

Directories and Files

Linux Administration

Asst. Prof. Ashwini Mathur

Overview

Goal :

To copy, move, create, delete, and organize files while working from the Bash shell prompt.

Objectives :

- Identify the purpose for important directories on a Linux system.
- Specify files using absolute and relative path names.
- Create, copy, move, and remove files and directories using command-line utilities.
- Match one or more file names using shell expansion as arguments to shell commands.

Linux File System Hierarchy

The file system hierarchy :

All files on a Linux system are stored on file systems which are reorganized into a single inverted tree of directories, known as a file system hierarchy.

Tree is inverted because the root of the tree is said to be at the top of the hierarchy, and the branches of directories and subdirectories stretch below the root.

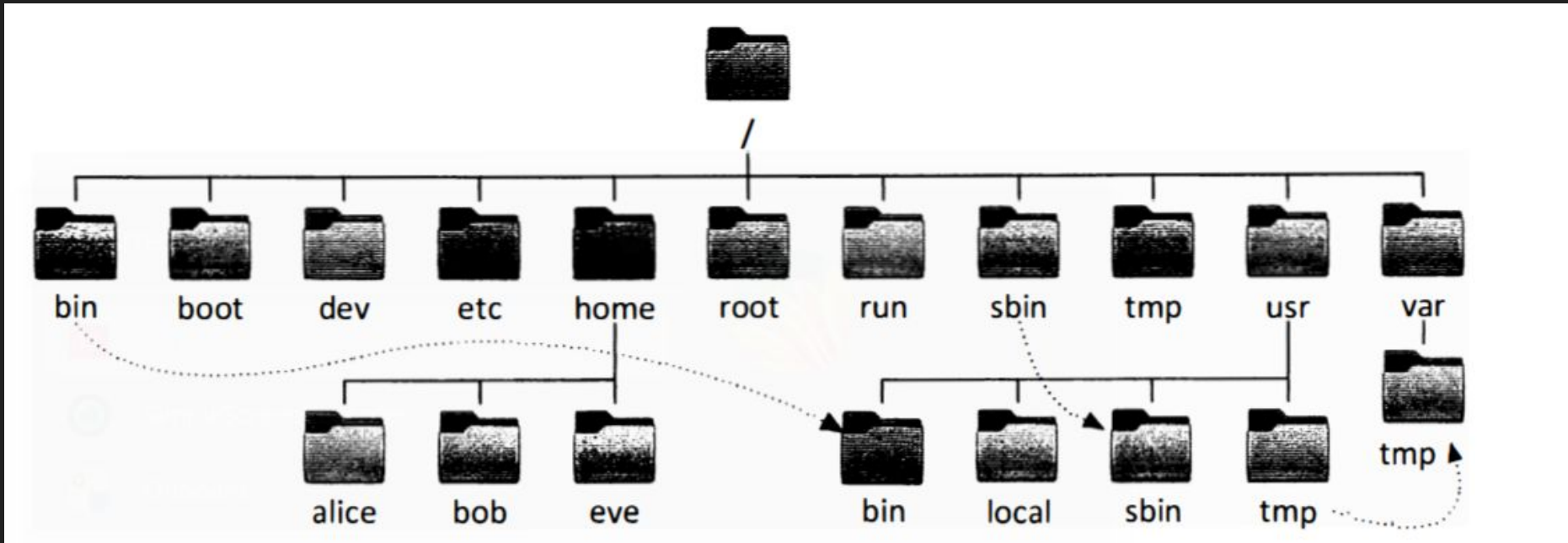


Figure 1. Significant file system directories

Significant File System Directories

The **directory** / is the root directory at the top of the file system hierarchy. The / character is a / so used as a directory separator in file names.

For example, if etc is a subdirectory of the / directory, we could call that directory /etc. Likewise, if the /etc directory contained a file named issue, we could refer to that file as /etc/issue.

Subdirectories of / are used for standardized purposes to organize files by type and purpose. This makes it easier to find files. For example, in the root directory, the subdirectory /boot is used for storing files needed to boot the system.

