Unit - I

classmate
Date
Page

## Brief History of UNIX:

UNIX is an operating system.

Origin - Invented at AT & T Bell research lab by Dennis Ritchie.

Initially it was open source but later it was commercialized.

Interaction - To interact with UNIX user has to use the "commands."

Medium : A command line interpreter, called "shell".

↓

Translates a command into a particular action.

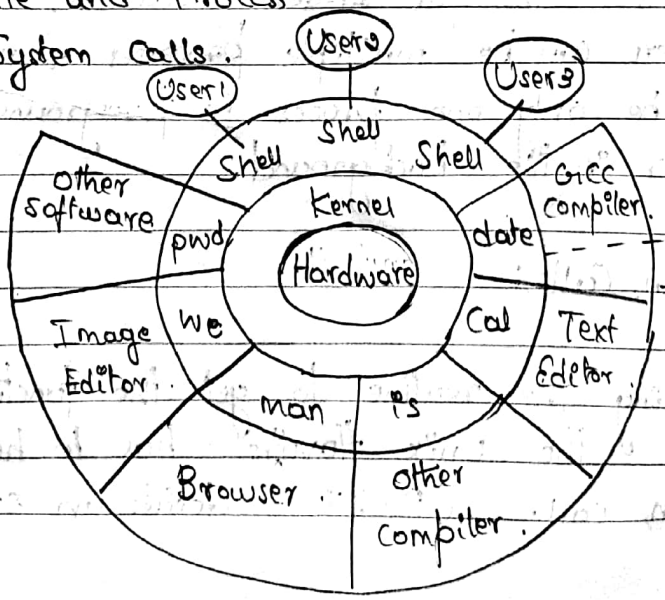
## Linux and Unix:

| Linux | Unix |
|---|---|
| Free source (Linus Torvaldes - Rewritten all features using c). | Commercial source. |
| Initially called GNU. | |

## Unix Architecture

Unix architecture is based on three fundamental concepts.

1. Division of labor : Kernel and Shell.
2. File and Process
3. System calls.




Scanned by CamScanner

## 1. Division of Labor : Kernel and Shell.

- "Kernel" is the core of unix operating system that can interact with the hardware of the system directly to offer the services.
- Kernel has only the routines that are written is 'c' language, and these routines are meant to offer one or more services.
- To get the services from the hardware, user will make use of c routines through the "system calls".
- Apart from offering user services it also controls all the activities of the system.

- "Shell" is a command line interpreter that acts as an interface between user and kernel. ($date)
- One kernel can have many shells.

## 2. File and Process :

"File" - Anything that exists under unix os is treated as file.
- It contains only an array of bytes that represent some kind of information

"Process" - A file under execution.
- There can be multiple processes running. But there will be only one process in foreground rest all appear in the background.

## 3. System Calls :

- Calling the routine to get the function executed.
- All the unix flavours has to have a common system call. (POSIX - decide on standards).

Scanned by CamScanner

## Salient features of Unix OS:

1. Multitasking system
2. Multiuser system
3. Building block approach
4. Unix Tool kit
5. Pattern Matching
6. Documentation
7. Programming facility

### 1. Multitasking system:

A single user can accomplish multiple jobs at a time.

But single job will be in foreground: all other jobs will be in background.

### 2. Multi user system

Multiple users can perform multiple jobs on a single system.

### 3. Building block approach

Multiple commands can be combined together to perform a task.

It facilitates the usage of '|' (pipe) to combine task of multiple commands.

Eg: command1 | command2

output of command1 is fetched as input by command2.

### 4. Unix tool kit:

All the flavours of unix operating system come with the essential utility tools that is required to work on the system.

Scanned by CamScanner

5. Pattern matching

Shell has some special characters that are used to match the pattern for command execution

Eg: * (metacharacters)

If chap, chap1, chap2 are files in current directory.

$ ls chap* displays all files chap, chap1, chap2.

Special characters used to match patterns are called as meta characters.

6. Documentation

Unix operating system comes with a manual page for all the commands it has.

'man' is the command that gives documentation part.

7. Programming facility:

Shell itself is a programming language. Like any other languages it provides all the features a programming language is expected to have.

Command structure

$ command_name [optionlist] [arguementlist]

(optionlist start with '-')

Examples:

1. echo - displays the message or values

```
echo "Hello"
→ Hello
a=10
echo "$a"
10
```

Scanned by CamScanner

2. date - displays current time and date.
has predefined arguments start with %.

