Capgemini

UNIX

# UNIX File System

# Lesson Objectives

In this lesson, you will learn:

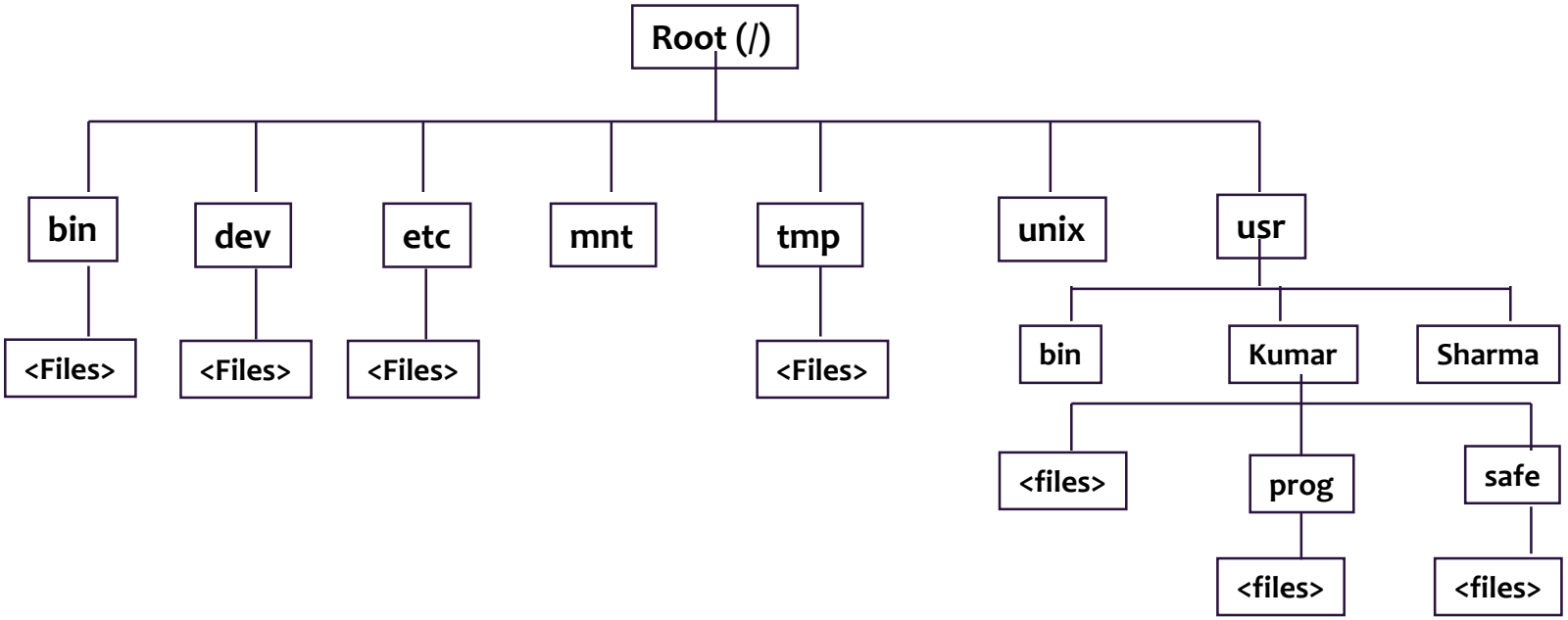
- UNIX File system
- File types
- File permissions
- Commands related to file permission
  - mkdir, cd, cat etc…

# Overview

Let us discuss a File System with respect to the following:

- Hierarchical Structure
- Consistent Treatment of Data: Lack of file format
- The Treatment of Peripheral Devices as Files
- Protection of File Data

# File System Structure



# File System Structure

/ bin : commonly used UNIX Commands like who, ls

/usr/bin : cat, wc etc. are stored here

/dev : contains device files of all hardware devices

/etc : contains those utilities mostly used by system administrator

- Example: passwd, chmod, chown

# File System

/tmp : used by some UNIX utilities especially vi and by user to store temporary files

/usr : contains all the files created by user, including login directory

/unix : kernel

Release V:

- It does not contain / bin.
- It contains / home instead of /usr.

