UNIX PROGRAMMING (JNTUK-R16)

UNIT-2

SYLLABUS:

i) The File System

- The Basics of Files
- What's in a File
- Directories and File Names
- Permissions
- Inodes
- The Directory Hierarchy

ii) File Attributes and Permissions

- The File Command-knowing the File Type
- The Chmod Command-Changing File Permissions
- The Chown Command-Changing the Owner of a File
- The Chgrp Command-Changing the Group of a File.

Text Books:

1. The Unix programming Environment by Brain W. Kernighan & Rob Pike, Pearson.
2. Introduction to Unix Shell Programming by M.G.Venkateshmurthy, Pearson.
3. UNIX and Shell Programming, Forouzan and Gilberg, Cengage Learning

1. THE BASICS OF FILES:

- In UNIX System, Everything is represented in the form of a file.
- A file is a sequence of bytes.

File Names:

- Files are constructed and are used by names.

Rules for Constructing File Names:

- The length of a file in UNIX can be up to 256 characters.
- Some file systems have the length of a file name is maximum of 14 characters.
- The file names in UNIX are Case-Sensitive, Upper-case and Lower-case are not same.
- All ASCII characters are allowed to represent a file name.
- The recommended characters of a file are,
    - Alphanumeric characters (combination of letters and numeric digits).
    - The period (.), the hyphen (-) and the underscore (_).
- Hidden file name starts with a dot.

Categories of Files:

- Files in UNIX are 3 types.

  i) Regular Files
  ii) Device Files or Special Files
  iii) Directory Files

i) Regular Files:

- It is a randomly addressable sequence of bytes.
- Also called as ordinary files.
- These files are created, changed or deleted by the user whenever he or she needs.

2

ii) Device Files:

- A device file is a point of interface to one of the computer's hardware device.
- It has 2 types.
  - Character Special Files
  - Block Special Files
- Character Special files are related I/O and used to model serial I/O devices like Terminals, Printers and Networks.
- These files process one character at a time.
- Block Special files are used to model devices like disk drives and magnetic tapes

iii) Directory Files:

- UNIX has a large number of files, so they need to be organized.
- UNIX uses directories to organize these files.
- Directories are known as folders under the windows environment.
- The directories point to some other directories called sub directories.
- The UNIX file system is organized as directories, where each directory can contain subdirectories and/or files.

Directory Types:

- There are 4 types of directories available in UNIX.

  a) Root Directory (/)
  b) Home Directory
  c) Working Directory
  d) Parent Directory

a) Root Directory (/):

- The Root directory is the highest level in the hierarchy
- It is the root of the whole file structure.
- In UNIX environment, the root directory always has several levels of sub directories
- The root directory belongs to the system administrator and can be changed by only the system administrator.

b) Home Directory:

- We use the home directory when we first log into the system.
- It contains any files we create.
- Our home directory is also the beginning of our personal directory structure.
- Each user has a home directory.
- The name of the home directory is the user login id or the user id.

Example:

- If your login id is *student501*, then your home directory is also *student501*.

c) Working Directory:

- The working or current directory is the one that we are in at any point in a session.
- When we are not changing from our home directory, then our working directory is our home directory.
- When we change directory, our working directory changes automatically.

d) Parent Directory:

- The parent directory is immediately above the working directory.

- When we are in our home directory, its parent is one of the system directories.
- When we move from our home directory to a subdirectory, our home directory becomes the parent directory.

2. PATH NAMES:

- Path name Specifies where a file is located in the hierarchically organized file system.
- It is necessary to know how to use path names to navigate the UNIX file system.
- There are two types of path names available in UNIX.

  a) Absolute path name

  b) Relative path name

a) Absolute path name:

- It tells how to reach a file beginning from the root.
- It always begins with slash (/).

Example:

/home/501/sample

b) Relative path name:

- It tells how to reach a file from the directory you are currently in.
- It never begins with a slash (/).

Example:

501/sample

3. OPERATIONS OR COMMANDS UNIQUE TO DIRECTORIES:

a) List Directory (ls)
b) Make Directory (mkdir)
c) Change Directory (cd)
d) Remove Directory (rmdir)
e) Present Working Directory or Locate Directory (pwd)

a) List Directory (ls):

- This command lists the contents in a directory.

Syntax:

```
$ls
```

Example:

```
$ls
```

- The different *options* used in ls command are,

-l	long list
-d	Working Directory
-n	User/ Group id's
-r	Reverse order
-i	Inodes of a file

b) Make Directory (mkdir):

- The mkdir command is used to create new directory.