3. echo> creates a file.
Eg: echo>abc.text.

4. cat - displays or reads file.

5. cat>abc.txt - writes into the file.
Hello. } will be written to abc.txt.
A section. }
→ ctrl D. (Terminate).
If the file exists it writes into file otherwise it creates and writes. If file already has some contents, it overwrites.

6. clear - is used to clear the screen.

7. wc count.txt. - word count.
output → no.of lines no.of words no.of chars filename
wc -l → no. of lines
wc -w → no. of words
wc -c → no. of characters.
wc -lc → no. of lines and characters.
Eg: wc.
Hello
Hi
→ ctrl D.
output: lines, words, chars.

8. ls - displays all files, directory, subdirectory etc.

Eg: ls | wc
output of ls is input to wc.

Scanned by CamScanner

## Locating commands.

Unix has a list of predefined commands that has a predetermined location and commands existing as an executable file.

type - command that gives the location of a command file.

$type command-name

Eg: $type date.

⇒ date is hashed (/bin/date)

/ → root directory.

$type echo.

⇒ echo is a shell builtin.

## PATH

It is a shell builtin variable that has the name of sequence of directories a shell has to search for executing the command.

echo "$PATH"

⇒ /usr/ / : /usr/ / : /usr/ / :

## Internal and External commands.

External commands: Commands that exist as a seperate file in the unix

Ex: date is found in the location /bin/date

Internal commands: Some commands exist with in a shell and they don't have a seperate file. Such commands are also called as shell builtin commands.

Ex: echo.

Scanned by CamScanner

Flexibility of command usage:

1. Combining commands: two or more commands can be executed on a single command line. Each command is seperated by a ;
   Ex: date;echo "hello"

2. A command line can overflow: For few commands, they can be written in multiple lines.
   Ex: echo "hi
   hello
   how"

3. Entering a command before the finish of previous command:

9. man:
   man is a command that is used to browse unix manual (documentation) page. Manual page has detailed description on all predefined commands present in unix.
   Syntax: man [options] command-name

Navigation commands, f → next page b-previous page
q- quit.

Using man sectionwise.
   Ex: man 2 mkdir. or man -s2 mkdir.
   It goes to section 2.

Using man to understand man.
   man man.

man with -k option - used to search the pattern

Scanned by CamScanner

in different sections.
syntax: man -k pattern.

10. whatis: command is used to know the sections containing a particular command.
whatis commandname.
Ex: whatis mkdir ⇒
mkdir (1) - make directories
mkdir (2) - create a directory.

11. apropos: is an alternative to man -k.
Syntax: apropos pattern.

## The File System.

File is an array of bytes which is used to store certain information.
If each file in unix is associated with its own set of properties called as file attributes and these file attributes are found in the secondary storage which is not accessible directly to the user.

Three categories of file.
1. Ordinary file (Regular)
2. Directory file.
3. Device file.

1. Ordinary files:
Ordinary files are also called as regular file which store some data inside it.
2 types:
1. Text file: Any file consisting of only printable characters
Ex: C program, text file.

