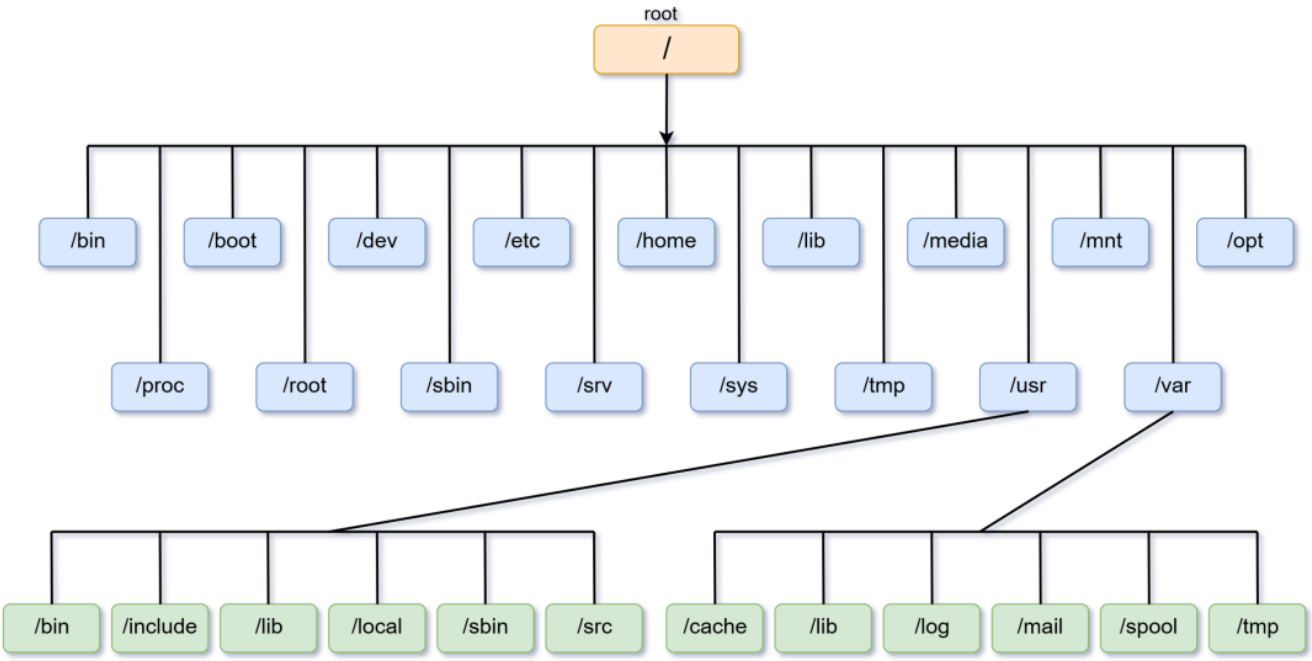
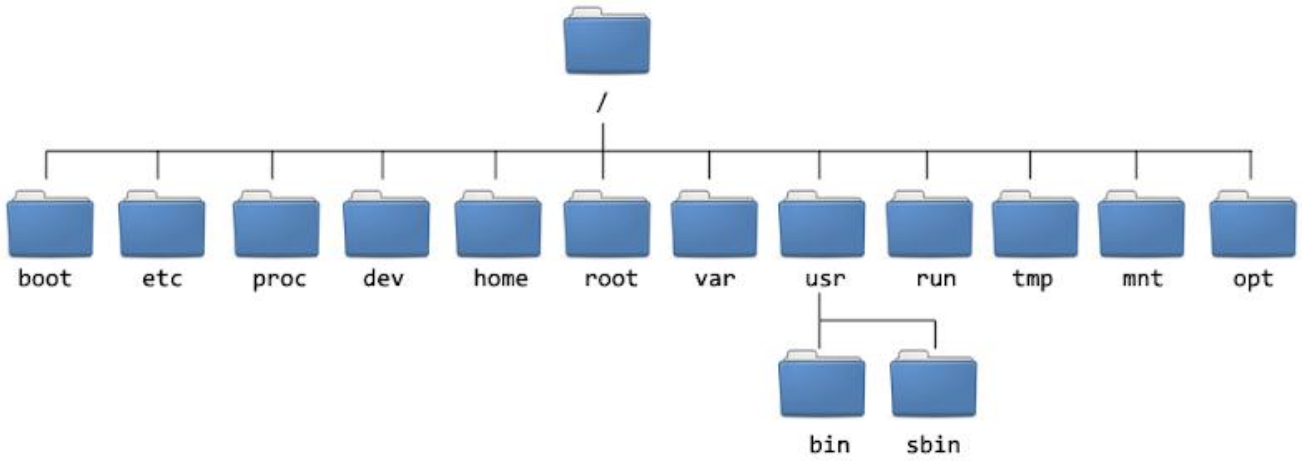
Unix / Linux - File Management





