File and Directory operations

Session Objectives

- File system in Linux-Ext2
- Listing of files & directories
- Create, change, rename, remove &copy directories
- Create, rename, append, copy & remove files
- Wild card searching
- Head & tail commands

The Linux filesystem

The central feature of the Linux operating system is its hierarchical filesystem. The features are

- A hierarchical, unified filesystem (directories within directories; and files, directories, and device drivers are treated as files),
- Support of 256-character filenames (avoid symbols and punctuation except for the dot (.) and note that you can have more than one nonadjacent dots in the filename).
- All command line entries are case sensitive.
- Linux uses the slash (/) and not the backslash (\).

There are basically four kinds of files:

- **Ordinary files -** Text files (plain ASCII)
- **Data files -** Contain special characters not contained in the ASCII set
- **Command text files -** Shell scripts, and executable files (binaries); directories; links
- **Special device files** (physical hardware)

Ordinary users essentially only have to deal with two types of files:

- ***Plain files****:* files which contain specific information, such as plain text, C source code, object code, executable code, postscript code, *etc.*
- ***Directories****:* special files, which are essentially containers for other files (including other directories).

All Linux files are rooted in a special directory (the *root directory*) called “/”. Every file within the filesystem has an *absolute pathname* which begins with “/” and which describes the path from the root directory to the file in question. For instance

```
/usr/bin/man
```

refers to a file named ``man" which resides in a directory with absolute pathname `/usr/bin` which itself lives in directory `/usr` which is contained in the root directory `/`.

Alternatively, files may be uniquely specified using *relative pathnames*. The shell always knows your current location in the directory hierarchy, which is referred to as your *working directory*. The name of this directory can be printed using the **pwd** command:

```
% pwd
/usr
%
```

To refer to a filename such as
`man`
or
`man/man1/man.1`

so that the reference does not begin with “/” then the reference is identical to an absolute pathname constructed by prefixing the working directory followed by ``/" to the relative reference. Thus, assuming that your working directory is

```
/usr
```

then the two previous relative pathnames are identical to the absolute pathnames `/usr/man and /usr/man/man1/man.1` respectively.

Every user of a Linux system has a directory, called his/her *home directory*, which serves as the base of his/her personal files. The command **cd** (change [working] directory) with no arguments always takes you to your home directory.

Linux has a very fast file system called the **Extended File System Version 2 (EXT2)**. The EXT2 file system was created by Remy Card.

The Second Extended File system (EXT2)

Rémy Card devised the Second Extended File system as an extensible and powerful file system for Linux. It is also the most successful file system so far in the Linux community and is the basis for all of the currently shipping Linux distributions.

The EXT2 file system, like a lot of the file systems, is built on the premise that the data held in files is kept in data blocks. These data blocks are all of the same length and, although that length can vary between different EXT2 file systems the block size of a particular EXT2 file system is set when it is created (using mke2fs that is make extended 2 file system). Every file's size is rounded up to an integral number of blocks. If the block size is 1024 bytes, then a file of 1025 bytes will occupy two 1024 byte blocks. Unfortunately this means that on average you waste half a block per file. Usually in computing you trade off CPU usage for memory and disk space utilisation. In this case Linux, along with most operating systems, trades off a relatively inefficient disk usage in order to reduce the workload on the CPU. Not all of the blocks in the file system hold data, some must be used to contain the information that describes the structure of the file system.