Syntax:

```
$mkdir  directory-name
```

Example:

```
$mkdir  user1
```

c) Change Directory (cd):

- The cd command is used to change the current working directory.

Syntax:

```
$cd  directory-name
```

Example:

```
$cd  user1
```

d) Remove Directory (rmdir):

- This command is used to delete a directory

Syntax:

```
$rmdir  directory-name
```

Example:

```
$rmdir  user1
```

e) Locate Directory (pwd):

- This command is used to determine the location of the current directory in the directory structure

Syntax:

```
$pwd
```

Example:

```
$pwd
/home/501
```

- It has no options and attributes

4. OPERATIONS OR COMMANDS UNIQUE TO FILES:

a) Create and Edit File (vi)
b) Display File (more)
c) Print File (lpr)

a) Create and Edit File (vi):

- The most common tool to create a file in UNIX system is text editor such as vi
- Other utilities are cat and ed
- UNIX provides several utilities to edit text files
- The most common is a basic text editor such as vi
- Other basic text editor is sed
- All of the basic edit utilities can create a file, but only some can edit one

b) Display File (more):

- This command is used to display a file.
- It allows us to set the output page size and pauses at the end of each page to allow us to read the file.

Syntax:

```
$more filename
```

Example:

```
$more story1
```

- Display options for the more command

-c	Clears screen before displaying
-d	Displays Error messages
-f	Does not screen wrap long lines
-s	Squeezes multiple blank lines (leaving only one blank line in output)

c) Print File (lpr):

- lpr means line printer
- This command is used to print a file

Syntax:

```
$lpr  filename
```

Example:

```
$lpr  story1
```

5. OPERATIONS OR COMMANDS COMMON TO BOTH FILES AND DIRECTORIES:

a) Copy (cp)
b) Move (mv)
c) Rename (mv)
d) Link (ln)
e) Remove (rm)
f) Find (find)

a) Copy (cp):

- This command is used to create a duplicate of a file, a set of files or a directory
- The cp command copies both text and binary files

Syntax:

```
$cp  source-file/directory  destination-file/directory
```

Examples:

```
i) $cp  file1  file2
ii) $cp  dir1/file1  dir1/file2
iii) $cp  dir1/file1  dir2
iv) $cp  file1  dir2/file2
```

b) Move (mv):

- This command is used to move either an individual file, a list of files or a directory

Syntax:

```
$mv  source-file/directory  destination-file/directory
```

Example:

```
$mv  dir1/file1  dir2
```

c) Rename (mv):

- UNIX does not have a specified rename command
- The mv command is used to rename a file or directory

Syntax:

```
$mv  old-file-name  new-file-name
```

Example:

```
$mv  file1  file7
```

d) Link (ln):

- The ln command is a standard Unix command utility used to create a hard link or a symbolic link (symlink) to an existing file.

Example:

```
$ln  file1  lndir
```

e) Remove (rm):

This command is used remove a file or a directory.

Syntax:

```
$rm  filename
```

Example:

```
$rm  file1
```

f) Find (find):

- In a large file environment, it is difficult to find a given file
- It is like a file search option in any OS environment