Important Note

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.

Important Red Hat Enterprise Linux directories

Location	Purpose
/usr	Installed software, shared libraries, include files, and static read-only program data. Important subdirectories include: <ul style="list-style-type: none">- /usr/bin: <i>User commands.</i>- /usr/sbin: <i>System administration commands.</i>- /usr/local: <i>Locally customized software.</i>
/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-spoiled 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	<i>Home directories</i> 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 <i>device files</i> which are used by the system to access hardware.

Practice : File system hierarchy

Match the following items to their counterparts in the table.

/	/etc	/home	/root	/run	/tmp	/usr
/usr/bin	/usr/sbin	/var				

Directory purpose	Location
This directory contains static, persistent system configuration data.	
This is the system's root directory.	
User home directories are located under this directory.	
This is the root account's home directory.	
This directory contains dynamic configuration data, such as FTP and websites.	
Regular user commands and utilities are located here.	
System administration binaries, for root use, are here.	
Temporary files are stored here.	

Directory purpose	Location
Contains dynamic, non-persistent application runtime data.	
Contains installed software programs and libraries.	

Match the following items to their counterparts in the table.

Directory purpose	Location
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

Locating files by Names .. Will see the demonstration also..

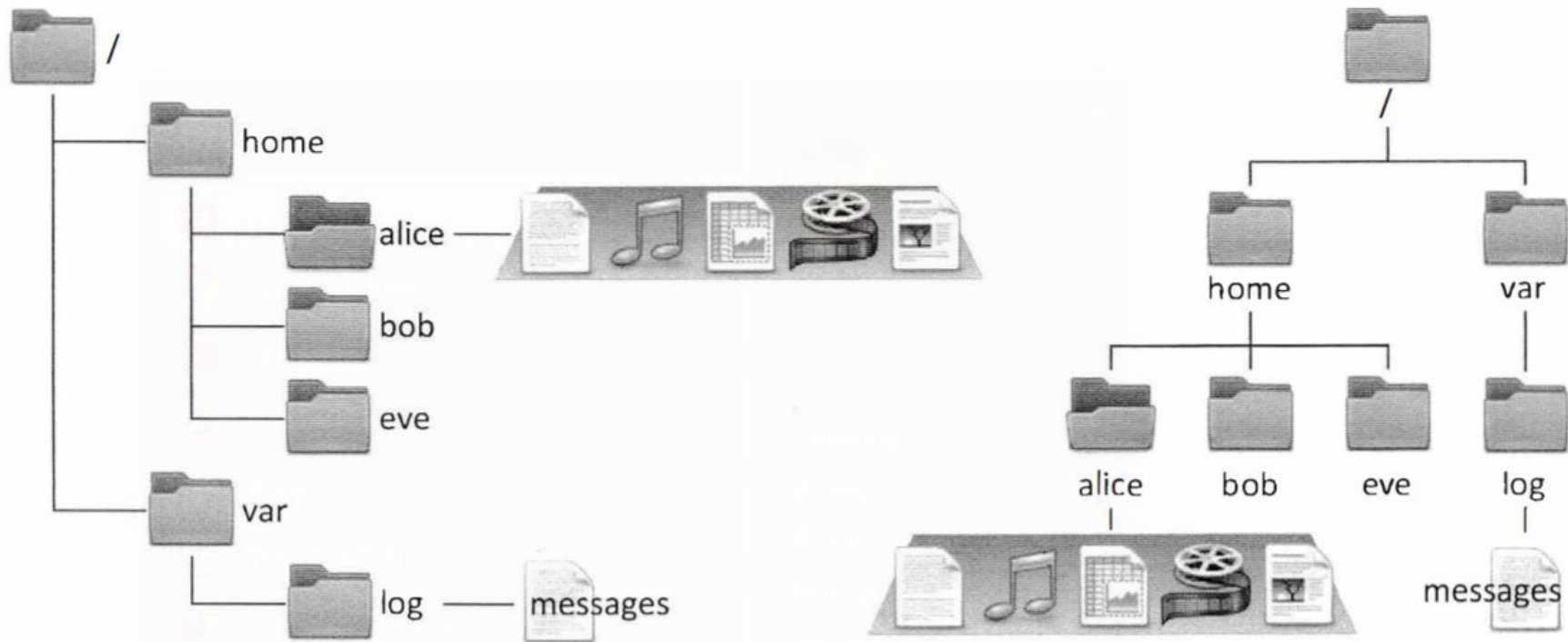


Figure 2.2: The common file browser view (left) is equivalent to the top-down view (right)

