HARD LINKS

File can have multiple filenames. It means the file has more than one **link**. We can access the file by any of its links. All names provided to a single file have same inode number.

Creating Hard Links (ln)

ln command is used to create the both hard and soft link.

Example:

```
$ ls –li student.lst

29550  -rwxr-xr-x  1  students   students   900  July   10  10:02  student.lst
```

The above student.lst has inode number 29550 and symbolic link 1.

Now create a hard link using ln command.

```
$ ln student.lst stud_data.lst                              stud_data.lst must not exist
```

Hard link for student.lst is created. Stud_data.lst

```
$ ls  -li  student.lst  stud_data.lst

29550  -rwxr-xr-x  2  students   students   900  July   10  10:02  student.lst

29550  -rwxr-xr-x  2  students   students   900  July   10  10:02  stud_data.lst
```

The link count was 1 for student.lst, now it becomes 2.

Generate another hard link.

```
$ ls  -li student.lst   stud_data.lst  stud1.lst

29550  -rwxr-xr-x  3  students   students   900  July   10  10:02  student.lst

29550  -rwxr-xr-x  3  students   students   900  July   10  10:02  stud_data.lst

29550  -rwxr-xr-x  3  students   students   900  July   10  10:02  stud1.lst
```

Now the link count becomes 3.

Remove link

The rm command is remove the link.

```
$ rm stud1.lst

$ ls  -li  student.lst  stud_data.lst

29550  -rwxr-xr-x  2  students   students   900  July   10  10:02  student.lst

29550  -rwxr-xr-x  2  students   students   900  July   10  10:02  stud_data.lst
```

The link count has now 2. Another rm will bring it down to one. A file is considered to be completely removed from the system when its link count drops to zero.

SYMOBLIC LINK OR SOFT LINK

The hard link has two limitations:

- You can not have two linked filenames in two file systems. In other word you can not link a file name in the /usr file system to another in the /home file system.
- You can not link a directory even within the same file system.

This serious limitation was overcome using symbolic link. The symbolic link is the forth file type. The symbolic link does not have the file's contents, but simply provided the pathname of the file that actually has the contents. A symbolic link is also known as a soft link.

Creating soft link

The ln command with –s option is used to create symbolic link.

Example :

```
$ ls –li student.lst

29550  -rwxr-xr-x  1  students   students   900  July   10  10:02  student.lst

$ ln –s student.lst stud_data.lst

$ ls –li student.lst stud_data.lst

29550  -rwxr-xr-x  1  students   students   900  July   10  10:02  student.lst

29560  -rwxr-xr-x  1  students   students     4   July   10  10:02  stud_data.lst -> student.lst
```

Both file have different inode number. The size is also different. The pointer stud_data.lst -> student.lst suggest that stud_data.lst contains the pathname for the filename student.lst.The student.lst file contain actual data. Where the stud_data.lst not contain the actual data but the path to the student.lst. So size of stud_data.lst is less than student.lst.

In soft link, now we have two different files and they are not identical. Removing stud_data.lst would not affect us much because we can easily recreate the link. But if we remove student.lst we would lost the file containing the data. In that case stud_data.lst would point to a non existent file and become a **dangling symbolic link.**

Symbolic links can also be used with relative pathnames. Unlike hard links, they can also span multiple file systems and also link directories.

locating files (find)

find recursively examines a directory tree to look for files matching some criteria, and then take some action on the selected files.

Syntax:

```
find   path_list    selection_criteria   action
```

The find work as follows:

- First, it recursively examines all files in the directories specified in path_list.
- It then matches each file for one or more selection_criteria
- Finally, it takes some action on those selected files.

The path_list comprises one or more subdirectories separated by whitespace. There is selection criteria you can use to match a file, and multiple action on a file.

Example:

use find command to locate all files named oddeven.sh

```
$ find  /  -name  oddeven.sh  -print

/home/amit/scripts/oddeven.sh

/home/sumit/scripts/oddeven.sh

/home/nilesh/oddeven.sh

/home/ramesh/scripts/report/oddeven.sh
```

The path list / indicate search should start from root directory.

Selection criteria is –name oddeven.sh (whose syntax is –name filename). If the filename found oddeven.sh matched the file is selected.

The third section specifies the action (-print) to be taken on files. It simply displays on terminal.

You can also **use relative names in path list** and also use **wild card pattern in selection criteria**.

Example:

$ find . –name "*.c" -print	Display all files with extension .c in current directory
$ find . –name '[A-Z]' -print	Display all files start with Capital in current directory Single quotes also works.