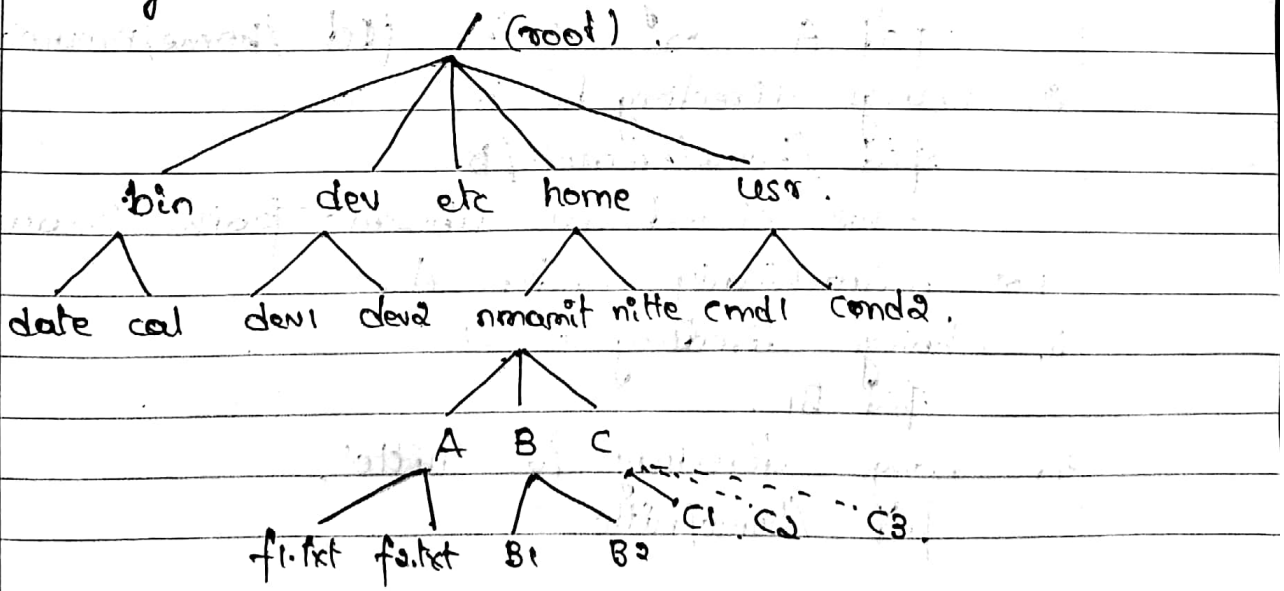
Scanned by CamScanner

2. Binary file : File consisting of both printable and unprintable characters.
Eg: multimedia files, executed object code.

2. Directory files:
A file that contains the information of other files and subdirectories if houses.
Every directory file maintains two information for each file: file name, Inode number (An unique identification number)
Inode table contains all attributes of a file.

3. Device files:
Each device is associated with a file called as device file. This file is meant for performing all I/O operations and for maintaining some information corresponding to the device. All the files devices files are found in a special directory called '/dev'.

Parent child relationship in unix file system:
File system in unix.




Scanned by CamScanner

HOME variable:

(Home) HOME variable is a shell builtin variable that stores the location of home directory of a user

Eg: $ echo "$HOME"
$ /home/nmamit.

absolute path name — path starting with root directory

12. pwd: It is a command that prints the current working directory of a user who has logged in

Eg: $pwd.
$ /home/nmamit/A. or $ ~/A.
↳ "$HOME"

13. cd: It is a command that is used to change the current working directory of a user.
cd [directory path]
directory path is either absolute or relative path which is legal.

Eg: Consider current working directory /home/nmamit
1. change directory to A.
$cd A or $ cd /home/nmamit/A.
2. change directory to B.
$cd /home/nmamit/B
- we should use absolute path because B is not heirarchically under A.
3. change directory to B1
$cd B1.
4. change directory to 'nitte'.
$cd /home/nitte.

cd command without any argument takes to home directory.

Scanned by CamScanner

14. mkdir : It is a command for creating new directories
$mkdir arg1 [arg2, ... argn].

Ex: Consider current working directory /home/nmamit.
1. Create a directory C1 under C.
$mkdir C/C1.
2. create directories C2 & C3 under C.
$mkdir C/C2 C/C3.
3. create a directory tree 'D/D1/D11'.
$mkdir -p D D/D1 D1/D11.
4. create a directory N1 under nitte.
$mkdir /home/nitte/N1.

Conditions when mkdir command fails to execute.
1. when there is an ordinary file existing with the same name of directory to be created.
2. When there is a directory existing with the same name of directory to be created.
3. When there is a permission denied.
Ex: if we try to create a directory under home.

15. rmdir: It is a command used to remove the existing directory.
rmdir arg1 [arg2, ..., argn]
Ex: consider current working directory /home/nmamit
1. remove directory A. $rmdir A (failed).
$rmdir A/f1.txt A/f2.txt.
$rmdir A.
2. remove directory C1 & C2 from C.
$rmdir C/C1 C/C2.
3. remove directory C.
$rmdir C/C3 C.
4. remove directory tree D/D1/D11.
$rmdir D/D1/D11 D/D1 D.

Scanned by CamScanner

Conditions when rmdir command fails to execute.

1. When there is no directory exist
2. When the directory permission is denied
3. When directory is not empty.
4. You cannot remove a current directory or a directory placed above it in the heirarchy.

