UNIX PROGRAMMING
BASIC FILE ATTRIBUTES

BASIC FILE ATTRIBUTES

- ls –l command is used to display the entire file attributes in the pwd.It is going to list all seven attributes all fetched from File Inode Table.
- $ls –l
- Total 72
- _rw_r__r__ 1 kumar metal 19514 May 10 13:45 chap01
- _rw_r__r__ 1 kumar metal 4174   May 10 15:01 chap02
- drwxr_xr_x   2 kumar metal 512   May 09 15:08 helpdir
- -d option :Listing Directory Attributes
- To force to list the attributes of a directory ,rather than its contents ,You need to use the –d option.

- $ls -ld helpdir progs
- drwxr_xr_x   2 kumar metal   512 May 9 10:31 helpdir
- drwxr_xr_x   2 kumar metal   512 May 9 09:57 progs.

- r -read permission – octal 4
- W –write permission –octal 2
- x-Execute Permission –octal 1
-

umask - Permission Mask

- Umask specifies what permission bits will be set on a new
- file or directory when created.
- New Directory: 777 – 022 755   rwxr_xr_x
- New File : 666 – 022 644   rw _r_ _ r_ _
- The default value of umask is set in /etc/profile. This can be
- changed for all the users or a particular user.
-

Chmod command changing File Permissions

- A file or directory is created with a default set of permissions and this default is determined by a simple setting [called umask].
- Generally the default setting write-protects a file from all except the user though all users may have read access.
- However this may not be so on your system.
- To know your system’s default create a file xstart
- $cat /usr/bin/startx >xstart
- ls –l xstart

Chmod command

- $cat /usr/bin/startx >xstart
- ls –l xstart
- _rw_r__r__ 1 kumar metal 1906 sep 5 23:28 xstart
- It seems that by default a file does’nt also have execute permissions.To do that permisssions of the file need to be changed.This is done with chmod.
- chmod command is used to set the permissions of one or more files for all three categories of users[user,group and others].
- It can be run only by the user [the owner] and the superuser.

Chmod Operation

- **Syntax:**
- **chmod *mode_list file...*** Change permissions of file(s)
- *mode_list [who[operator]permission] [,...]*
- *who* user, group, other or all(u/g/a)
- *operator* + (add), - (subtract), = (set equal to)
- *permission* read, write, execute
- Example:
- Original permissions: mode user group other
- rw-r--r-- rw- r-- r--
- $ chmod u+x, g+x, o+x file *or $ chmod +x file*
- *Final permissions: mode user group other*
- *rwxr-x r_*x rwx r_x r_x

Chmod Operation

- File and directory permissions can also be specified as an octal number:
- Read Permission :4
- Write Permission :2
- Execute Permission:1
- We can just add the numbers to specify the permission for each category
- Example: 6 means read and write, 7 means read, write and execute.
- eg:
- $ chmod 664 f1 will give read and write permissions for owner and
- group while only read for others

Chmod Operation

- The default protections for newly created files and directories are:
- File -rw-r—r-- 644
- Directory drwxr-xr-x 755 .
- These default settings may be modified by changing the **umask**
- value.

What is Vi ?

- A screen-oriented text editor
- • Included with most UNIX system distributions
- • Command driven
- • Categories of commands include
- – General administration
- – Cursor movement
- – Insert text
- – Delete text
- – Paste text
- – Modify text

What is Vi?

- vi [filename] Start a vi edit session of file
- Example:
- $ vi testfile
- - If the file doesn’t exist, it will be created
- - Otherwise vi will open the existing file
- All modifications are made to the copy of the file brought into
- memory.
- :wq or :x or <shift-zz> write and quit
- :w write
- :q quit
- :q! Quit without saving

Chown command

- Syntax
- **chown *owner [:group] filename*** Changes owner of a file(s) and, optionally, the group ID. Only the owner of a file (or root) can change the ownership of the file.
- Example:
- $ id
- uid=101 (user1), gid=101 (group1)
- $ cp f1   /tmp/user2/f1

- $ ls -l /tmp/user2/f1
- -rw-r----- 1 user1 group1 3967   Jan 24 13:13   f1
- $ chown user2 /tmp/user2/f1

- $ ls –l   /tmp/user2/f1
- -rw-r----- 1 user2 class 3967 Jan 24 13:13 f1

The chgrp Command

- **Syntax:**
- **chgrp *newgroup filename* ...**
- Example:
- $ id
- uid=303 (user3), gid=300 (class)
- $ ls -l f3
- -rw-r----- 1 user3 class 3967 Jan 24 13:13 f3
- $ chgrp class2 f3
- $ ls -l f3
- -rw-r----- 1 user3 class2 3967 Jan 24 13:13 f3
- $ chown user2 f3
- $ ls -l f3
- -rw-r----- 1 user2 class2 3967 Jan 24 13:13 f3