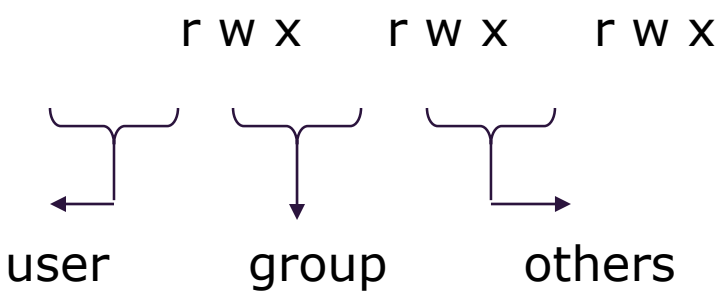
# File Types in UNIX

We have the following file types in UNIX:

- Regular File
- Directory File
- Device File

# File Permissions in UNIX

## File Access Permissions



# File Permissions in UNIX

Permissions are associated with every file, and are useful for security.

There are three categories of users:

- Owner (u)
- Group (g)
- Others (o)

There are three types of “access permissions”:

- Read (r)
- Write (w)
- Execute (e)

# pwd Command

The pwd command checks current directory.

```
$ pwd
```

- **Output:** /usr/Kumar

# cd Command

The cd command changes directories to specified directory

The directory name can be specified by using absolute path (Full Path) or relative path

```
$ pwd
```

- **Output:** /usr/kumar

```
$ cd Prog
$ pwd
```

- **Output:** /usr/kumar/Prog

# cd Command

Moving one level up:

```
$ cd ..
```

Switching to home directory:

```
$ cd
```

Sw

```
$ cd /usr/Sharma
```

Sw

```
$ cd /
```

# logname Command

The logname command checks the login directory.

```
$ logname
```

**Output:** Kumar

# ls Command

The ls command lists the directory contents.

Example:

```
$ ls
```

**Output:**

```
a.out
chap1
chap2
test
test.c
```

# ls Command

Options available in ls command:

| Option | Description |
|---|---|
| **-x** | **Displays multi columnar output (prior to Release 4)** |
| **-F** | **Marks executables with *and directories with /** |
| **-r** | **Sorts files in reverse order (ASCII collating sequence by default)** |
| **-l** | **The long listing showing seven attributes of a file** |
| **-d** | **Forces listing of a directory** |
| **-a** | **Shows all files including ., .. And those beginning with a dot** |

# ls Command

Options available in ls command:

| Option | Description |
|---|---|
| **-t** | **Sorts files by modification time** |
| **-R** | **Recursive listing of all files in sub-directories** |
| **-u** | **Sorts files by access time (when used with the –t option)** |
| **-i** | **Shows i-node number of a file** |
| **-s** | **Displays number of blocks used by a file** |

# ls Command

Example:

```
$ ls – l
```

- It displays output as follows which includes 7 columns total 8:

```
-rw-rw-rw-        1 Kumar group 44   May 9 09:08 dept.h
-rw-rw-rw-        1 Kumar group 212  May 9 09:08 dept.q
-rw-rw-rw-        1 Kumar group 154  May 9 09:08 emp.h
```

# ls Command

Consider the first column:

Field1 --> mode

- r w x r w x r w x

□ --> user permissions

□ --> group permissions

□ --> others permissions

# ls Command

File type

- 1 st character represents file type:

  - r w x  r w x  r w x

  - -  --> regular file
  - d --> directory file
  - c --> character - read
  - b --> block read

# ls Command

Field2 : indicates number of links

Field3 : File owner id

Field4 : Group id

Field5 : File size in bytes

Field6 : Date/time last altered

Field7 : Filename

# cat Command

The cat command is used for displaying and creating files.

- To display file:

```
$ cat dept.lst
```

```
01|accounts|6213
02|admin|5423
:
06|training|1006
```

- To create a file:

```
$cat > myfile
```

  - This is a new file
  - Press ctrl-d to save the contents in  file myfile

# cat Command

The cat command can be used to display contents of more than one file.

- It displays contents of chap2 immediately after displaying chap1.

```
$ cat chap1 chap2
```

# Input and Output Redirection

Standard Input : Keyboard

Standard Output: Monitor

Standard Error : Monitor

Redirection operators:

- < : Input Redirection
- > : Output Redirection
- 2> : Error Redirection
- >> : Append Redirection

# Redirection

Input redirection: Instead of accepting i/p from standard i/p(keyboard) we can change it to file.