Absolute pathnames:

Pathname that starts from the root (\) are termed as absolute pathname.

Eg: /home/nmamit.

When the path of the file is not heirarchically found under current directory.

Relative pathnames:

Any pathname that does not start from (\) is termed as relative pathname.

Eg: A/a.txt.

When the path of the file is heirarchically found under the current directory.

. and .. :

. (single dot): refers to current directory.

.. (double dot): refers to parent directory of the current directory.

Ex: cd ..

cd ../..

$pwd

/home/nmamit/A/A1

cd ..

pwd

/home/nmamit/A

cd

Scanned by CamScanner

4.16. cat : command used to display the contents of a file or to create a file.

cat for displaying a file.

Syntax : cat [options] file1 [file2, ..., file n]

Examples :

1. $cat hell.txt. - displays contents of hell.txt.
2. $cat hell.txt well.txt.

options : 1. -v : displays even the unprintable characters. Ex: $cat /bin/date

$cat -v /bin/date.

2. -n : displays the line numbers along with text. the contents.

$cat -n hell.txt.

Example : $cat -nv hell.txt (/bin/date)

cat for creating a file.

cat >filename : > ⇒ redirection symbol

It creates the 'filename' if it is not existing. If it exists, contents will be overwritten.

16. cp : cp command is used to ~~the~~ copy the contents of one file into another file.

cp [options] sourcefile destinationfile.

sourcefile : an ordinary file & has to exist.

destinationfile : an ordinary file that may exist or not. If exist it will be overwritten else it creates a new file.

$cp hell.txt f1.txt.

$cp f1.txt A/f2.txt.

Copying the contents of multiple files into destination directory :

Scanned by CamScanner

syntax: cp s1, s2 sn target_directory
Here target_directory should exist.
s1, s2, ... sn must be ordinary files.

cp command to copy a file into current directory.
cp A/f1.txt . indicates current directory

cp options:
1. -i : invokes an interactive copying if there is a overwriting of destination file.
cp -i f1.txt f2.txt
if f2.txt exists ⇒ It displays
overwrite f2.txt ?
Press y-yes n-no.

2. -R : performs recursive copying of files from one directory into another directory.
cp -R [source-directory] [Dest. directory]
source_directory - should exist.
Dest. directory - may not exist. If does not exist, a new directory will be created.
Ex: cp -R A B.
If B exists, B contains A.
files f1 f2 ...
If B does not exist.
B contains - B → files inside A.
(not A-directory)

rm : this command is used to remove the ordinary files or files from the directory.
rm [options] file1 [file2 ... filen].
Examples: rm chap1.
rm chap1 chap2.
rm * . (removes all files)

Scanned by CamScanner

options:

1. -i : interactive deletion of a file.
2. -R : recursive deletion of all the files from a directory including that directory.
3. -f : forces the removal of write protected file.

→ syntax: rm -R directorypath.

Ex: rm -R

18. mv : is used for
    1. renaming a file or directory
    2. to move a file or files into different directory

renaming a file.

mv sourcefile destfile

sourcefile - should exist.

destfile - may exist or not.

If exist it overwrites.

After renaming sourcefile doesnot exist.

renaming a directory:

mv sourcedirectory targetdirectory

(sourcedirectory: should exist; targetdirectory: may exist)

Ex: mv D DEE

DEE does not exist, D will be named as DEE.

If DEE exist, it will copy D into DEE.

To move multiple files into directory:

mv file1 file2 .. filen targetdirectory

files & targetdirectory should exist.

Scanned by CamScanner

# The Shell.

## Shell Interpretive Cycle:

- Shell is an interface between a kernel and the user.
- Shell interprets a command and converts it into an activity (Command line interpreter).

## Activities performed in Shell interpretive cycle:

1. Shell basically puts a listener in the command line interpreter specifying that it is expecting a command to be entered.
2. On entering a command shell scans an entire command line for the meta character and expands the meaning of meta character to recreate a simplified version of command.
3. It then sends the command for execution to the kernel.
4. Shell then waits for the command to complete the execution and,
5. On executing the command reissues a prompt specifying that it is ready to accept next command.

## Wild Cards:

Shell uses metacharacters for matching the filenames. and these characters are called wildcards.