In Unix, there are three basic types of files −

- **Ordinary Files** − An ordinary file is a file on the system that contains data, text, or program instructions. In this tutorial, you look at working with ordinary files.
- **Directories** − Directories store both special and ordinary files. For users familiar with Windows or Mac OS, Unix directories are equivalent to folders.
- **Special Files** − Some special files provide access to hardware such as hard drives, CD-ROM drives, modems, and Ethernet adapters. Other special files are similar to aliases or shortcuts and enable you to access a single file using different names.

Listing Files

```
$ls
```

To list the files and directories stored in the current directory, use the following command

```
[hayder@fedora29 /]$ ls
bin   dev  home  lib64       media  opt   root  sbin  sys  usr
boot  etc  lib   lost+found  mnt    proc  run   srv   tmp  var
```

The command **ls** supports the **-l** option which would help you to get more information about the listed files

The command **ls** supports the **-h** option which would help you to get more information and human readable to the listed files

```
[hayder@fedora29 /]$ ls -l
total 68
lrwxrwxrwx.   1 root root     7 Jul 13  2018 bin -> usr/bin
dr-xr-xr-x.   7 root root  4096 Feb 14 17:10 boot
drwxr-xr-x.  22 root root  4420 Feb 22 09:26 dev
drwxr-xr-x.   2 root root  4096 Feb 22 15:56 dir1
drwxr-xr-x. 160 root root 12288 Feb 22 13:28 etc
-rw-r--r--.   1 root root     0 Feb 22 15:56 file.txt
drwxr-xr-x.   5 root root  4096 Feb 19 19:08 home
lrwxrwxrwx.   1 root root     7 Jul 13  2018 lib -> usr/lib
.
drwxr-xr-x.  13 root root  4096 Dec 17 21:06 usr
drwxr-xr-x.  22 root root  4096 Dec 17 21:14 var
```

```
[hayder@fedora29 /]$ ls -hl
total 68K
lrwxrwxrwx.   1 root root    7 Jul 13  2018 bin -> usr/bin
dr-xr-xr-x.   7 root root 4.0K Feb 14 17:10 boot
drwxr-xr-x.  22 root root 4.4K Feb 22 09:26 dev
drwxr-xr-x.   2 root root 4.0K Feb 22 15:56 dir1
drwxr-xr-x. 160 root root  12K Feb 22 13:28 etc
.
.
drwxrwxrwt.  24 root root  500 Feb 22 15:56 tmp
drwxr-xr-x.  13 root root 4.0K Dec 17 21:06 usr
drwxr-xr-x.  22 root root 4.0K Dec 17 21:14 var
[hayder@fedora29 /]$
```

Here is the information about all the listed columns −

- **First Column** − Represents the file type and the permission given on the file. Below is the description of all type of files.
- **Second Column** − Represents the number of memory blocks taken by the file or directory.
- **Third Column** − Represents the owner of the file. This is the Unix user who created this file.
- **Fourth Column** − Represents the group of the owner. Every Unix user will have an associated group.
- **Fifth Column** − Represents the file size in bytes.
- **Sixth Column** − Represents the date and the time when this file was created or modified for the last time.
- **Seventh Column** − Represents the file or the directory name.

In the **ls -l** listing example, every file line begins with a **d**, **-**, or **l**. These characters indicate the type of the file that's listed.