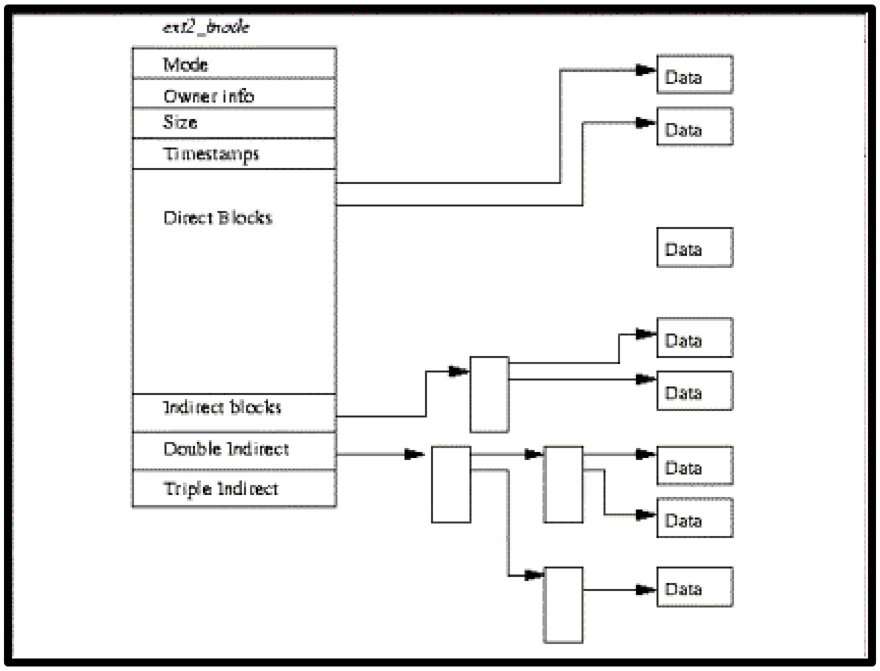
EXT2 defines the file system topology by describing each file in the system with an **inode** data structure.
An inode describes the following:

- the blocks occupied by the data within a file
- the access rights of the file,
- the file's modification times and
- the type of the file.

Every file in the EXT2 file system is described by a single inode and each inode has a single unique number identifying it. The inodes for the file system are all kept together in inode tables. EXT2 directories are simply special files (themselves described by inodes) which contain pointers to the inodes of their directory entries.

The figure given below shows the layout of the EXT2 file system as occupying a series of blocks in a block structured device. So far as each file system is concerned, block devices are just a series of blocks that can be read and written. A file system does not need to concern itself with where on the physical media a block should be put, that is the job of the device's driver. Whenever a file system needs to read information or data from the block device containing it, it requests that its supporting device driver reads an integral number of blocks. The EXT2 file system divides the logical partition that it occupies into Block Groups.

Each group duplicates information critical to the integrity of the file system as well as holding real files and directories as blocks of information and data. This duplication is necessary should a disaster occur and the file system need recovering.



In the EXT2 file system, the inode is the basic building block; every file and directory in the file system is described by one and only one inode. The EXT2 inodes for each Block Group are kept in the inode table together with a bitmap that allows the system to keep track of allocated and unallocated inodes

Listing of files, directories and subdirectories

The `ls` command can be used to list the contents of a directory. It displays the names of the files and subdirectories in a directory.

```
$ ls
myfile      first myfiles
```

Options with the ls command

Option	Function
`$ ls – l`	The option `ls –l` displays a detailed list of files and directories.
`$ ls –a`	The option `ls –a` displays all files and directories including hidden files.
`$ ls –A`	The option `ls –A` displays the files of all directories except the **.** and **..** directories.
`$ ls –F`	The option `ls –F` displays the file type along with the name of the file (directories appear with a / appended at the end, executable files appear with a *)

$ ls -r	The option `ls -r` displays the files and sub directories in the reverse order.
$ ls -R	The option `ls -R` displays the contents of the specified directory and all the sub-directories (Recursive listing)
$ ls -m	List the files as a single line with comma separations
$ ls -x	Sort the files horizontally instead of vertically.

It is possible to combine more than one option with the ls command. For example the command `ls -al` gives a detailed listing of all files including hidden files.

NOTE
The directories . and .. appear in all directory listings. A dot (.) represents the current directory and two dots (..) represent the parent directory. These two are hidden directories.