Wild Cards example:

1. The * : Matches zero or ~~more~~ any number of characters.

chap* ep chap01 chap02 chapx chapy cap3
abc.txt pqr.txt hell.java well.java chap03
chap04 chap05

Scanned by CamScanner

1. A command to list the names of files beginning with chap only.
   $ls chap*
2. List out all the files with extension .txt.
   $ls *.txt
3. Remove the files that has extension .java.
   $rm *.java
4. Write the command which lists only files chapo1, chapo2. Use wild cards.
   $ls chapo*

2. The ? : Matches a single character with any symbol but not Null.
   1. List the names of files with five chars in which first four are chap.
      $ls chap?
   2. Remove all the files from the directory having the length of four chars.
      $ls ????
      rm
   3. What is the output of
      $ls chap??
      = chapo1, chapo2

3. The [ ] (character class): It matches a single character that is specified within the pair of braces.
   1. List out the filenames with ~~five~~ six chars in which first ~~four~~ five chars are 'chapo' & sixth char should not exceed the no. 3.
      $ls chapo[1,2,3]
      or
      $ls chapo[1-3]
      or
      $ls chapo[!4-5]

Scanned by CamScanner

2. List out the filenames with names chapa, chapy & chapz.
$ls chap[x y z]
3. List the filenames beginning with capital letter.
$ls [A-Z]*
4. List the files ~~with~~ ~~start~~ which doesn't begin with numbers
$ls [!0-9]*

4. The {} : Matches the patterns specified within the braces.
Syntax : {pat1, ..., patn}
1. List the files with an extension .java or .c or .txt.
$ls *{.java, .txt, .c}
or $ls *.{java, txt, c}

## Escaping and Quoting :

Escaping and quoting are the techniques used to supress the meaning of meta characters.

Escaping : Adding a back slash (\) just before a meta character to supress the meaning.
To remove file chap*
$rm chap*
To remove chap[1-3]
$rm chap\[1-3\]

Quoting : Enclosing a meta character or command argument with a single set of quotes is called as quoting.
$rm chap'*' or $rm 'chap*'
$rm chap'[1-3]' or $rm 'chap[1-3]'

Scanned by CamScanner

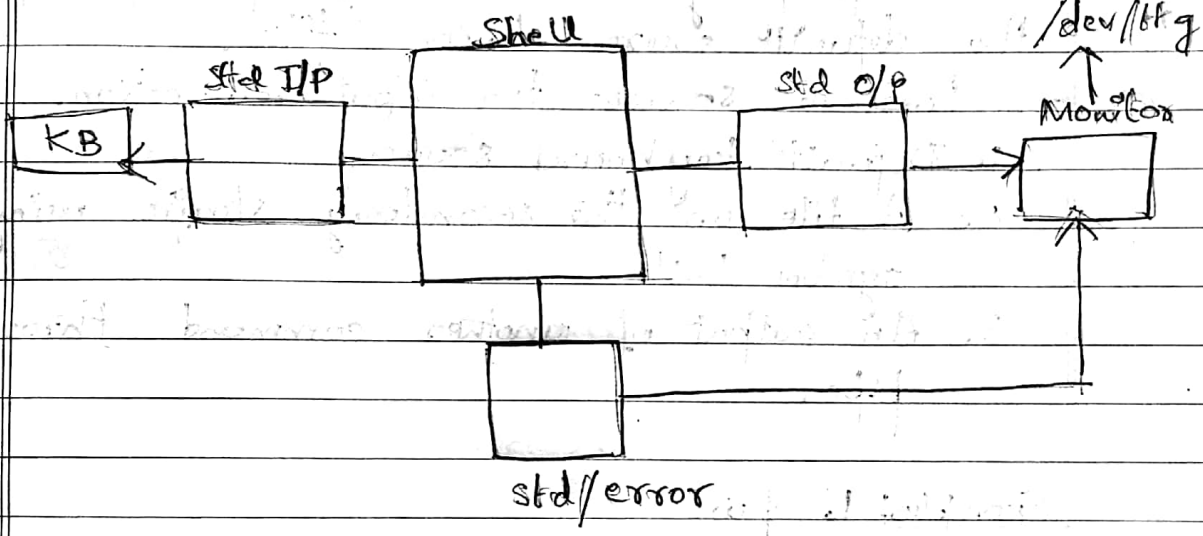
Write a command to print the message value of $1 = 60 RS.

echo "Value of $1 = 60 RS" – X. $1 is considered as variable.

echo "Value of \$1 = 60 RS".

echo 'Value of $1 = 60 RS'

## Redirection : The three standard files




1. Standard input file : It is a file / stream representing the input operations. Default source is keyboard. file descriptor number of this is 0.

