Linux Files System

Prabhjeet Singh

File and File System

1. know directory and files in the Linux distribution

$ man hier -> it will tell about file system in the linux.
Output –

NAME
hier - description of the filesystem hierarchy

DESCRIPTION
A typical Linux system has, among others, the following directories:

/ This is the root directory. This is where the whole tree starts.

/bin This directory contains executable programs which are needed in single user mode and to bring the system up or repair it.

/boot Contains static files for the boot loader. This directory holds only the files which are needed during the boot process. The map installer and configuration files should go to /sbin and /etc. The operating system kernel (initrd for example) must be located in either / or /boot.

/dev Special or device files, which refer to physical devices. See mknod(1).

/etc Contains configuration files which are local to the machine. Some larger software packages, like X11, can have their own subdirectories below /etc. Site-wide configuration files may be placed here or in /usr/etc. Nevertheless, programs should always look for these files in /etc and you may have links for these files to /usr/etc.

/home On machines with home directories for users, these are usually beneath this directory, directly or not. The structure of this directory depends on local administration decisions (optional).

/lib This directory should hold those shared libraries that are necessary to boot the system and to run the commands in the root filesystem.

/media This directory contains mount points for removable media such as CD and DVD disks or USB sticks. On systems where more than one device exists for mounting a certain type of media, mount directories can be created by appending a digit to the name of those available above starting with '0', but the unqualified name must also exist.

/mnt This directory is a mount point for a temporarily mounted filesystem. In some distributions, /mnt contains subdirectories intended to be
used as mount points for several temporary filesystems.

/opt This directory should contain add-on packages that contain static files.

/proc This is a mount point for the proc filesystem, which provides information about running processes and the kernel. This pseudo-filesystem is
described in more detail in proc(5).

/root This directory is usually the home directory for the root user (optional).

/run This directory contains information which describes the system since it was booted. Once this purpose was served by /var/run and programs
may continue to use it.

/sbin Like /bin, this directory holds commands needed to boot the system, but which are usually not executed by normal users.

/srv This directory contains site-specific data that is served by this system.

/sys This is a mount point for the sysfs filesystem, which provides information about the kernel like /proc, but better structured, following the
formalism of kobject infrastructure.

/tmp This directory contains temporary files which may be deleted with no notice, such as by a regular job or at system boot up.

/usr This directory is usually mounted from a separate partition. It should hold only shareable, read-only data, so that it can be mounted by various machines running Linux.

Commands to see the system files

1. $ less /etc/fstab
2. $ mount -> to show the currently mounted devices.
3. df -h -> displays disk file system space used by all the mounted partition.

   ```
   $ df -h
   Filesystem      Size  Used Avail Use% Mounted on
   udev            953M     0  953M   0% /dev
   tmpfs           198M  1.2M  197M   1% /run
   /dev/sda1        78G   19G   56G  25% /
   tmpfs           988M     0  988M   0% /dev/shm
   tmpfs           5.0M     0  5.0M   0% /run/lock
   tmpfs           198M   64K  198M   1% /run/user/1000
   ```

4. du -> displays the disk usage of files or directories on the disk

   ```
   du -sh < directory name>
   $ du -sh ~
   2.2G    /home/kali
   ```

Absolute and Relative path -:

Absolute Path -: trace path from the root folder to the resource.

```
$ ls /home/kali/linux_learning
combine1.txt  file1.txt  file2.txt
```

Relative path – starts tracing from current directory to the resource.

```
 $ ls kali/linux_learning
combine1.txt  file1.txt  file2.txt
```

```
$ ls ./linux_learning
combine1.txt  file1.txt  file2.txt
```

```
$ ls ../linux_learning
```

Commands for the files and directories.

1. **$ ls -l notes.txt ->** it will show the latest timestamp when it was edited. **-l is small L**

   ```
   $ ls -l file1.txt
   -rw-r--r-- 1 kali kali 21 Aug 29 06:04 file1.txt
   ```

2. **touch <filename>** -> it will create an empty new file if not existed.

```
$ touch file3.txt
```

3. **cp <source> <destination>** -> it will copy the source file or directory into destination file or directory.

   ```
   ┌──(kali㉿kali)-[~/linux_learning]
   └─$ cp file1.txt file3.txt

   ┌──(kali㉿kali)-[~/linux_learning]
   └─$ ls
   combine1.txt  file  file1.txt  file2.txt  file3.txt

   ┌──(kali㉿kali)-[~/linux_learning]
   └─$ cat file1.txt
   hello
   world
   good bye

   ┌──(kali㉿kali)-[~/linux_learning]
   └─$ cat file3.txt
   hello
   world
   good bye
   ```

4. **mv <source> <destination>**
   move source files or directory to the destination. It will cut the source file/folder and paste at the destination.

