Linux Links

Soft Links
Aka Symbolic link
It is a pointer to the original file
Just like a shortcut in windows and smaller file size
Different inode number
If we delete the original file, the soft links will become useless.

Hard links
Different name of the same file
Same file size as original file
Same inode number
If the original file is deleted, the hardlinks will still contain the data that were in the original file

Inode

- It is a data structure in Unix that denotes a file or a directory on a file system.
- Every file in the system has an inode (Index node), (information node)
- Contains all file information except the file contents and name.
- It contains
  a. Mode - permissions and file descriptor(file type, directory, no. of link, block device)
  b. Owner Information
  c. Size
  d. location of the file on the disk
  e. Timestamp - creation & modification
  f. Data Blocks - Pointers to the actual data
- Just like a personal ID or a passport (Without a name)

Each Inode has a number that is used in the index table.
Linux kernel uses Inode number to access the contents of an Inode

```
ls -l
ls -i
```

```
touch abc.txt
stat abc.txt
```

```
df -h
df -i
df -hi
```

E.g
Softlinks
to list the inode info of a file

```
ls -li fileNmae
```

```
ln -s {path}/targetFile {path}/srcFile
```

Create file abc.txt and add some contents in it.

```
cat > abc.txt
some contents
```

Preview the contents of abc.txt

```
cat abc.txt
```

Create soft link

```
ln -s /home/student/abc.txt /tmp/slabc.txt
```

Print inode number and links count.

```
ls -li abc.txt
ls -li /tmp/slabc.txt
```

Print the contents of soft link file

```
cat /tmp/slabc.txt
```

Remove original file

```
rm abc.txt
```

Print the contents of the soft link file again. It shows error.

```
cat /tmp/slabc.txt
```

Hardlinks

```
ln {path}/targetFile {path}/srcFile
```

Create a file xyz.txt and add some contents.

```
cat > xyz.txt
some contents
```

Preview the file contents.

```
cat xyz.txt
```

Create hard link

```
ln /home/student/xyz.txt /home/student/hlxyz.txt
```

Print inode number and links count.

```
ls -li xyz.txt
```

```
ls -li hlxyz.txt
```

Print the contents of hard link file

```
cat hlxyz.txt
```

Remove the original file

```
rm xyz.txt
```

Print the contents of the hard link file again, it prints file contents.

```
cat hlxyz.txt
```

Types of files in Linux

Linux file types

Seven types of files in Linux

- Regular files
- Directories
- Character device files

A character special file represents a device that transfers data in bytes such as a monitor or a printer. These are also device files that provide unbuffered serial access to system hardware components. They work by providing a way of communication with devices by transferring data one character at a time.

Listing character files sockets in a directory:

```
ls -l /dev | grep "^c"
```

- Block device files

A block special file represents a device that transfers data in blocks such as a hard drive. These are device files that provide buffered access to system hardware components. They provide a method of communication with device drivers through the file system.
One important aspect about block files is that they can transfer a large block of data and information at a given time.

Listing block files sockets in a directory:

```
ls -l /dev | grep "^b"
```

- local domain sockets

These are files that provide a means of inter-process communication, but they can transfer data and information between processes running on different environments.

This means that sockets provide data and information transfer between processes running on different machines on a network.

An example to show the work of sockets would be a web browser making a connection to a web server.

```
ls -l /dev/ | grep "^s"
```

```
sudo find / -type s -ls
```

- Named pipes

These are files that allow inter-process communication by connecting the output of one process to the input of another.

A named pipe is actually a file that is used by two processes to communicate with each and it acts as a Linux pipe.