Locating a file by inode number (inum)

```
$ find /   -inum  13750   -print

find : can not read dir /usr/lost+found : permission denied

/usr/bin/gzip

/usr/bin/gunzip

/usr/bin/gzcat
```

find throws an error message when it cannot have access permission on particular directory. To avoid this message simply redirect the standard error to /dev/null.

File type and Permissions (-type and -perm)

The –type option followed by the letter f, d or l selects file of the ordinary, directory and symbolic link type.

Example:

To locate all directory of your home directory

```
$ cd

$ find  .  –type  d  -print  2>/dev/null
```

```
.
./.netscape
./students
./programs
```

The –perm option specifies the permissions to match.

Example:

Locate the files having read and write permission for all categories of users

```
$ find /  -perm 666  -print
```

Example:

Locate the directories in home directory which have read write and execute permissions.

```
$ find  $home  \( -perm  777 –a   -type   d  \)  -print
```

In above find uses AND condition (-a option) to select directories that provide all access rights to everyone. It select only those files only if both selection criteria (-perm and –type) are fulfilled.

Finding unused files (-mtime and –atime)

Some files in disk are unaccessed or unmodified for months-even years. find command's –mtime option can easily match file's modification and –a option match the access time to select those files.

Example:

–mtime helps in backup opterations by providing a list of those files that have been modified say in less than two days.

```
$ find  .  –mtime  -2  -print
```

- 2 here means less than 2 days.

Example:

To select from the /home directory all files that have not been accessed for more than a year, a positive value has to be used with –atime

```
$find  /home  -atime  +365  -print  | mailx  root
```

Because find uses standard output, the list can be stored in a file or used to mail a message.

+365 means greater than 365 days.

-365 means less than 365 days

365 means exactly 365 days.

The find operators (!, -o and –a)

! operator

In find command ! operator is used before an option to negate its meaning.

Example:

```
$ find . ! -name "*.c" -print
```

Select all files except C program files.

o operator

In find command, -o option is used to represent or condition

Example:

To look for both shell and perl script in home directory

```
$ find /home \( -name "*.sh" -o -name "*.pl" \)
```

-a operator

In find command, -a operator represent AND condition.

Example:

Locate the directories in home directory which have read write and execute permissions.

```
$ find $home \( -perm 777 –a -type d \) -print
```

Options in Action component

Displaying the listing (-ls)

Instead of list the filename only, some time you want to display detail list of selected files. It can be done using –ls option.

Example:

Give the detail list of files that modified in more than 2 days and less than 5 days.

```
$ find  .  –type  f  -mtime  +2  -mtime  -5  -ls                              -a option implied

477556   1   -rw-r- -r- -   1  students   bcagrp    710  Aug  15   10:05   ./bca/file.c
```

Taking action on selected files (-exec and –ok)

-exec option

-exec option lets you take any action by running a UNIX command on the selected files. –exec takes the command to execute as its own argument, followed by { } and finally cryptic symbols \;
Example:

Remove those files which are not accessed in more than one year.

```
$  find  $HOME  -type  f  -atime  +365  -exec  rm { }  \;
```

-ok option

-ok option is an interactive option which get confirmation before any action.

Example:

Move the files in to $home/safe which do not use in more than one year.

```
$ find  $HOME  -type  -f  -atime   +365  -ok  mv { }  $home/safe \;

<mv  …  ./archive.tar.gz  > ? y

< mv …  ./f1.txt  >  ? n
```

mv turns interactive with -i but only if the destination file exists. Here, -ok seeks confirmation for every selected file to be moved to the $HOME/safe directory irrespective of whether the files exist at the destination or not. A y delete the file.

Table: Expressions used by find

Selection Criteria	**Select File**
-inum n	Having inode number n
-type x	If of type x, x can be f (ordinary file), d (directory file) or l (symbolic link file)
-type f	If an ordinary file
-perm nnn	Permission read,write and execute (octal permissions)
-links n	If having n links
-user usname	If owned by usname
-group gname	If owned by group name

-size +x[c]	If size greater then x block (characters if c is also specified)
-mtime -x	If modified in less than x days
-newer flname	If modified after flname
-mmin -x	If modified in less than x minutes
-atime +x	If accessed in more than x days
-amin +x	If accessed in more than x minutes
-name flname	flname
-iname flname	As above, but match is case insensitive
-follow	After following a symbolic link
-prune	But don’t descend directory if matched
-mount	But don’t look in other file system

Action	**Significance**
-print	Prints selected file on standard output
-ls	Executes ls –lids command on selected files
-exec cmd	Executes UNIX command cmd followed by { } \;