Directory commands

Creating a directory

The command mkdir is used to create a directory.

```
$ mkdir mydir
```

The above command creates a directory named `mydir` under the current directory. It is also possible to create a directory in a different path. In this case, the entire path has to be specified.

```
$ mkdir mydir/mysubdir
```

The above command will create a directory named `mysubdir` under the directory `mydir`.

Path of the directory

The command `pwd` is used to display the full path name of the current directory

```
$ pwd
/home/meerav
```

Changing the current directory

The command cd (change directory) is used to change the current directory to the directory specified by the user.

```
$ pwd
/home/meerav
```

```
$ cd mydir
$ pwd
/home/meerav/mydir
$ cd mysubdir
```

The user is now in the directory mysubdir. Now to move to the parent directory of the current directory, cd .. can be used. If no path is given after the cd command, the user is taken to the home directoy.

```
$ pwd
/home/meerav/mydir/mysubdir
$ cd ..
/home/meerav/mydir
```

NOTE
There should be a space between **cd** and **..** and no space between the two dots.

Renaming/Moving a directory

The mv command can be used to rename a directory or move a directory from one location to another.

For example, consider the following command to rename a directory:

```
$ mv  mydir firstdir
```

The above command renames the directory mydir as firstdir
The following command moves the directory mydir to the temp directory. Make sure the temp directory exists in the current directory.

```
$ mv mydir temp
```

Removing a directory

The command rmdir is used to remove the directory specified.

```
$ rmdir mydir
```

The above command deletes the mydir directory.

A directory can be deleted using the rmdir command only
- If it is not the current directory and
- If it is empty

To delete the directory `mysubdir` inside the `mydir` directory, the following command can be issued from the parent directory:

```
$ rmdir mydir/mysubdir
```

File commands

Creating an empty file

The `touch` command can be used to create an empty file.

```
$ touch emptyfile
$_
```

The above command creates a file named emptyfile.

Creating a file

The cat command can be used to create a file as shown below:

```
$ cat  >myfile
Rose is a beautiful flower.
Ctrl+d
$ _
```

Ctrl+d is used to come back to the shell prompt.

Viewing the contents of a file

The contents of a file can be viewed as shown below:

```
$ cat myfile
Rose is a beautiful flower.
$_
```

Appending to a file

The contents of one file can be appended to another file by the following command:

```
$ cat >>myfile
Lotus is our national flower.
Ctrl+d
$ _
```

Now view the contents of the file

```
$ cat myfile
Rose is a beautiful flower.
Lotus is our national flower.
$_
```

Viewing more than one file simultaneously

Create another file as shown below:

```
$ cat >yourfile
The tiger is our national animal.
Ctrl+d
```

The contents of more than one file can be viewed as shown below:

```
$ cat myfile yourfile
Rose is a beautiful flower
Lotus is our national flower
The tiger is our national animal
$ _
```

Viewing files with line numbers

To display the file with line numbers, the –n option can be used with the cat command.

```
$ cat –n myfile
1 Rose is a beautiful flower
2 Lotus is our national flower
$_
```

Copying files

The `cp` command is used to copy the contents of a source file to a destination file.

```
$ cp myfile myfile1
```

The above command copies the contents of `myfile` to a new file `myfile1`. If `myfile1` already exists, its contents will be overwritten.

To copy the contents of one directory (all its files and sub-directories) to another directory, the `cp` command can be used with the `–r` option.

```
$ cp –r mydir yourdir
```

The above command copies the `mydir` directory and all its files and sub-directories to the `yourdir` directory. If the directory `yourdir` already exists, all the contents are put inside that directory. Otherwise, a directory `yourdir` is created in the current directory.

Another option that can be used with cp is –i. When the cp command is used with the –i option, it asks for confirmation from the user before copying the file.(this option is called interactive copying)

Consider the following command

```
$ cp –i myfile myfile1
cp: overwrite ‘myfile1’? y
$_
```

Removing a file

The rm command is used to remove a file.

```
$ rm myfile
```

The above command deletes the file `myfile` if it is present in the current directory. More than one file can also be deleted simultaneously.

```
$ rm myfile yourfile
```

The above command will delete both the files `myfile` and `yourfile`.