Listing pipes sockets in a directory:

```
ls -l | grep "^p"
```

```
sudo find / -type p -ls
```

- symbolic links

```
sudo find / -type l -ls | less
ls -l /dev/ | grep "^l"
```

I/O redirection

Redirection is a process where we can copy the output of any command(s), file(s) into a new file.
>
>>

Redirecting only error to a file "2>>"
Redirecting all the output to a file "&>>"

Standard Input/Output and Errors

Standard Input
1) wc <file_name>
→ gives no. of lines, words and no. of characters.
2) wc
→ Waits for the standard input if no argument is passed. Give some standard input and press ctrl+d to exit and see the output.

If you provide a file name, it operates on filename. If you do not give the file name, it operates on standard input.

Standard output
1) ls
2) whoami
3) date

Standard Errors
1) $ lss
→ command not found
2) $ ls /binn

→ No such file or directory
3) $ cat /etc/shadow
→ permission denied

Redirect standard input/output

1) $ date
$ date > command.out --> output goes to the file.
$ cat command.out --> Using a single greater sign, we simply overwrite contents.

Append
1) $ date >> command.out --> Appends the new contents in old contents, does not replace.
$ cat command.out
$ whoami >> command.out

Redirect standard error output.

1) $ lss 2> error.out --> standard error output redirected into the file. overwrites.
$ cat error.out

2) $ lss /binn 2>> error.out --> appends
$ cat error.out

3) $ lss 2> /dev/null
$ cat /dev/null
$ ls /binn 2> /dev/null
$ cat /dev/null

4) $ ls /usr/local
$ /binn

--> to run a command that might generate alot of standard error output that we want to ignore.

$ lss 2> /dev/null
$ cat /dev/null
$ ls /binn 2> /dev/null --> dev/null is the special that whatever is written to is essentially thrown away.
$ cat /dev/null

$ ls /usr/local
$ ls /binn   --> no such file or directory
$ ls /usr/local/ /binn > command.out   --. only the standard output goes to the file while standard error printed in the screen.

$ cat command.out

$ ls /usr/local /binn > command.out 2>&1   --> to redirect both. It's like make 2 which is the standard error go wherever standard output goes.

$ cat command.out

Absolute path and Relative path

Absolute path starts from /
Relative path starts from the present working directory .

Package management in Linux

APT Advanced Packaging Tool

• High level package manager.
• Provides a way to retrieve and install packages including dependencies.
• Works with the package proper name.
• Does not work with .deb.

apt-get install [package_name]
→ installs packages with dependencies.
apt-get remove [package_name]
→ removes packages but no dependencies.
apt autoremove
→ removes packages installed but no longer required.
apt-get clean
→ removes downloaded packages for .deb for software which is already installed.
apt-get purge [package_name]
→ combination of remove and clean.

APT

apt-get update
→ Reads the /etc/apt/sources.list file and updates the system's database of packages available for installation
apt-get upgrade
→ Upgrades all packages if there are updates available. Run this after running apt-get update.

APT-CACHE

apt-cache search [package_name]
→ Looks additional packages available. Useful when apt-get install fails.
apt-cache show [package_name]

→ Shows dependency information, version numbers and a basic description of the package

APTITUDE

• High level package manager.

• to perform management tasks (installing, upgrading, and removing packages, also handling dependency resolution automatically

aptitude update, aptitude install, aptitude remove, aptitude clean, or aptitude purge

→ Same as their apt-get counterparts.

aptitude search or aptitude show

→ Same as their apt-cache counterparts.

/etc/apt/sources.list

→ The file /etc/apt/sources.list controls repositories from which APT constructs its database.

→ Contains lines in the following format.

deb location-of-resources distribution component(s)

→ Needs apt-get update after changes in this file.

dpkg

• Low level package manager.

• It can install, remove, provide information about and build *.deb packages but it can't automatically download and install their corresponding dependencies.

dpkg -i install [package_file_name].deb

→ Installs a .deb file

dpkg --list search-pattern

→ Lists packages currently installed on the system.

dpkg --configure package-name(s)

→ Run a configuration interface to set up a package.

```
dpkg-reconfigure package-name(s)
```

→ Runs a configuration interface on an already installed package.