No.		Prefix & Description
1	**-**	Regular file, such as an ASCII text file, binary executable, or hard link
2	**b**	Block special file. Block input/output device file such as a physical hard drive
3	**c**	Character special file. Raw input/output device file such as a physical hard drive.
4	**d**	Directory file that contains a listing of other files and directories.
5	**l**	Symbolic link file. Links on any regular file.
6	**p**	Named pipe. A mechanism for interprocess communications.
7	s	Socket used for interprocess communication.

Metacharacters -Wildcards -globbing patterns

Mark	name	mean
.	dot	will match *any single character*
\	backslash	is used as an "escape" character, i.e. to protect a subsequent special character. Thus, "\\" searches for a backslash. Note you may need to use quotation marks and backslash(es)
.*	Dot and asterisk	is used to match any string, equivalent to * in standard wildcards.
*	asterisk	the proceeding item is to be matched *zero or more* times
^	caret	means "the beginning of the line". So "^a" means find a line starting with an "a".
$	Dollar sign	means "the end of the line". So "a$" means find a line ending with an "a". cat myfile \| grep '^s.*n$'
[ ]	Square brackets	specifies a range. If you did m[a,o,u]m it can become: mam, mum, mom if you did: m[a-d]m it can become anything that starts and ends with m and has any character a to d inbetween. For example, these would work: mam, mbm, mcm, mdm. This kind of wildcard specifies an “or” relationship (you only need one to match
|	pipe	This wildcard makes a logical OR relationship between wildcards. This way you can search for something or something else (possibly using two different regular expressions). You may need to add a '\' (backslash) before this command to work, because the shell may attempt to interpret this as a pipe.
!	exclamation	logical NOT

For Example

```
[ali@fedora29 ~]$ ls
Desktop  dir2   dir33    doc3.txt   file1   file22  file4     Public
dir1     dir22  doc1.txt  Documents  file11  file3   Music     Templates
dir11    dir3   doc2.txt  Downloads  file2   file33  Pictures  Videos
```

```
[ali@fedora29 ~]$ ls *.txt
doc1.txt  doc2.txt  doc3.txt
```

```
[ali@fedora29 ~]$ ls d*.txt
doc1.txt  doc2.txt  doc3.txt
```

```
[ali@fedora29 ~]$ ls file*
file1  file11  file2  file22  file3  file33  file4
```

```
[ali@fedora29 ~]$ ls file?
file1  file2  file3  file4
```

```
[ali@fedora29 ~]$ ls file?1
file11
[ali@fedora29 ~]$ ls file??
file11  file22  file33
[ali@fedora29 ~]$
```

Hidden Files

An invisible file is one, that first character of which is the dot or the period character (.). Unix programs (including the shell) use most of these files to store configuration information.

Some common examples of the hidden files include the files −

- **.profile** − The Bourne shell (sh) initialization script
- **.kshrc** − The Korn shell (ksh) initialization script
- **.cshrc** − The C shell (csh) initialization script
- **.rhosts** − The remote shell configuration file

To list the invisible files, specify the **-a** option to **ls** −

```
[ali@fedora29 ~]$ ls -a
.             .bash_profile  Desktop  dir22     doc2.txt   .esd_auth  file22  .filex1      .mozilla  Public     .xauth7aXrHb
..            .bashrc        dir1     dir3      doc3.txt   file1      file3   .filex2      Music     .ssh
.bash_history  .cache        dir11    dir33     Documents  file11     file33  .ICEauthority  Pictures  Templates
.bash_logout  .config        dir2     doc1.txt  Downloads  file2      file4   .local       .pki      Videos
```

- **Single dot (.)** − This represents the current directory.
- **Double dot (..)** − This represents the parent directory.