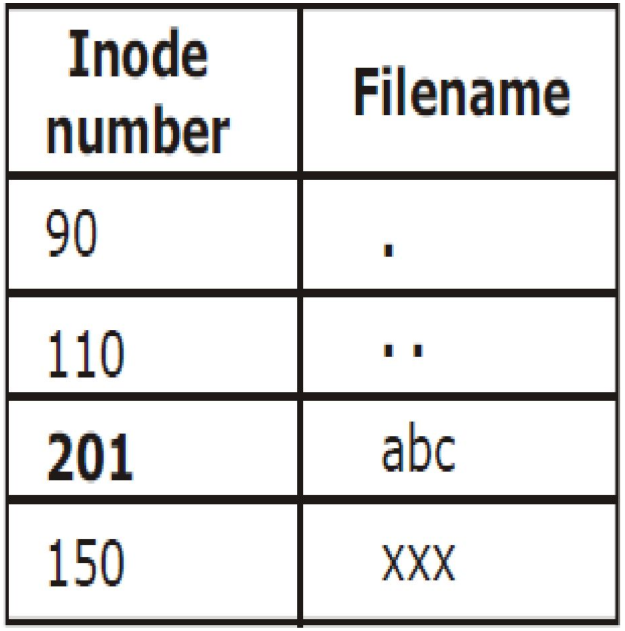
Hard Link and Symbolic Link Files

- .)Hard link is a UNIX pathname for a file. Generally most of the UNIX files will be having only one hardlink.In order to create a hard link, we use the command ln.
- Example :
- Consider a file /usr/ syed/f1, to this we can create a hard link by
- **ln /usr/syed/f1   /usr/syed/f2**
- .
- Symbolic link can be creates by the same command '*ln'*but with option –s
- Example: **ln –s /usr/syed/f1   /usr/syed/sf1**

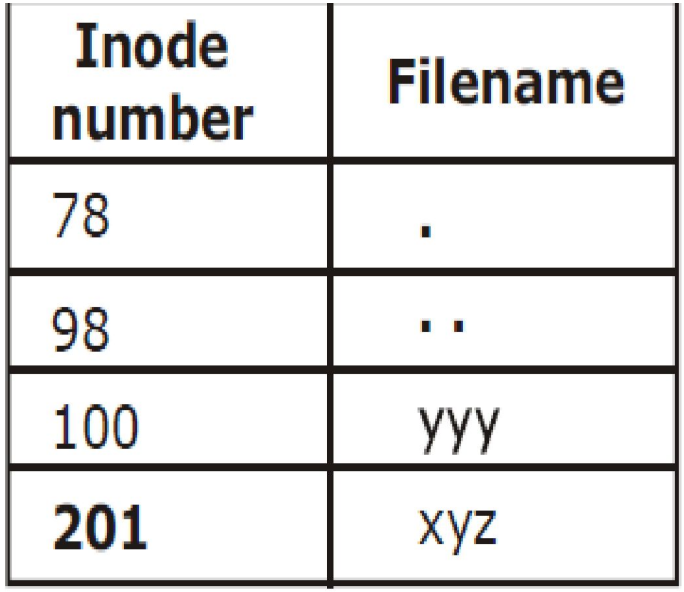
Hard Link and Symbolic Link

- **ln** command differs from the **cp**(copy) command in that **cp** creates a duplicated copy of a file to another file with a different pathname.
- Where as **ln** command creates a newlink to reference a file.
- Let’s visualize the content of a directory file after the execution of command **ln**.
- Case 1 for hard linkfile
- **ln /usr/syed/abc /usr/mustafa/xyz.**
- The content of the directory files/usr/syed and /usr/Mustafa are:

Hard Link and Symbolic Link File



Inode number	Filename
90	.
110	..
201	abc
150	xxx



Inode number	Filename
78	.
98	..
100	yyy
201	xyz

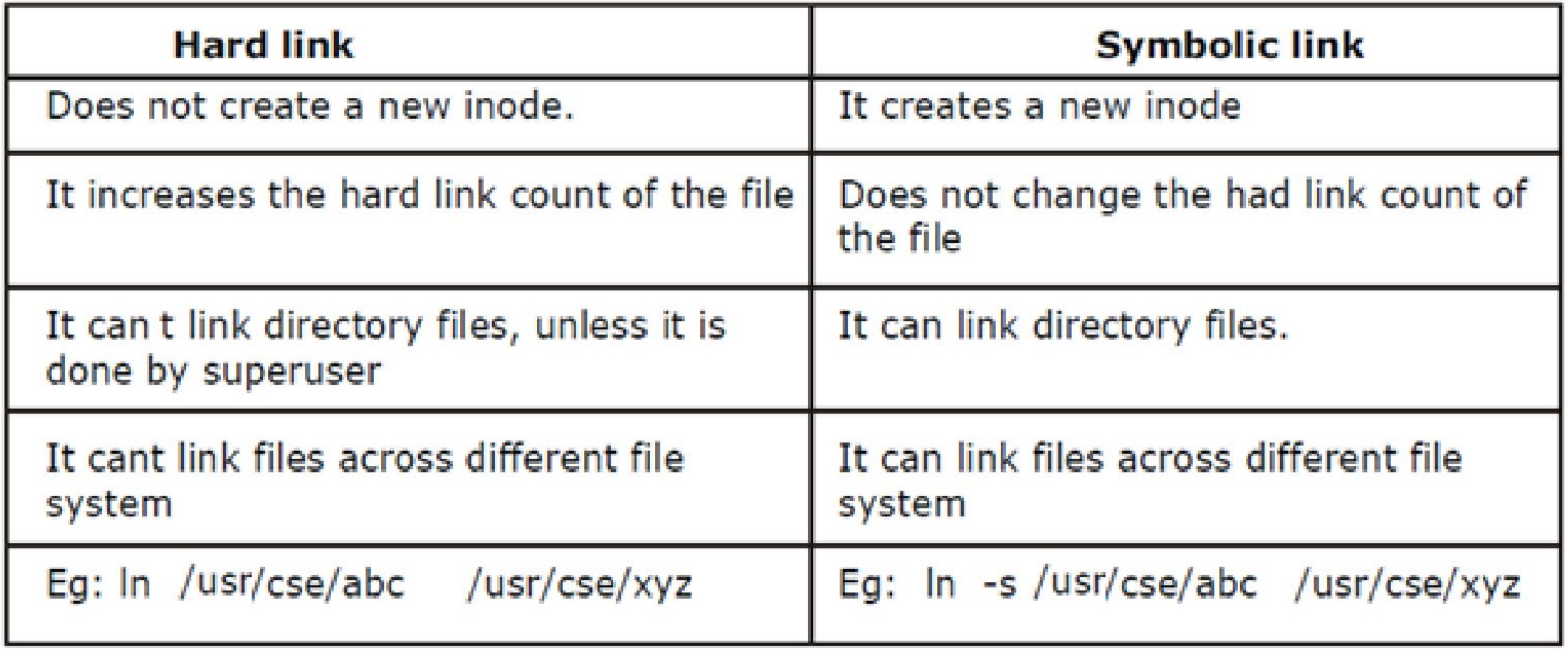
Hard Link and Symbolic Link File

- Both /usrs/syed/abc and /usr/Mustafa/xyz refer to the same Inode number 201 ,thus no new file is created.
- **Case 2:** For the same operation, if ln –s command is used then a new inode will be created.
- **ln –s /usr/syed/abc   /usr/mustafa/xyz**
- The content of the directory files syed and Mustafa will be

Inode Number	Filename
90	.
110	. .
201	Abc
150	xxx

Inode Number	FilenAME
78	.
98	. .
100	yyy
450	xyz

Hard Link and Symbolic Link File

- **If cp command was used then the data contents will be identical and the 2 files will be separate objects in the filesystem, where as in ln – s the data will contain only the pathname.**
- 

Hard link	Symbolic link
Does not create a new inode.	It creates a new inode
It increases the hard link count of the file	Does not change the had link count of the file
It can t link directory files, unless it is done by superuser	It can link directory files.
It cant link files across different file system	It can link files across different file system
Eg: ln /usr/cse/abc /usr/cse/xyz	Eg: ln -s /usr/cse/abc /usr/cse/xyz

Hard and Symbolic Link Files

- **Hard and Symbolic Links**
- **Limitations of hard link:**
- User cannot create hard links for directories, unless he has super-user privileges.
- User cannot create hard link on a file system that references files on a different file system, because inode number is unique to a file system.
- Differences between hard link and symbolic link are listed above:
- A symbolic link file contains a path name which references another file in either local or a remote file system.
- A symbolic link may be created in UNIX via the **ln** command