5. Remove directory or file

$ rm <file>   -> remove/delete the file or directory.

```
$ rm file3.txt
```

Remove non-empty directory -> $ rm -rf <directory name>

Remove empty directory -> $ rmdir <directory name>

6. Make directory

$mkdir <directory name> -> command to create an empty directory.

7. Space in directory or file names.
   $ ls
   'File space.txt' 'dir name'
   $ cat file space.txt – throws error
   $ cd dir name -> throws error
   So we can resolve this issue by below two ways.
   1. Use \ char -> it will skip 1 spce at a time , if more spces re present than we need more \ char before each space

      ```
      $ cat File\ space.txt

      $ cd dir\ name
      ```

   2. Use double quotes ""

      ```
      $ cat "File space.txt"
      $ cd "dir name"
      ```

Files and Path Expansion.

* -> for multiple character

? -> for single character

```
$ ls File*.txt

File1.txt File2.txt File12.txt Fileabc.txt, Fileq.txt

$ ls File?.txt

File1.txt File2.txt Fileq.txt
```

Search in sub directories

```
$ ls **/*.txt
```

It will show all txt file in current and sub directories.

globstar

```
$ shopt -s globstar
```

```
$ ls file[123].txt
```

Displays files whose name is file1, file2 and file3.

```
$ ls file[a-zA-Z].txt
```

Displays files whose name is like filea.txt, fileA.txt

Head and tail command

1. $ head file1.txt -> it shows first 10 lines of the file
2. $ head -5 file1.txt -> it shows first 5 lines of the file1

3. $ tail File1.txt -> it shows the last 10 lines of the file.
4. $ tail -5 file1.txt -> it shows last 5 lines of the file1

To watch file for changes, we can use tail command.

```
$ tail -f /var/log/auth.log
```

diff command

Find the difference between two files line by line.

```
└─$ diff file1.txt file2.txt
1,3c1,4
< hello
< world
< good bye
---
> this is
> another file
> to show you the
> demo
```

Hard and Soft links

Link is reference or pointer to a file or directory on the file system.

Both type of links created by ln command.

Hard link -

```
$ ln file1.txt file1_hard.txt
```

It will create one more copy file1.txt into file1_hard.txt

And if we change anything in the file1.txt, it will reflect in the file1_hard.txt

If we delete or move file1.txt, file1_hard.txt will be still present in the same directory.

Soft link –

```
$ ln -s file1.txt file_soft.txt
```

It will create one more copy file1.txt into file1_hard.txt

And if we change anything in the file1.txt, it will reflect in the file1_soft.txt

But if we delete or move file1.txt, file1_soft.txt will be deleted or not show.

Compressing and archiving Files:

Zip and unzip command for compressing the data.

Syntax for files-: $ zip <zipped folder name> <files names to zip one by one.>

Syntax -: $ unzip -l <zipped file name>

Syntax for directories-: $ zip -r <zipped folder name> <directories names to zip one by one.>

Syntax -: $ unzip -l <zipped file name>

```
$ zip zip__backup.zip file2.txt file3.txt space.txt
  adding: file2.txt (deflated 2%)
  adding: file3.txt (stored 0%)
  adding: space.txt (stored 0%)
```

```
$ unzip -l  zip__backup.zip
Archive:  zip__backup.zip
  Length      Date    Time    Name
---------  ---------- -----   ----
       44  2022-08-29 06:07   file2.txt
       21  2022-08-29 07:03   file3.txt
        0  2022-08-29 08:17   space.txt
---------                     -------
       65                     3 files
```

Tar command – To archive the data not to compress.

```
$ tar
```

```
$ gzip
```

```
$gunzip
```

Searching the File System.

find command – to search files in a directory hierarchy.

```
$ find . -name ‘file*.txt’
```

It will search all files named file*.txt in the current directory.

-name -> is to search case sensitive names

(.) dot used for the current directory.

```
$ find . -iname ‘file*.txt’
```

iname -> used to disable case sensitive search of files.

```
$locate file*.txt
```

It will search file*.txt in all directories.

```
$which ls
/usr/bin/ls
```

which returns the pathnames of the files (or links) which would be executed in the current environment, had its arguments been given as commands in a strictly POSIX-conformant shell. It does this by searching the PATH for executable files matching the names of the arguments. It does not canonicalize path names.

```
$whereis ls
ls: /usr/bin/ls /usr/share/man/man1/ls.1.gz
```

whereis locates the binary, source and manual files for the specified command names. The supplied names are first stripped of leading pathname components. Prefixes of s. resulting from use of source code control are also dealt with. whereis then attempts to locate the desired program in the standard Linux places, and in the places specified by $PATH and $MANPATH.