2. Standard output file : It is a file / stream representing the output operations. Default destination is display system. File descriptor number is 1.

3. Standard error file : It is a file / stream representing the error messages thrown by the shell. Default destination is display system. File descriptor number is 2.

Scanned by CamScanner

## Redirection :

It is the process of unplugging the standard file from its default source / destination and connecting it with a file present on the secondary storage.

1. < represents input redirection
2. > an output redirection for writing.
   >> an output redirection for appending.

## Input redirection :

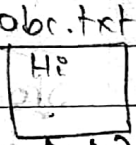
Unplugging the standard input file from the default source device / file.

Possible sources for input redirection.

1. Default keyboard source.
2. A file on the secondary storage using the symbol '<'
3. An output of another command through the pipe.

Examples: 1. $ wc
Hi ↵
ctrl D
1 1 3 (Default keyboard source).

2. $ wc abc.txt ↵
1 1 3 abc.txt.
(wc reads from abc.txt)




3. $ wc <abc.txt. or $ wc 0<abc.txt.
(file descriptor no)
1 1 3.
(input redirection takes place).

Activities performed by the shell are :

1. On scanning '<' symbol, shell opens the file abc.txt for reading.

Scanned by CamScanner

2. It unplugs the standard input from the default source and connects it to abc.txt.
3. wc finally reads the content from the standard input.

Output redirection:

Unplugging standard output from the default destination.

The possible destinations

1. Default display.
2. A file on secondary storage using '>' or '>>'
3. An output of another command through the pipe.

Examples: 1. $wc hell.txt

hell.txt
Hi

1 1 3 hell.txt (Default display system)

(2) $wc hell.txt >out.txt. (output redirection takes place).

out.txt
1 1 3 hell.txt

Activities performed by the shell are.

1. On scanning '>', shell opens out.txt for writing.
2. Shell then unplugs standard output from default device and connects it to out.txt.
3. ~~the~~ f "wc" then opens hell.txt for reading.
4. "wc" finally writes output onto standard file which is redirected onto out.txt.

(2) can also be written as

$wc hell.txt 1>out.txt or $wc hell.txt 1>>out.txt (appends).

Error redirection:

Unplugging error file from the default device.

Scanned by CamScanner

Example: Current directory contains only hell.txt

| hell.txt |
| --- |
| Hi |

1. $cat well.txt ↵
   cat : well.txt : file not existing.

2. $cat well.txt > err.txt.  (output redirection)
   cat : well.txt : file not existing

3. $cat well.txt 2> err.txt ↵ →
   (error redirection)

| err.txt |
| --- |
| cat : well.txt : file not : existing |

4. $cat hell.txt 2> err.txt ↵
   Hi.

5. Execute the cat command for the file pin.txt in which case output has to be redirected onto out.txt. and error onto err.txt.
   $cat pin.txt 1> out.txt 2> err.txt

Filters :

Four categories of commands based on standard files:

1. Set of commands that neither reads from std input source nor writes onto std o/p file
   Ex: mkdir, cd.
2. Set of commands that reads from std input source but do not write onto std output.
   Ex: lp
3. Set of commands that do not read from std input but write onto std output.
   Ex: pwd, ls
4. Set of commands that read from std input and writes onto std output. ↵
   Ex: wc, cat  are called as filters.

Scanned by CamScanner

## Two special files : /dev/null and /dev/tty.

1. **/dev/null :-** is a file

null is a file inside the directory -/dev that is having the size of '0' bytes.

It never grows in size even after writing the contents.

Ex: $ls a.txt > /dev/null

2. **/dev/tty :**

tty is a file present inside the directory /dev.

- This file represents the terminal of the user who has logged in.

Ex: $ls >/dev/tty

## Pipe :

A pipe is a special operator that is used to connect the output stream with an input stream.

Symbol : |

Pipe redirects output of a command as an input to the other command.

Syntax : comm1 | comm2

Ex: To count number of files in current directory

$ls | wc -l

## tee :

tee is a command that is used to create duplicate form of an input and one copy of input is stored in the file and another copy of input is written on to the std output.