Link Files

- You can Link multiple files (I,e create a hard link for each) but then the destination filename should be a directory.
- $ln   chap??   Projects_dir
- if chap?? Matches 20 files starting from chap01 to chap20 ,there will be 25 entries in that directory.
- Links provide some protection against accidental deletion,especially when they exist in different directories.
- when you execute ln command with another file
- ln   employee emp.dat;
- ls –li emp*
- 29518   _rwxr_xr__   2   employee
- 29518   _rwxr_xr__x 2   emp.dat

Difference Between ln and rm command

- The role of rm and ln are complimentary,which is evident from the names of their corresponding system calls link and unlink.
- The ln command adds an entry in the directory and also increases the link count in the inode by one.
- The rm command simply reverses the action of ln.
-

Difference Between ln and rm command

DIRECTORY	
Filename	InodeNumber
. . . Date.sh	886444 417585 254414
Inode for date.sh	
Link Count =1	

DIRECTORY	
Filename	InodeNumber
. . . Date.sh Who.sh	886444 417585 254414 254414
Inode for date.sh	
Link Count =2	

DIRECTORY	
Filename	InodeNumber
. . . Date.sh	886444 417585 254414
Inode for date.sh	
Link Count =1	

UNIX FILES

- **Directory Files**

- It is a record-oriented file.
- Each record contains the information of a file residing in that directory
- The record data type is *struct dirent* in UNIX System V and POSIX.1 and *struct direct* in BSD UNIX.
- The record content is implementation-dependent
- They all contain 2 essential member fields:
- 1.File name
- 2.Inodenumber
- Usage is to map file names to corresponding inode number.

find command

- find is one of the very good tools used in Unix System.
- It recursively examines a directory tree to look for file matching some criteria and then takes some action on the selected files.
- However find is easily tamed if you break up its arguments into three components
- find path_list selection_criteria action
- This is how find operates:
- First it recursively examines all files in the directories specified in the path_list.
- It then matters each file for one or more selection criterion.
- Finally it takes some action on those selected Files.

Find Command

- The path_list comprises one or more subdirectories separated by whitespace.There can also be a host of selection criteria ,that you can use to match a file, and multiple actions to dispose of the file.
- find / -name a.out -print
- /home/kumar/scripts/a.out
- /home/tiwary/scripts/reports/a.out
- /home/sharma/a.out

Find Command

- The path list (/) indicates that the search should start from the root directory.Each file in the List is then matched against the selection_criteria (-name a.out) which always consists of an expression in the form –operator argument.
- You can also use relative pathnames (like the .) in the path_list and find will then output a list of relative pathnames. When find is used to match a group of filenames with a wild-card pattern ,the pattern should be quoted to prevent the shell from looking at it.

Find Command

- Eg find . -name “*.c” -print
- find . -name ‘[A-Z]*’ -print
- Selection Criteria
- Locating a File by Inode Number(-inum)
- find allows us to locate files by their inode number .Use the –inum option to find all filenames that has the same inode number.

▪ find / -inum 13975 -print

Note that find throws an error message when it can’t change to a directory.Sometimes there will be so many of them on your screen that you would find it difficult to spot the files that actually show up as find’s output.

Find command

- To avoid these messages simply redirect to the std error /dev/null/
- $find . -type d -print 2 >/dev/null
- Note the relative pathname find displays ,but that because the pathname itself was relative. (.).
- The -perm option specifies the permission to match.For instance –perm 666 selects files having read and write permission for all categories of users.
- You’ll often want to use two options in combination to restrict the search to only directories.
- find $HOME –perm 777 -type d -print
- find uses an AND condition(an implied –a operator b/wn –perm and –type)

to select directories that provide all access rights to every one.

Find Operators(!, -o and –a)

- There are three operators that are commonly used with find.The ! Operator is used before an option to negate its meaning.
- $find . ! –name “*.c” -print
- The meaning of the above command suggests that It selects all files except the c-program files.To look for both shell and perl scripts use the –o operator which represents an OR condition. We need to use an escape pair of parenthesis here.
- $ find /home \(-name “*.sh” –o -name “*.pl” \) -print
- The (and) are special characters that are Interpreted by the shell to group commands.
-

Find Operators

- Major Expressions used by find.

Sele ction criteria	Selects File
-inum n	Having Inode number n
-type x	if of type x:X can be file(ordinary)
	d(directory) or l(symbolic link file).
- perm nnn	Octal permission match nnn complete.
-user Username	if owned by username
-group gname	if owned by group groupname.