Some of the common switches for the **rm** command are

- **rm -i** : This operates the rm command in interactive mode. It will prompt you before deleting a file. This gives you a second chance to say "no do not delete the file" or "yes delete the file".
- **rm -f** : Will force bypassing any safeguards that may be in place such as prompting. Again this command is handy to know but care should be taken with its use.
- **rm -r** : Will delete every file and sub directory below that in which the command was given. Be very careful with this command as no prompt will be given in most linux systems and it will mean instant good bye to your files if misused.

```
$ rm –r mydir
```

The above command deletes the `mydir` directory along with all its sub-directories.

NOTE

The `rmdir` command can delete a directory only if it is empty. If the subdirectories in a directory are to be deleted, the `rm –r` option can be used.

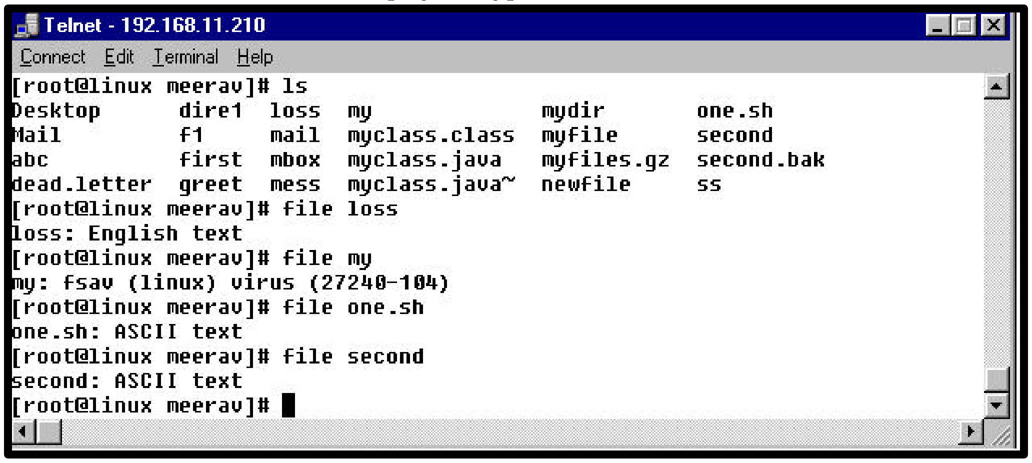
more

This command allows you too scroll through a file one screen at a time allowing you to more easily read the files contents. Some files are very big and using this command allows you to view the contents of large files more efficiently. To go forward one screen use the space bar and to go back one screen use the B key

```
$ more filename
```

Displaying the type of the file

The `file` command can be used to display the type of the file.



```
[root@linux meerav]# ls
Desktop      dire1  loss  my             mydir       one.sh
Mail         f1     mail  myclass.class  myfile      second
abc          first  mbox  myclass.java   myfiles.gz  second.bak
dead.letter  greet  mess  myclass.java~  newfile     ss
[root@linux meerav]# file loss
loss: English text
[root@linux meerav]# file my
my: fsav (linux) virus (27240-104)
[root@linux meerav]# file one.sh
one.sh: ASCII text
[root@linux meerav]# file second
second: ASCII text
[root@linux meerav]# █
```

Wild card searching

It is possible to perform operations on a set of files without specifying all the names of the files. This can be done by using certain special characters in the command instead of the actual filenames. These special characters are interpreted as a specific pattern of characters. The file names are then compared to see if they match the pattern. The specified operation is performed on files that match the pattern. The following table sumarrises the wild cards and their significance.

