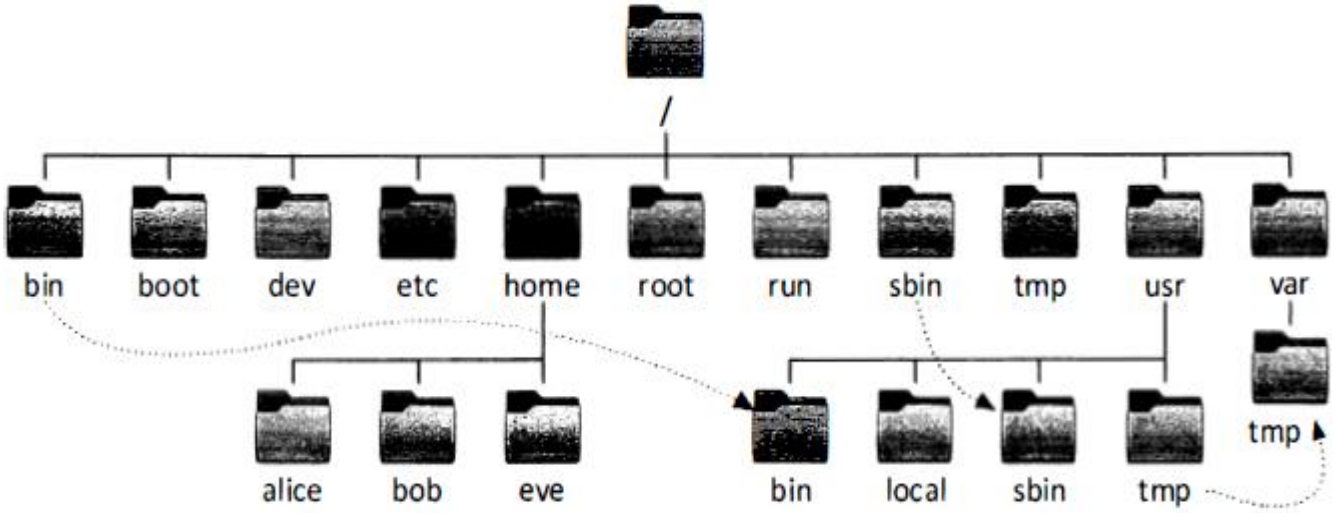
File System Hierarchy

- All files on a Linux system are stored on file systems which are organized into a single inverted tree of directories, known as a *file sys tem hierarchy.*



- The directory / is the root directory at the top of the file system hierarchy.
- Subdirectories of / are used for standardized purposes to organize files by type and purpose. This makes it easier to find files.
- The following terms are encountered in describing file system directory contents:
  - static is content that remains unchanged until explicitly edited or reconfigured.
  - dynamic or variable is content typically modified or appended by active processes.
  - persistent is content, particularly configuration settings, that remain after a reboot.
  - runtime is process- or system-specific content or attributes cleared during reboot.

Why File System Hierarchy is important..?

- Basic Security
- Categorizing Sharable and UN-sharable files in different buckets
- Organizing application files as per the FHS complaint
- Which gives more flexibility for the developers to write or develop programs
- Easy to manage Operating system for Administrators

Location	Purpose
/usr	Installed software, shared libraries, include files, and static read-only program data. Important subdirectories include: - **/usr/bin**: *User commands.* - **/usr/sbin**: *System administration commands.* - **/usr/local**: *Locally customized software.*
/etc	Configuration files specific to this system.
/var	Variable data specific to this system that should persist between boots. Files that dynamically change (e.g. databases, cache directories, log files, printer-spooled documents, and website content) may be found under **/var**.
/run	Runtime data for processes started since the last boot. This includes process ID files and lock files, among other things. The contents of this directory are recreated on reboot. (This directory consolidates **/var/run** and **/var/lock** from older versions of Red Hat Enterprise Linux.)
/home	*Home directories* where regular users store their personal data and configuration files.

/root	Home directory for the administrative superuser, root.
/tmp	A world-writable space for temporary files. Files which are more than 10 days old are deleted from this directory automatically. Another temporary directory exists, **/var/tmp**, in which files that have not been accessed, changed, or modified in more than 30 days are deleted automatically.
/boot	Files needed in order to start the boot process.
/dev	Contains special *device files* which are used by the system to access hardware.