Absolute paths

An absolute path is a **fully qualified name**, **beginning at the root (!) directory** and specifying each subdirectory traversed to reach and uniquely represent a single file.

#Every file in a file system has a unique absolute path name, recognized with a simple rule:-

A path name with a **forward slash (/) as the first character is an absolute path name**. For example, the absolute path name for the system message log file is /var /log/messages. Absolute path names can be long to type, so files may also be located relatively. When a user logs in and opens a command window, the initial location is normally the user's home directory.

Relative Paths

Like an absolute path, a relative path identifies a unique file, **specifying only the path necessary to reach the file from the working directory.**

Recognizing relative path names follows a simple rule: A path name with anything **other than a forward slash (/) as a first character is a relative path name.** A user in the /var directory could refer to the message log file relatively as log/messages.

Additional Note

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.

Navigating Paths

Commands

The `pwd` command displays the **full path name of the current location**, which helps determine appropriate syntax for reaching files using relative path names.

The `ls` command **lists directory contents** for the specified directory or, if no directory is given, for the current directory.

Practical Demonstration

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

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.	

Solutions

Solution

Match the following items to their counterparts in the table.

Action to accomplish	Command
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

Managing Files Using Command Line Tools

Command Line File Management

File management involves **creating, deleting, copying, and moving** files. Additionally, directories can be created, deleted, copied, and moved to help organize files logically.

When working at the command line, **file management requires awareness of the current working directory** to choose either absolute or relative path syntax as most efficient for the immediate task.

File management commands

Activity	Single source ^(note)	Multiple source ^(note)
Copy file	cp file1 file2	cp file1 file2 file3 dir ⁽⁵⁾
Move file	mv file1 file2 ⁽¹⁾	mv file1 file2 file3 dir ⁽⁴⁾
Remove file	rm file1	rm -f file1 file2 file3 ⁽⁵⁾
Create directory	mkdir dir	mkdir -p par1/par2/dir ⁽⁶⁾
Copy directory	cp -r dir1 dir2 ⁽²⁾	cp -r dir1 dir2 dir3 dir4 ⁽⁴⁾
Move directory	mv dir1 dir2 ⁽³⁾	mv dir1 dir2 dir3 dir4 ⁽⁴⁾
Remove directory	rm -r dir1 ⁽²⁾	rm -rf dir1 dir2 dir3 ⁽⁵⁾
Note:	<p>⁽¹⁾The result is a rename.</p> <p>⁽²⁾The "recursive" option is required to process a source directory.</p> <p>⁽³⁾If dir2 exists, the result is a move. If dir2 doesn't exist, the result is a rename.</p> <p>⁽⁴⁾The last argument must be a directory.</p> <p>⁽⁵⁾Use caution with "force" option; you will not be prompted to confirm your action.</p> <p>⁽⁶⁾Use caution with "create parent" option; typing errors are not caught.</p>	

Guided Exercise

Problem - 1

In your home directory, create sets of empty practice files to use for the remainder of this lab.

- Use the shell tab completion to locate and complete path names more easily.

1. Create six files with names of the form songX . mp3.
2. Create six files with names of the form snapX . jpg.
3. Create six files with names of the form filmX . avi. In each set, replace X with the numbers 1 through 6.