Wild Card	Significance
*	Matches any number of characters including none
?	Matches a single character
[ijk]	Matches a single character -- either an i,j ork
[!ijk]	Matches a single character which is not an i,j or k
[x-z]	Matches a single character which is within the ascii range of the characters x and z
[!x-z]	Matches a single character which is not within the ascii range of the characters x and z

```
$ ls tra*
transed
trans
transys
$ _

$ ls *sys
transys
unisys
netsys
$ _

$ ls tra?.?
tran.c
tram.c
$ _
```

The above example displays files having exactly one character after tra followed by a dot(.) and exactly one character after the dot.

For example if we have the files, a1, a2, a3, a4, a5, a6 then the following command lists these files

```
$ ls a[1-6]
```

$ ls a[!2-6]

The above command will list the file a1 since it matches a single character which is not within 2 and 6.

$ ls a[123]

The above command will list files a1,a2,a3 since it matches single character either an 1,2 or 3.

NOTE
The * does not match all files beginning with a dot.Such files must be matched explicitly.

Head/tail commands

The head command is used to display the initial part of a file. The tail command is a complement to the head command and displays the last part of the file. By default, the head command displays first 10 lines of a file and tail command displays the last 10 lines.

The syntax of the head command is

```
$ head [-count] [filename]
```

For example,

```
$ head –4 abc
```

will display the first 4 lines of the file abc.

The syntax of the tail command is

```
$ tail [+/- count] [filename]
```

If we want to display the last 5 lines of the file abc, the following command can be used:

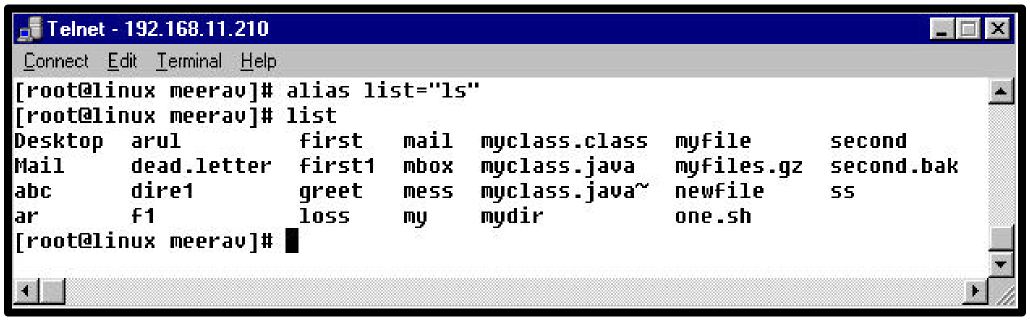
```
$ tail –5 abc
```

If we want to display from the 8th line from the beginning of the file abc, we can write

```
$ tail +8 abc
```

alias and unalias commands

The alias command can be used to give an alias name for any command. For example, if we want to use the word **list** to list out the files and directories instead of the **ls** command, we can do it in the following manner.



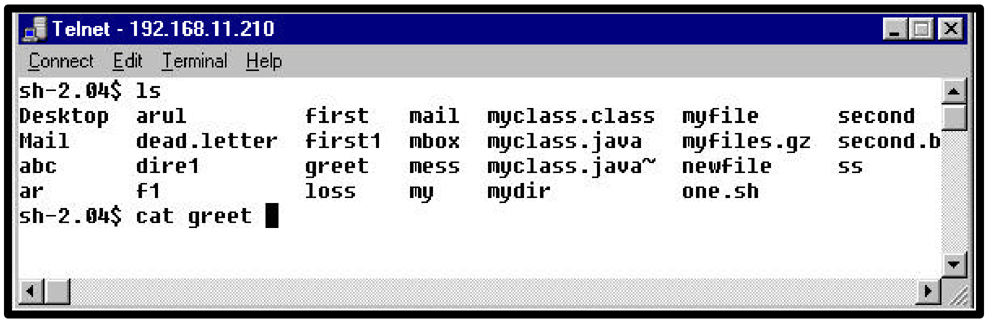
```
[root@linux meerav]# alias list="ls"
[root@linux meerav]# list
Desktop  arul         first   mail  myclass.class   myfile      second
Mail     dead.letter  first1  mbox  myclass.java    myfiles.gz  second.bak
abc      dire1        greet   mess  myclass.java~   newfile     ss
ar       f1           loss    my    mydir           one.sh
[root@linux meerav]# 
```