- **Example:** $cat < myfile  will work same as  $cat myfile
- < indicates, take i/p form myfile and display o/p on standard o/p device.

Output redirection: To redirect o/p to some file use >

- **Example:** $cat < myfile > newfile
- The above command will take i/p from myfile and redirect o/p to new file instead of standard o/p (monitor).

# Redirection

- $ cat < file1.txt > result  is same as $cat file1.txt > result.

```
$ cat result
```

**Output:** 2  12  60

- >> is append redirection
- The given command will append the contents of file1.lst in result file.

```
$ cat  < file1.lst >> result
$ cat result
```

**Output:** 2  12  60
4  4  8

# cat file exist/not exist

Consider an example of cat –(file exist/not exist):

```
$ cat abc.txt > pqr.txt 2> errfile.txt
```

- If file abc.txt exists:
  - Then contents of the file will be sent to pqr.txt. Since no error has occurred nothing will be transferred to errfile.txt.
- If abc.txt file does not exist:
  - Then the error message will be transferred to errfile.txt and pqr.txt will remain empty.

# cp Command (copy file)

The cp (copy file) command copies a file or group of files.

The following example copies file chap1 as chap2 in test directory.

- Example:

```
$ cp chap1 temp/chap2
Option - i (interactive)
        $cp  - i  chap1     chap2
        cp: overwrite chap2   ? y
Option -r (recursive) to copy entire directory
$cp  -  r   temp   newtemp
```

# rm Command (delete file)

The rm (remove file) command is used to delete files:

```
$ rm chap1        chap2        chap3
        $ rm   *
          Are you sure? y
Option - i (interactive delete)
        $ rm  -  i     chap1        chap2
        chap1 : ? Y
        chap2  :?  Y
Option   -   r (recursive delete)    (Avoid using this option)
```

# mv Command

The mv command is used to rename file or group of files as well as directories.

```
$ mv   chap1  man1
```

The destination file, if existing, gets overwritten:

- Example: $ mv temp doc
- Example: $ mv chap1 chap2 chap3 man1
  - It will move chap1, chap2 & chap3 to man1 directory

# wc Command

The wc command counts lines, words, and character depending on option.

It takes one or more filename as arguments.

no filename is given or - will accept data from standard i/p.

```
$ wc infile
3  20  103 infile
$wc       or    $wc  -
This is standard input
press ctrl-d to stop
```

- **Output:** 2  8  44

# wc Command

```
$ wc  infile test
```

**Output:**

| | | | |
|---|---|---|---|
| 3 | 20 | 103 | infile |
| 10 | 100 | 180 | test |
| 13 | 120 | 283 | total |

```
$ wc  -  l  infile
```

**Output:** 3 infile

```
$ wc  -  wl  infile
```

**Output:** 20 3 infile

The following command will take i/p from infile and send o/p to result file

```
$ wc < infile > result
$ cat result
```

**Output:** 2 12 60

# cmp Command

cmp Command:

```
$ cmp file1.txt file2.txt
file1.txt file2.txt differ: char 41, line 2
$ cmp file1.txt file1.txt
```

# comm Command

comm Command:

- The comm command compares two sorted files. It gives a 3 columnar output:
  - First column contains lines unique to the first file.
  - Second column contains lines unique to the second file.
  - Third column displays the common lines.

# comm Command

```
$ cat cfile1.lst
A
G
K
X
```

```
$ cat cfile2.lst
A
F
K
W
X
Z
```

```
$ comm cfile1.lst cfile2.lst
                A
        F
G
                K
        W
                 X
        Z
```

```
$ comm -12 cfile1.lst cfile2.lst
A
K
X
```

# diff Command

The diff command is used to display the file differences. It tells the lines of one file that need to be changed to make the two files identical.

- Example:

```
$ diff cfile1.lst cfile2.lst
2c2
< G
> F
3a4
> W
4a6
> Z
```

# tr Command

The tr command accepts i/p from standard input.

This command takes two arguments which specify two character sets.

The first character set is replaced by the equivalent member in the second character set.

The –s option is used to squeeze several occurrences of a character to one character.