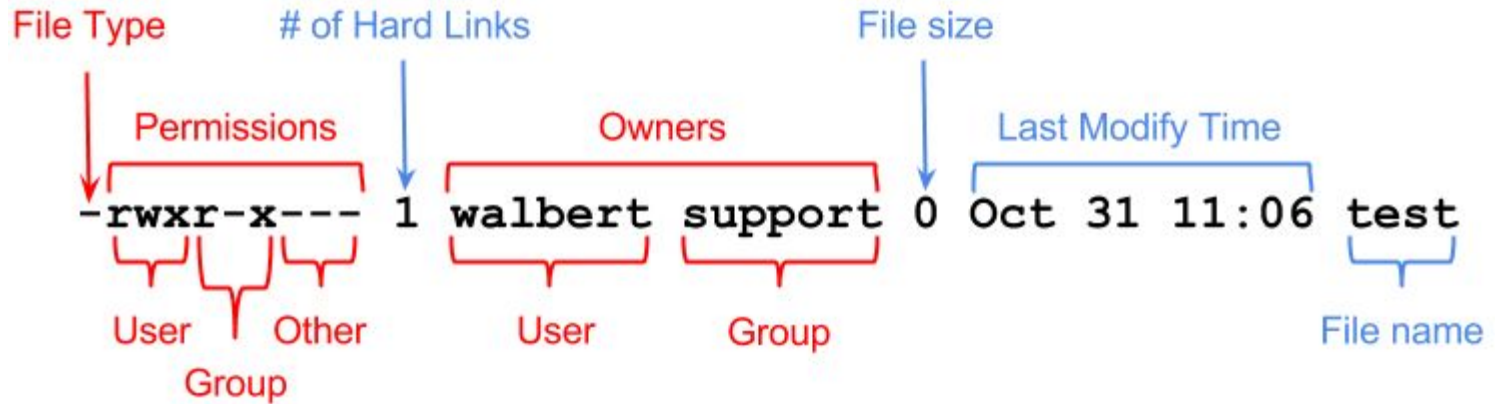
Problem - 2

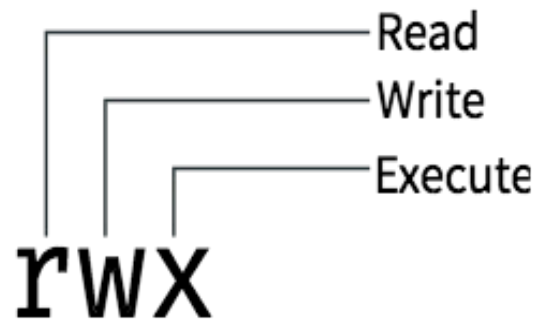
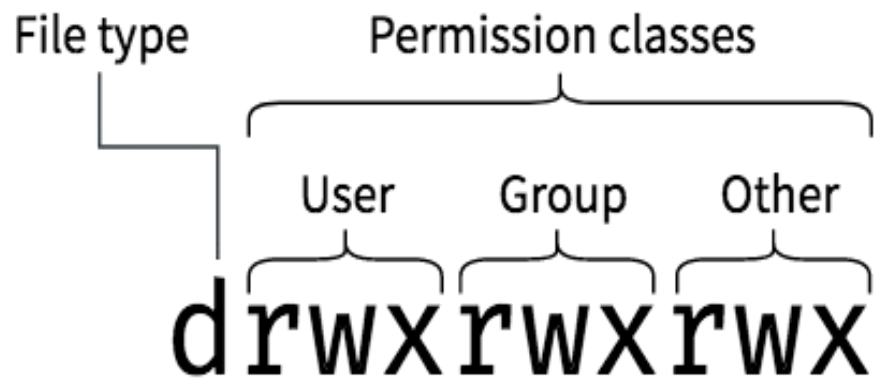
From your home directory, move the song files into your Music subdirectory, the snapshot files into your Pictures subdirectory, and the movie files into your Videos subdirectory.

When distributing files from one location to many locations, first change to the directory containing the source files. Use the simplest path syntax, absolute or relative, to reach the destination for each file management task.