This directory contains static, persistent system configuration data.	/etc
This is the system's root directory.	/
User home directories are located under this directory.	/home
This is the root account's home directory.	/root
This directory contains dynamic configuration data, such as FTP and websites.	/var
Regular user commands and utilities are located here.	/usr/bin
System administration binaries, for root use, are here.	/usr/sbin
Temporary files are stored here.	/tmp
Contains dynamic, non-persistent application runtime data.	/run
Contains installed software programs and libraries.	/usr

Absolute Vs Relative Path

- An absolute path is a fully qualified name, beg inning at the root (/) directory and specifying each subdirectory traversed to reach and uniquely represent a single file.
- Everyfile in a file system has a unique absolute path name.
- Like an absolute path , a relative path identifies a unique file , specifying only the path necessary to reach the file from the working directory.

- For standard Linux file systems , the path name of a file, including all / characters, may be no more than 4095 bytes long.
- Each component of the path name separated by / characters may be no more than 255 bytes long .
- File names can use any UTF-8 encoded Unicode character except / and the NUL character.
- Linux file system s- including ext4, XFS , BTRFS,GFS2, are case sensitive.

Quiz

Action to accomplish	Command
List the current user's home directory (long format) in simplest syntax, when it is not the current location.	
Return to the current user's home directory.	
Determine the absolute path name of the current location.	
Return to the most previous working directory.	
Move up two levels from the current location.	
List the current location (long format) with hidden files.	
Move to the binaries location, from any current location.	
Move up to the parent of the current location.	
Move to the binaries location, from the root directory.	

cd	cd -	cd ..	cd ../..	cd /bin	cd bin	ls -al
ls -l ~	pwd					

Navigating Paths

- pwd
- ls (a,l,R)
- cd
- ~
- touch
- . (current directory)
- .. (parent directory)
- cd – (alternate between two directories)

List the current user's home directory (long format) in simplest syntax, when it is not the current location.	ls -l ~
Return to the current user's home directory.	cd
Determine the absolute path name of the current location.	pwd
Return to the most previous working directory.	cd -
Move up two levels from the current location.	cd ../..
List the current location (long format) with hidden files.	ls -al
Move to the binaries location, from any current location.	cd /bin
Move up to the parent of the current location.	cd ..
Move to the binaries location, from the root directory.	cd bin

Command line file management

- mkdir dir
- mkdir -p par1/par2/dir
- cp file1 file2
- cp file1 file2 file3 dir
- mv f1 f2 (renaming in the same dir)
- mv f1 /dir/f2
- rm (option irf)
- rmdir

Task

- Create a directory.
- Create multiple directories in a single command.
- Create a directory when parent dir is missing.
- Copy one file to another in the current dir.
- Copy multiple file to another dir .
- Copy multiple file and dir to another dir.
- Rename one file in the same dir.
- Move one file to another dir.
- Remove an empty dir.
- Remove a non empty dir.
- Remove a non empty dir forcefully.

File globbing: path name expansion

- The Bash shell has a path name-matching capability historically called globbing, abbreviated from the "global command " file path expansion program of early UNIX .
- The Bash globbing feature,commonlycalled pattern matching or " wild cards " , makes managing largenumbers of files easier.
- Using meta-characters that "expand " to match file and path names.

Pattern	Matches
*	Any string of 0 or more characters.
?	Any single character.
~	The current user's home directory.
~*username*	User *username*'s home directory.
~+	The current working directory.
~-	The previous working directory.
[*abc...*]	Any one character in the enclosed class.
[!*abc...*]	Any one character *not* in the enclosed class.
[^*abc...*]	Any one character *not* in the enclosed class.
[[:alpha:]]	Any alphabetic character.(1)
[[:lower:]]	Any lower-case character.(1)
[[:upper:]]	Any upper-case character.(1)
[[:alnum:]]	Any alphabetic character or digit.(1)
[[:punct:]]	Any printable character not a space or alphanumeric.(1)
[[:digit:]]	Any digit, **0-9**.(1)
[[:space:]]	Any one whitespace character; may include tabs, newline, or carriage returns, and form feeds as well as space.(1)
Note	(1)pre-set POSIX character class; adjusts for current locale.

- Tilde expansion: The tilde character (~), when followed by a slash delimiter, matches the current user's home directory. When followed by a string of characters up to a slash, it will be interpreted as a username, if one matches.
- Brace expansion:

```
[student@desktopX glob]$ echo {Sunday,Monday,Tuesday,Wednesday}.log
Sunday.log Monday.log Tuesday.log Wednesday.log
[student@desktopX glob]$ echo file{1..3}.txt
file1.txt file2.txt file3.txt
[student@desktopX glob]$ echo file{a..c}.txt
filea.txt fileb.txt filec.txt
[student@desktopX glob]$ echo file{a,b}{1,2}.txt
filea1.txt filea2.txt fileb1.txt fileb2.txt
[student@desktopX glob]$ echo file{a{1,2},b,c}.txt
filea1.txt filea2.txt fileb.txt filec.txt
[student@desktopX glob]$
```