Syntax:

```
$find filename
```

Example:

```
$find file23
```

6. INODES:

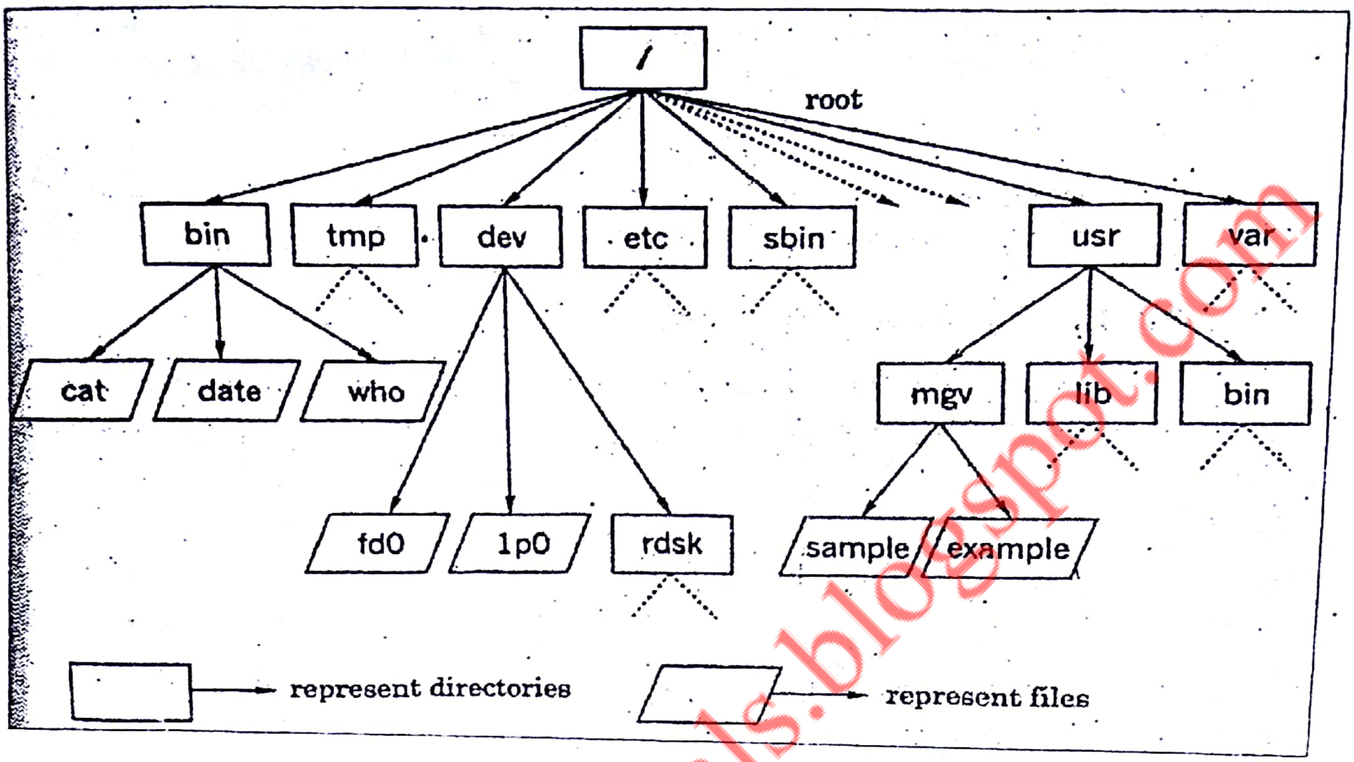
- A number points to an Inode.
- An Inode is a data structure that stores the following information about a file :
  - Size of file
  - Device ID
  - User ID of the file
  - Group ID of the file
  - File protection flags
  - The timestamps for file creation, modification etc
  - link counter to determine the number of hard links

- ls -i command is used to represent inode in the list
- Here i represents inode

Example:

```
$ ls -i
1441807 Desktop
1447344 mydata
1441813 Pictures
1441808 Downloads
1441810 Public
1447099 Unsaved Document 1
```

7. THE DIRECTORY HIERARCHY:



- It is also called as UNIX File System, because it is used to organize UNIX files.
- The different directories in the Directory Hierarchy are,

Directory	Meaning
root (/)	Root of the file system
/bin	Essential programs in executable form (binaries)
/tmp	Temporary files; cleaned when system is restarted
/dev	Device files
/etc	System miscellany
/sbin	System binaries
/usr	User file system
/var	Variables
/usr/lib	Libraries for C, FORTRAN. Etc
/usr/bin	User binaries

8. FILE ATTRIBUTES AND PERMISSIONS:

i) Ownership of a file:

- The person who actually creates a file is owner, also called as user
- The owner has the ability to modify permissions of the file of their own, group or others
- The Supervisor or the System Administrator also have this power
- User names (User Identifications - UIDs) are available in the /etc/passwd file
- Group names (Group Identifications - GIDs) are available in /etc/group file

ii) File Attributes:

- The different File Attributes are,
    - File name
    - Creation, Modification and Access times
    - File size
    - Owner of the file
    - Group to which the owner belongs to
    - Link information
    - Permissions
    - Inode number associated with it

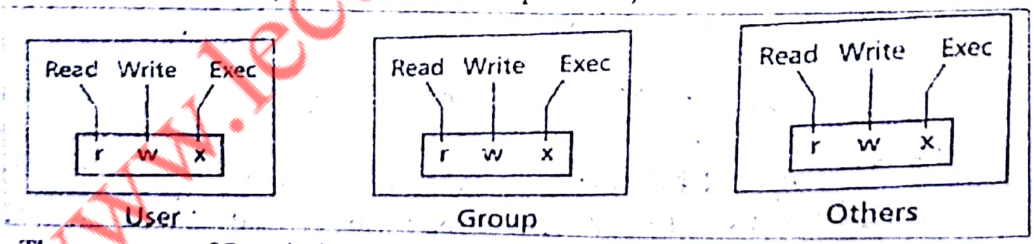
iii) File Permissions:

- Without permission, No one is allowed to access and use a file
- A file may be accessed for one or more of the following purposes
    - Reading
    - Writing
    - Executing
- These permissions may differ depending on the category of users
    - Owner/ User
    - Group
    - Others
- By default,
    - The Owner will have read and write permissions
    - Group and Others have only read permission

Example:

- rw - r - - r - -

- The File and Directory Permissions are depicted as,



- The summary of Permission Rules is depicted in a table as,

Permission	read (r)	write (w)	execute (x)
Directory	Read contents of directory	Add or delete entries (files) in directory using commands	Reference or move to directory
File Level	Read or copy files in directory	Change or delete files	Run executable files

9. THE FILE PERMISSION COMMANDS:

a) The file command (Knowing the File Type):

- The file command is used to identify the type of the files on the basis of their contents.
- When this command is used, it reads either the header or first few hundreds of bytes of the file and an educated guess is made on the type of the file.

Example:

```
i) $file  mgv
    mgv: ASCII text
ii) $file  /bin
     /bin: directory
iii) $file  sum.c
      sum.c: ASCII C program text
```

b) The chmod command (Changing File Permissions):

- The chmod command is used to change permissions of a file after its creation
- Only the Owner or Super User can change file permissions

Syntax:

```
$chmod  assignment-expression  filename
```

- The assignment expression holds the following information
    - The information about category of users { user - u, group - g, others - o, all - a }
    - The information about grant or deny of the permission { operators +, - and = }
    - The information about the type of permission { read - r, write - w, execute - x }

Example1: (giving execute permission to the user)

```
$chmod  u+x  sample
$ls  -l  sample
- r w x r - - r - -   sample
```

Example2: (giving execute permission to all)

```
$chmod  ugo+x  sample
$ls  -l  sample
- r w x r - x r - x
(OR)
$chmod  a+x  sample
$ls  -l  sample
(OR)
$chmod  +x  sample
$ls  -l  sample
```

Example3: (work on more than one file)

```
$chmod  u+x  sample1  sample2  sample3
$ls  -l  sample1  sample2  sample3
-rwxr--r--    sample1
-rwxr--r--    sample2
-rwxr--r--    sample3
```

Example4: (more than one permission)

```
$chmod  u-x, go+x  sample
$ls  -l  sample
-rw-r-xr–x   sample
```

Example5: (Relative and absolute permissions assignment)

```
$chmod  a=r  sample
$ls  -l  sample
-r--r--r--    sample
```

Example6: (permissions with octal numbers)

- File Permissions can also be assigned using octal numbers
  - Octal 4 (100) equal to read permission
  - Octal 2 (010) equal to write permission
  - Octal 1 (001) equal to execute permission

```
$chmod  644  sample
$ls  -l  sample
-rw-r--r--    sample
```

Example7:

```
$chmod  761  sample
```

is equal to

```
$chmod  u=rwx, g=rw, o=x   sample
```

c) The chown command (Changing the owner of a file):

- Every file has a Owner.
- Only the owner can change the major attributes of a file
- Sometimes it is necessary to change the ownership of a file.

Example:

```
$ls -l sample
-rwxr--r-x 1 dhoni July 19 11:55   sample
$chown virat sample
$ls -l sample
-rwxr--r-x 1 virat July 19 11:55   sample
```

d) The chgrp command (Changing the group of a file):

- In UNIX, all files are not belong to an owner but also to a group
- Sometimes changing the Group of a file is needed
- The owner or the System Administrator can change the group of a file
- This is possible by using the chgrp command

Example:

```
$chgrp innovative sample
```

- Using -R option with this command, the *group of all files under the current directory* is changed

Example:

```
$chgrp -R innovative
```

- Here, all directories and files in the current directory are changed to innovative group.

IMPORTANT QUESTIONS:

1. a) What is a file? Write its types.
   b) What are the rules for constructing File Names?
   c) What is a directory? Write its types.
2. Explain about File Permission Commands

   OR

   Explain the following file commands
   a) file b) chmod c) chown d) chgrp