Creating Files

You can use the **vi** editor to create ordinary files on any Unix system. You simply need to give the following command

```
[ali@fedora29 ~]$ vi filetest.txt
```

The above command will open a file with the given filename. Now, press the key **i** to come into the edit mode. Once you are in the edit mode, you can start writing your content in the file as in the following program

```
this is my first file created on unix operating system

~
~
~
~
~
~
```

Once you are done with the program, follow these steps −

- Press the key **esc** to come out of the edit mode.
- Press **:** and type **wq** to save and quit
- press **q!** To quit without saving

Once the file is opened, you can come in the edit mode by pressing the key **i** and then you can proceed by editing the file. If you want to move here and there inside a file, then first you need to come out of the edit mode by pressing the key **Esc**. After this, you can use the following keys to move inside a file

- **l** key to move to the right side.
- **h** key to move to the left side.
- **k** key to move upside in the file.
- **j** key to move downside in the file.

Create directories

There are many different ways to create a directory , and subdirectory on a computer

```
[ali@fedora29 Downloads]$ mkdir dir1 dir2 dir3
[ali@fedora29 Downloads]$ ll
total 12
drwxrwxr-x. 2 ali ali 4096 Mar  2 11:23 dir1
drwxrwxr-x. 2 ali ali 4096 Mar  2 11:23 dir2
drwxrwxr-x. 2 ali ali 4096 Mar  2 11:23 dir3
```

```
[ali@fedora29 Downloads]$ mkdir  dir4/dir5/dir6
mkdir: cannot create directory ‘dir4/dir5/dir6’: No such file or directory
[ali@fedora29 Downloads]$ mkdir -p   dir4/dir5/dir6
```

```
[ali@fedora29 Downloads]$ ls -Rl .
.:
total 16
drwxrwxr-x. 2 ali ali 4096 Mar  2 11:23 dir1
drwxrwxr-x. 2 ali ali 4096 Mar  2 11:23 dir2
drwxrwxr-x. 2 ali ali 4096 Mar  2 11:23 dir3
drwxrwxr-x. 3 ali ali 4096 Mar  2 11:26 dir4

./dir1:
total 0

./dir2:
total 0

./dir3:
total 0

./dir4:
total 4
drwxrwxr-x. 3 ali ali 4096 Mar  2 11:26 dir5

./dir4/dir5:
total 4
drwxrwxr-x. 2 ali ali 4096 Mar  2 11:26 dir6

./dir4/dir5/dir6:
total 0
```

Display Content of a File

You can use the cat command to see the content of a file. Following is a simple example to see the content of the above created file

```
[ali@fedora29 ~]$ cat filetest1.txt this is my first file
created on unix operating system
```

Counting Words in a File

You can use the **wc** command to get a count of the total number of lines, words, and characters contained in a file. Following is a simple example to see the information about the file created above

```
[ali@fedora29 ~]$ wc filetest1.txt
 1 10 55 filetest1.txt
```

Here is the detail of all the four columns −

- **First Column** − Represents the total number of lines in the file.
- **Second Column** − Represents the total number of words in the file.
- **Third Column** − Represents the total number of bytes in the file. This is the actual size of the file.
- **Fourth Column** − Represents the file name.

You can give multiple files and get information about those files at a time. Following is simple syntax

Copying Files

To make a copy of a file use the **cp** command. The basic syntax of the command is

```
[ali@fedora29 ~]$ cp filetest1.txt copyfiletest2.txt
```

Renaming Files

To change the name of a file, use the **mv** command. Following is the basic syntax

```
$ mv old_file new_file
```

```
[ali@fedora29 ~]$ mv filetest1.txt newfiletest2.txt
```

Remove files and directories

To delete an existing file, use the **rm** command. Following is the basic syntax

```
$ rm filename
$rm -f file1 file2 file3      (force remove  file1 file2 file3)
$rm -d dir1      (removes an empty directory)
$ rmdir dir1      (removes an empty directory)
$rm -rf dir1     (removes a non-empty directory)
```

```
[ali@fedora29 ~]$ rm filetest1.txt
```

```
[ali@fedora29 Downloads]$ ll
total 16
drwxrwxr-x. 2 ali ali 4096 Mar  2 11:23 dir1
drwxrwxr-x. 2 ali ali 4096 Mar  2 11:23 dir2
drwxrwxr-x. 2 ali ali 4096 Mar  2 11:23 dir3
drwxrwxr-x. 3 ali ali 4096 Mar  2 11:26 dir4
[ali@fedora29 Downloads]$
[ali@fedora29 Downloads]$ rm -d dir1/
[ali@fedora29 Downloads]$ ls
dir2  dir3  dir4
[ali@fedora29 Downloads]$ rmdir  dir2/
[ali@fedora29 Downloads]$ ls
dir3  dir4
[ali@fedora29 Downloads]$ rm -d  dir4/
rm: cannot remove 'dir4/': Directory not empty
[ali@fedora29 Downloads]$ rm -rf  dir4/
```