File Permissions

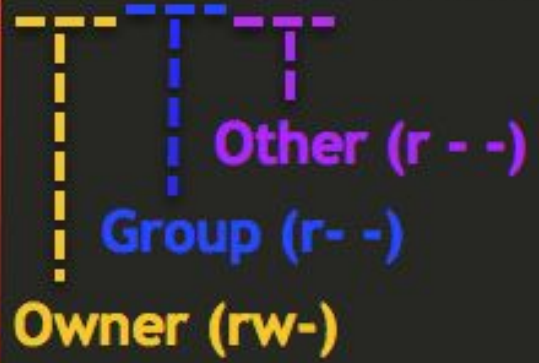
Command : ls -l





```
# ls -l file
```

```
-rw-r--r-- 1 root root 0 Nov 19 23:49 file
```



- `r` = Readable
- `w` = Writeable
- `x` = Executable
- `-` = Denied

Octals File Permissions

		u	g	o					
		7	5	4					
		/		\					
access	r	w	x	r	w	x	r	w	x
binary	4	2	1	4	2	1	4	2	1
enabled	1	1	1	1	0	1	1	0	0
result	4	2	1	4	0	1	4	0	0
total		7		5			4		

Understanding the permission syntax

To use `chmod` to set permissions, we need to tell it:

- *Who*: Who we are setting permissions for.
- *What*: What change are we making? Are we adding or removing the permission?
- *Which*: Which of the permissions are we setting?

We use indicators to represent these values, and form short “permissions statements” such as `u+x`, where “u” means “user” (who), “+” means add (what), and “x” means the execute permission (which).

The “who” values we can use are:

- *u*: User, meaning the owner of the file.
- *g*: Group, meaning members of the group the file belongs to.
- *o*: Others, meaning people not governed by the *u* and *g* permissions.
- *a*: All, meaning all of the above.

The “what” values we can use are:

- `-`: Minus sign. Removes the permission.
- `+`: Plus sign. Grants the permission. The permission is added to the existing permissions. If you want to have this permission and only this permission set, use the `=` option, described below.
- `=`: Equals sign. Set a permission and remove others.

The “which ” values we can use are:

- `r`: The read permission.
- `w`: The write permission.
- `x`: The execute permission.

Short Example: Using Octal Permission or using "who what which" Rules. There are many more ways ...

```
ashu@AshwiniMathur: ~/Desktop
ashu@AshwiniMathur:~/Desktop$ cd Desktop/
ashu@AshwiniMathur:~/Desktop$ touch Demo_file.txt
ashu@AshwiniMathur:~/Desktop$ ls -l Demo_file.txt
-rw-rw-r-- 1 ashu ashu 0 Aug 12 15:11 Demo_file.txt
ashu@AshwiniMathur:~/Desktop$ chmod 777 Demo_file.txt
ashu@AshwiniMathur:~/Desktop$ ls -l Demo_file.txt
-rwxrwxrwx 1 ashu ashu 0 Aug 12 15:11 Demo_file.txt
ashu@AshwiniMathur:~/Desktop$ chmod u=rw,og=r Demo_file.txt
ashu@AshwiniMathur:~/Desktop$ ls -l Demo_file.txt
-rw-r--r-- 1 ashu ashu 0 Aug 12 15:11 Demo_file.txt
ashu@AshwiniMathur:~/Desktop$
```

Permission string	Octal code	Meaning
<code>rwxrwxrwx</code>	777	Read, write, and execute permissions for all users.
<code>rwxr-xr-x</code>	755	Read and execute permission for all users. The file's owner also has write permission.
<code>rwxr-x---</code>	750	Read and execute permission for the owner and group. The file's owner also has write permission. Users who aren't the file's owner or members of the group have no access to the file.
<code>rwx-----</code>	700	Read, write, and execute permissions for the file's owner only; all others have no access.
<code>rw-rw-rw-</code>	666	Read and write permissions for all users. No execute permissions for anybody.
<code>rw-rw-r--</code>	664	Read and write permissions for the owner and group. Read-only permission for all others.
<code>rw-rw----</code>	660	Read and write permissions for the owner and group. No world permissions.
<code>rw-r--r--</code>	644	Read and write permissions for the owner. Read-only permission for all others.
<code>rw-r-----</code>	640	Read and write permissions for the owner, and read-only permission for the group. No permission for others.
<code>rw-----</code>	600	Read and write permissions for the owner. No permission for anybody else.
<code>r-----</code>	400	Read permission for the owner. No permission for anybody else.