To remove the alias, we can use the unalias command.

```
$ unalias list
$ list
bash: list: command not found
$_
```

Auto File Completion

The tab key can be used for auto file completion. Typing the first letter of the file and pressing tab will automatically complete the file name for us. Assume we have a file called `greet` and we want to display the contents of the file using the `cat` command. It is not necessary to type the entire file name. If we just type `cat gr` and press `tab`, Linux will complete the file name for us.



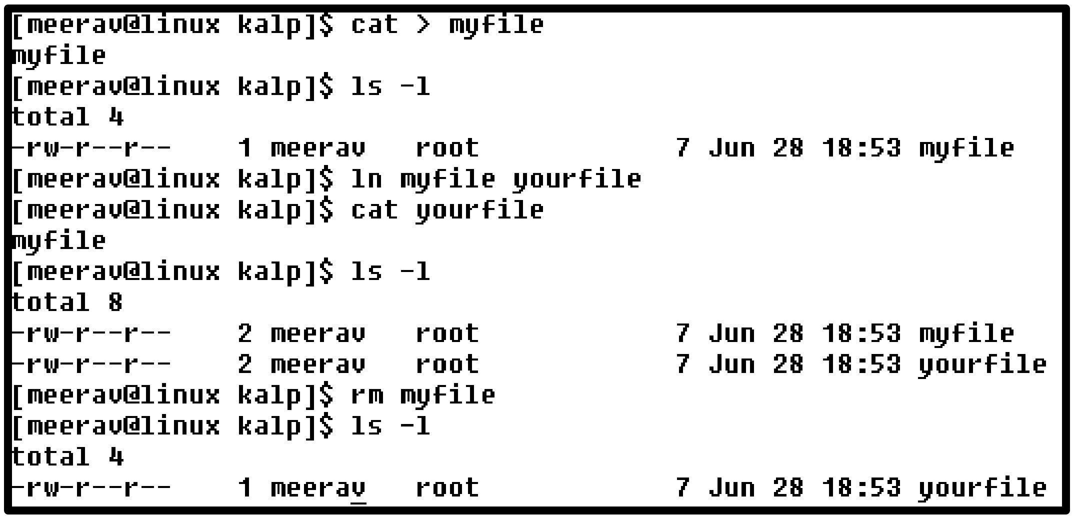
```
sh-2.04$ ls
Desktop  arul         first   mail   myclass.class   myfile      second
Mail     dead.letter  first1  mbox   myclass.java    myfiles.gz  second.b
abc      dire1        greet   mess   myclass.java~   newfile     ss
ar       f1           loss    my     mydir           one.sh
sh-2.04$ cat greet
```

The ln (link) command

A file in the linux system can have several names. The ln command is used to make a link to a file. A link is a directory entry referring to a file. A single file may have several links to it.

There are two kinds of link: hard links and symbolic links.