# tr Command

Example 1: To squeeze number of spaces by single space:

```
$ tr –s “ ”  < file1.txt
```

Example 2: To convert small case into capital case:

```
$ tr “[a-z]” “[A-Z]” < file1.txt
ONE
TWO
THREE
FOUR
```

# more Command

The more command, from the University of California, Berkeley, is a paging tool.

The more command is used to view one page at a time. It is particularly useful for viewing large files.

Syntax for more command is as follows:

```
more <options> <+linenumber> <+/pattern> <filename(s)>
```

Example: To display file1.txt one screenful at a time

```
$ more file1.txt
```

# chmod Command (Alter File Permissions)

The chmod command is used to alter file permissions:

Syntax:

```
chmod <category> <operation> <permission> <filenames>
```

| Category | Operations | Attribute |
|---|---|---|
| u-user | +assigns permission | r-read |
| g-group | -remove permission | w-write |
| o-others | =assigns absolute permission | x-execute |
| a-all | | |

# chmod Command (Alter File Permissions)

Example 1:

```
$ chmod u+x note
$ ls - l note
      -rwx r-- r --1 ……     note
```

Example 2:

```
$ chmod ugo+x          note
$ ls - l note
  -rwxr-xr-x ……       note
```

- When we use + symbol, the previous permissions will be retained and new permissions will be added.
- When we use = symbol, previous permissions will be overwritten.

# chmod Command (Alter File Permissions)

Example 3:

```
$ chmod u-x, go+r        note
      $ chmod u+x        note      note1      note2
      $ chmod o+wx       note
      $ chmod ugo=r      note
```

# chmod Command (Alter File Permissions)

Octal notation:

- It describes both category and permission.
- It is similar to = operator (absolute assignment).
  - read permission: assigned value is 4
  - write permission: assigned value is 2
  - execute permission: assigned value is 1
- Example 1:

```
$ chmod  666 note
```

  - It will assign read and write permission to all.

# chmod Command (Alter File Permissions)

- Example 2:

```
$ chmod  777 note
```

  - It will assign all permissions to all.

- Example 3:

```
$ chmod  753 note
```

# mkdir Command

The mkdir command creates a directory.

- Example: 1:

```
$ mkdir doc
```

- Example 2:

```
$ mkdir doc    doc/example        doc/data
```

- Example 3:

```
$ mkdir        doc/example        doc
```

  - It will give error - Order important.

# rmdir Command

The rmdir command is used to remove directory.

Only empty dir can be deleted.

More than one dir can be deleted in a single command.

Command should be executed from at least one level above in the hierarchy.

# rmdir Command

Example 1:

```
$ rmdir doc
```

Example 2:

```
$ rmdir doc/example doc
```

Example 3:

```
$ rmdir doc  doc/example
```

- It will give error.

# Internal and External Commands:

External commands

- A new process will be set up
- The file for external command should be available in BIN directory
- E.g – cat, ls , Shell scripts

Internal commands

- shell’s own built in statements, and commands
- No process is set up for such commands.
- E.g cd , echo

# Summary

In this lesson, you have learnt:

- UNIX organizes files in hierarchical manner.
- File access can be secured using different file permissions.
- < - Input Redirection
- > - Output Redirection
- 2> - Error Redirection
- chmod command is used to change file permissions.

# Review Questions

Question 1: To copy all files with extension txt to mydir directory ___ command is used, if mydir is parent directory of current directory.

- Option 1: cp *.txt ..
- Option 2: cp *.txt ../mydir
- Option 3: cp mydir *.txt


Question 2: 2> symbol is used as error redirection

- True / False

Question 3: cd . changes the directory to ___.

Question 4: Which of the following command will give only read permission to all for file file1.txt?

- Option 1: chmod a=r file1.txt
- Option 2: chmod a+r file1.txt
- Option 3: Chmod 666 file1.txt

# Review – Match the Following

| | |
|---|---|
| 1. To change directory to home directory | a. rm *.dat |
| 2. To remove all files with extension *.dat | b. cat <abc.txt |
| 3. To display contents of file abc.txt | c. cat > abc.txt |
| 4. To create abc.txt file | d. cd |
| | e. cd \ |
| | f. mkdir mydir |