User and Group Management

Managing Users and Groups

There are four main user administration files :

/etc/passwd – Keeps the user account and password information. This file holds the majority of information about accounts on the Unix system.

/etc/shadow – Holds the encrypted password of the corresponding account. Not all the systems support this file.

/etc/group – This file contains the group information for each account.

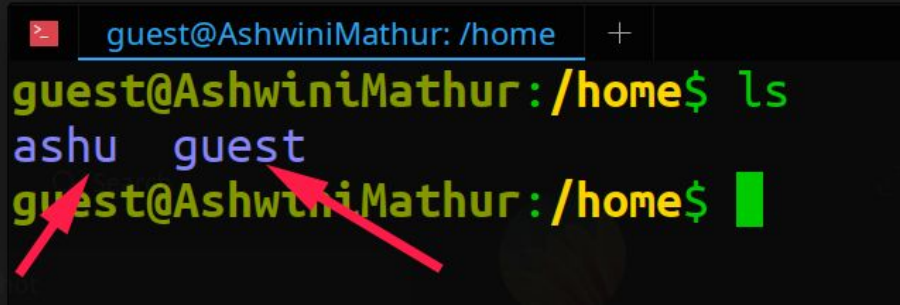
/etc/gshadow – This file contains secure group account information. Check all the above files using the cat command

Following Commands to Managing Users and Groups

- `adduser`: add a user to the system.
- `userdel`: delete a user account and related files.
- `addgroup`: add a group to the system.
- `delgroup`: remove a group from the system.
- `usermod`: modify a user account.
- `chage`: change user password expiry information.
- `sudo`: run one or more commands as another user (typically with superuser permissions).
- Relevant files: `/etc/passwd` (user information), `/etc/shadow` (encrypted passwords), `/etc/group` (group information) and `/etc/sudoers` (configuration for `sudo`).

```
ashu@AshwiniMathur:~$ sudo adduser guest
Adding user `guest' ...
Adding new group `guest' (1002) ...
Adding new user `guest' (1002) with group `guest' ...
Creating home directory `/home/guest' ...
Copying files from `/etc/skel' ...
New password:
Retype new password:
passwd: password updated successfully
Changing the user information for guest
Enter the new value, or press ENTER for the default
  Full Name []: Ashish
  Room Number []: 001
  Work Phone []: 7905281329
  Home Phone []:
  Other []:
Is the information correct? [Y/n] y
ashu@AshwiniMathur:~$
```

SU command : Switch user



```
guest@AshwiniMathur: /home +
guest@AshwiniMathur: /home$ ls
ashu  guest
guest@AshwiniMathur: /home$
```

The screenshot shows a terminal window with a dark background. The title bar at the top reads "guest@AshwiniMathur: /home" followed by a "+" icon. The terminal content shows a prompt "guest@AshwiniMathur: /home\$" followed by the command "ls". The output of the command is "ashu" and "guest". The prompt then changes to "guest@AshwiniMathur: /home\$". Two red arrows point to the "ashu" and "guest" output lines, and a green vertical bar is positioned at the end of the final prompt line.

Pattern matching Globbing is a shell command-parsing operation that expands a wildcard pattern into a list of matching path names.

Command-line meta-characters are replaced by the match list prior to command execution.

Patterns, especially square-bracketed character classes, that do not return matches display the original pattern request as literal text. The following are common metacharacters and pattern classes.

Pattern	Matches
*	Any string of 0 or more characters.
?	Any single character.
~	The current user's home directory.
~ <i>username</i>	User <i>username</i> 's home directory.
~+	The current working directory.
~-	The previous working directory.
[<i>abc...</i>]	Any one character in the enclosed class.
[! <i>abc...</i>]	Any one character <i>not</i> in the enclosed class.
[^ <i>abc...</i>]	Any one character <i>not</i> 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.