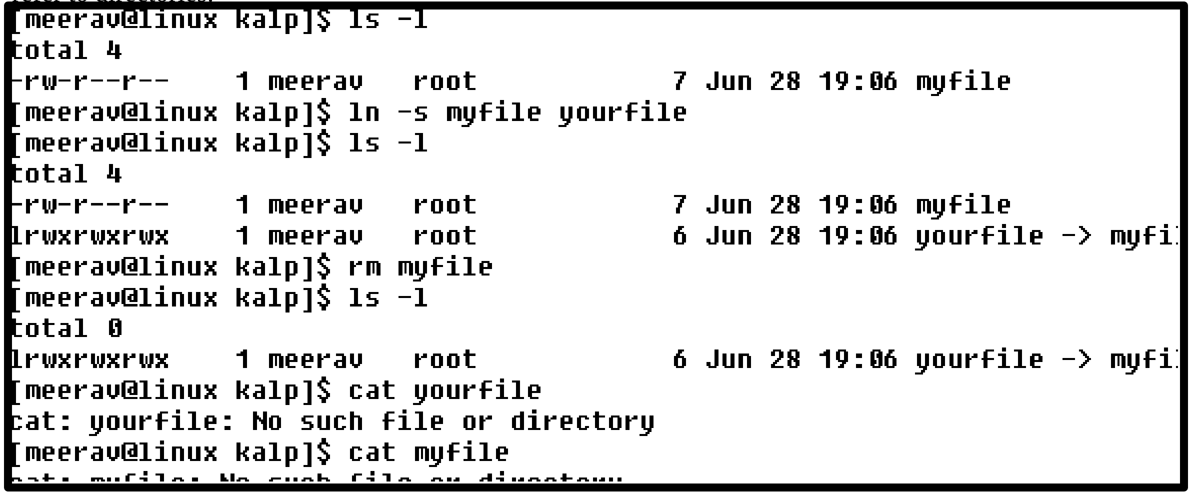
By default ln makes hard links. A hard link to a file enables us to give a file another name. This is done by creating an entry in a directory; no additional disk space is consumed.



```
[meerav@linux kalp]$ cat > myfile
myfile
[meerav@linux kalp]$ ls -l
total 4
-rw-r--r--    1 meerav   root           7 Jun 28 18:53 myfile
[meerav@linux kalp]$ ln myfile yourfile
[meerav@linux kalp]$ cat yourfile
myfile
[meerav@linux kalp]$ ls -l
total 8
-rw-r--r--    2 meerav   root           7 Jun 28 18:53 myfile
-rw-r--r--    2 meerav   root           7 Jun 28 18:53 yourfile
[meerav@linux kalp]$ rm myfile
[meerav@linux kalp]$ ls -l
total 4
-rw-r--r--    1 meerav   root           7 Jun 28 18:53 yourfile
```

Take a look at the above commands where a file called myfile is created with some contents. The file is linked to another file called yourfile. We can try making some changes to myfile it will reflect in yourfile. On deleting myfile, yourfile still exists because it is created through a hard link.

The –s option is used to create symbolic links. A symbolic link contains the name of the file to which it is linked. This file need not exist prior to the symbolic link.Symbolic links may span file systems and may refer to directories.



```
[meerav@linux kalp]$ ls -l
total 4
-rw-r--r--    1 meerav   root            7 Jun 28 19:06 myfile
[meerav@linux kalp]$ ln -s myfile yourfile
[meerav@linux kalp]$ ls -l
total 4
-rw-r--r--    1 meerav   root            7 Jun 28 19:06 myfile
lrwxrwxrwx    1 meerav   root            6 Jun 28 19:06 yourfile -> myfi
[meerav@linux kalp]$ rm myfile
[meerav@linux kalp]$ ls -l
total 0
lrwxrwxrwx    1 meerav   root            6 Jun 28 19:06 yourfile -> myfi
[meerav@linux kalp]$ cat yourfile
cat: yourfile: No such file or directory
[meerav@linux kalp]$ cat myfile
cat: myfile: No such file or directory
```

The above series of commands illustrates the working of symbolic links. Note the size of the files myfile and yourfile. On deleting myfile, yourfile is not accessible.

☞ TOTAL RECALL

1. What is the significance of the head command?
2. What is the command for renaming a directory?
3. Is it possible to delete a directory that is not empty?

REVIEW QUESTIONS

1. The ____________________ command can be used to display the contents of a file.
2. The ________________ command is used to rename a directory.
3. The _____________ command is used to display the last portion of a file.