Create a Soft link

The command “ln -s” offers you a solution by letting you create a soft link. The **ln** command in Linux creates links between files/directory. The argument “s” makes the the link symbolic or soft link instead of hard link.

```
ln -s /my/long/path/to/the/directory easyPath
```

we will see the new simple path directory in the current directory. And the simple path will be linked to the long path, you can see that by using “ls -l”, like

```
easyPath -> /my/long/path/to/the/directory
```

```
[ali@fedora29 ~]$ ls
Desktop  dir2   dir33    doc3.txt   error.txt  file2   file33        foo.txt     Pictures    script3.sh  script.sh  users
dir1     dir22  doc1.txt  Documents  file1      file22  file4         Music       Public      script4.sh  Templates  Videos
dir11    dir3   doc2.txt  Downloads  file11     file3   filetest1.txt  output.txt  script2.sh  script5.sh  test.txt
[ali@fedora29 ~]$ ln -s /dir1/ mydir
[ali@fedora29 ~]$ ls
Desktop  dir2   dir33    doc3.txt   error.txt  file2   file33        foo.txt  output.txt  script2.sh  script5.sh  test.txt
dir1     dir22  doc1.txt  Documents  file1      file22  file4         Music    Pictures    script3.sh  script.sh   users
dir11    dir3   doc2.txt  Downloads  file11     file3   filetest1.txt  mydir    Public      script4.sh  Templates   Videos
```

```
[ali@fedora29 ~]$ ls -l
-rw-rw-r--. 1 ali ali   55 Feb 22 17:19 filetest1.txt
-rw-rw-r--. 1 ali ali   15 Feb 25 20:14 foo.txt
drwxr-xr-x. 2 ali ali 4096 Feb 19 19:11 Music
lrwxrwxrwx. 1 ali ali    6 Feb 28 18:45 mydir -> /dir1/
-rw-rw-r--. 1 ali ali   40 Feb 25 20:31 output.txt
```

Remove a Soft link

In case you decide to remove the soft or symbolic link, it is pretty easy to do. There are two linux commands you can use to remove soft link
One is simply use "rm" command

```
$rm easyPath
$unlink easyPath
```

```
[ali@fedora29 ~]$ rm -r  mydir
```

note that for remove directories use -r option

```
[ali@fedora29 ~]$ unlink mydir
```

Find Linux Files by Name or Extension

`find` expressions take the following form:

```
find options starting/path expression
```

- The `options` attribute will control the behavior and optimization method of the `find` process.
- The `starting/path` attribute will define the top level directory where `find` begins filtering.
- The `expression` attribute controls the tests that search the directory hierarchy to produce output.

find syntax

```
find [-H] [-L] [-P] [-D debugopts] [-Olevel] [path...] [expression]
```

Simple **find** commands don't require a lot more effort, but they do require a starting point for the search and some kind of search criteria. The simplest find command

```
[ali@fedora29 ~]$ ll
total 100
drwxr-xr-x. 2 ali    ali    4096 Feb 19 19:11 Desktop
drwxrwxr-x. 2 ali    ali    4096 Feb 22 16:30 dir1
drwxrwxr-x. 2 ali    ali    4096 Feb 22 16:30 dir11
drwxrwxr-x. 2 ali    ali    4096 Feb 22 16:30 dir2
drwxrwxr-x. 2 ali    ali    4096 Feb 22 16:30 dir22
drwxrwxr-x. 2 ali    ali    4096 Feb 22 16:30 dir3
drwxrwxr-x. 2 ali    ali    4096 Feb 22 16:30 dir33
-rw-rw-r--. 1 ali    ali      18 Feb 25 18:57 doc1.txt
-rw-rw-r--. 1 ali    ali       0 Feb 22 16:32 doc2.txt
-rw-rw-r--. 1 ali    ali       0 Feb 22 16:32 doc3.txt
drwxr-xr-x. 2 ali    ali    4096 Feb 19 19:11 Documents
drwxr-xr-x. 2 ali    ali    4096 Feb 19 19:11 Downloads
-rw-rw-r--. 1 ali    ali      40 Feb 25 20:25 error.txt
-rw-rw-r--. 1 ali    ali       0 Feb 22 16:29 file1
-rw-rw-r--. 1 ali    ali       0 Feb 22 16:29 file11
-rw-rw-r--. 1 ali    ali       0 Feb 22 16:29 file2
-rw-rw-r--. 1 ali    ali       0 Feb 22 16:29 file22
-rw-rw-r--. 1 ali    ali       0 Feb 22 16:29 file3
-rw-rw-r--. 1 ali    ali       0 Feb 22 16:29 file33
-rw-rw-r--. 1 ali    ali       0 Feb 22 16:29 file4
-rw-rw-r--. 1 ali    ali      55 Feb 22 17:19 filetest1.txt
-rw-rw-r--. 1 ali    ali      15 Feb 25 20:14 foo.txt
drwxr-xr-x. 2 ali    ali    4096 Feb 19 19:11 Music
lrwxrwxrwx. 1 ali    ali       6 Feb 28 18:59 mydir -> /dir1/
-rw-rw-r--. 1 ali    ali      40 Feb 25 20:31 output.txt
drwxr-xr-x. 2 ali    ali    4096 Feb 19 19:11 Pictures
drwxr-xr-x. 2 ali    ali    4096 Feb 19 19:11 Public
-rwxrwxr-x. 1 ali    ali     136 Feb 25 14:43 script2.sh
-rwxrwxr-x. 1 ali    ali     127 Feb 25 19:41 script3.sh
-rwxrwxr-x. 1 ali    ali     117 Feb 25 19:21 script4.sh
-rwxrwxr-x. 1 ali    ali      82 Feb 25 19:34 script5.sh
-rwxrwxr-x. 1 ali    ali      99 Feb 25 14:32 script.sh
drwxr-xr-x. 2 ali    ali    4096 Feb 19 19:11 Templates
-rw-rw-r--. 1 ali    ali       0 Feb 25 14:44 test.txt
-rw-r--r--. 1 root   root      0 Mar  1 19:42 userfile2.txt
-rw-rw-r--. 1 hayder hayder    0 Mar  1 19:38 userfile.txt
-rw-rw-r--. 1 ali    ali      14 Feb 24 20:26 users
drwxr-xr-x. 2 ali    ali    4096 Feb 19 19:11 Videos
```

with option -name

```
[ali@fedora29 ~]$ ls
[ali@fedora29 ~]$ find . -name  file1
./file1
```

Searching from the current position in the file system by file name as shown will also involve searching all subdirectories unless a search depth is specified

More than just file names

The **find** command allows you to search on a number of criteria beyond just file names. These include file owner, group, permissions, size, modification time, lack of an active owner or group and file type. And you can do things beyond just locating the files. You can delete them, rename them, change ownership, change permissions, or run nearly any command against the located files.

with option -user

```
[ali@fedora29 ~]$ find . -user hayder -ls
 10617191      0 -rw-rw-r--   1  hayder   hayder          0 Mar  1 19:38 ./userfile.txt
```

with option -perm

```
[ali@fedora29 ~]$ find . -perm 640  -ls
 10617204      0 -rw-r-----   1  root     root            0 Mar  1 19:42 ./userfile2.txt
 10616949     76 -rw-r-----   1  ali      ali         75517 Feb 19 19:12 ./.local/share/tracker/data/tracker-store.ontology.journal
 10616961    140 -rw-r-----   1  ali      ali        141941 Feb 19 19:12 ./.local/share/tracker/data/tracker-store.journal
```

this example is to be applied after viewing the next lecture ((file permission))