Syntax : $tee filename.extension.

```
$tee 01.txt ↵
hello ↵
ctrlD
→.
```

01.txt: hello

std.output: hello

Scanned by CamScanner

1. An example to display the files in the current directory and to display the count of files in the current directory.

```
$ls | tee dev/tty | wc -l.
```

2. List all the files in the current directory and display the list as well as store it in a file list.txt

```
$ls | tee list.txt.
```

3. 
```
$ls | tee list.txt | wc -l.
```
⇒ no. of files present in current directory.

Command Substitution:

It is the process of executing one command inside another command.

A command is executed and its output is taken as an argument to another command.

Command substitution is performed using ` ` [Back quotes].

Ex: Display current date.

```
echo "Todays date is `date`".
```

Scanned by CamScanner

# Basic File Attributes.

There are 7 attributes → basic attributes.

**Listing of file attributes:**

Command: $ls -l

Lists 7 attributes of a file and lists the names of the file in alphabetical order.

**syntax:** $ls -l :

or $ls -l filename(s)

Consider → the current directory has chap – ordinary file and dir – directory.

$ls -l.

total : 8 → it occupies 8 blocks in harddisk.

-rw-rw-r-- 1 nitte nitte 0 Aug 24 16:45 chap

drw-rw-r-- 2 nitte nitte 4096 Aug 24 16:45 dir

① ② ③ ④ ⑤ ⑥ ⑦

① **File type and Permissions:**

**File type:**

first character : - : ordinary file

d : directory file.

**File Permissions:** (2-10)

2-4 : owner permission

5-7 : Group permission

8-10 : Other permission.

3 characters for each type of permissions.

1→ Read permission. r: read & -: no read

2- Write permission w: write & -: no write

3- Execute permission x: execute & -: no execute

⑤ Links: different names for a single file. Chand(links)

When created, directory has '2' as link value it has its own name and '.'.

Scanned by CamScanner

③ owner name : Name of user who has created the file.

④ group name : Name of group to which user belongs to. If user does not belong to any group, user itself be a group

⑤ size of files in byte.

⑥ date and time of last modification.

⑦ Name of the file.

Listing only directory attributes :

Command : ls -ld

Syntax : ls -ld → without argument.
It lists all the directory attributes of current directory.
- 1st attribute here is . indicating current directory.

ls -ld directory name (s)
Ex : ls -ld A B C
ls -ld * → all the d attributes of all directories in current directory.

* chmod :
It is a command that is used to change the permisions of the file.
There are two types of permisions in chmod.
① Relative permision.
- change the permision of a file to relative to the current permision.

Scanned by CamScanner

② Absolute permision
- change the permision of a file from the scratch without the knowledge of current permisions.
- This command can be executed only by the owner of the file and the super user.

1. Relative permision:
It changes the permision of only the required set leaving behind the old permision as it is.

Syntax: $chmod categoryoperationpermision filename(s)

(owner) user - u
group - g
others - o
all - a

+ → Add.
- → Remove.

read - r
write - w
execute - x.

Assume that test. is file with permision
rw-r--r--

Examples:
1. To add execute permision for user and group.
- $chmod ug+x test (rwxr-xr--).
2. To remove write permision from user and read permision from group.
$chmod u-w, g-r test. (r-x--xr--)
3. To add write permision for all categories of user.
$chmod a+w test. (rwx-wxrw-)

2. Absolute permision:
This changes the entire set of the permision without having to know the current permision.

Syntax: chmod 010203 filename(s).
↓
octal numbers

Scanned by CamScanner

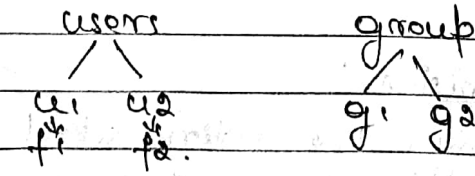
01 - octal for user (owner)
02 - octal for group.
03 - octal for others.

| Octal | Binary | Permision |
|---|---|---|
| 0 | 000 | --- |
| 1 | 001 | --x |
| 2 | 010 | -w- |
| 3 | 011 | -wx |
| 4 | 100 | r-- |
| 5 | 101 | r-x |
| 6 | 110 | rw- |
| 7 | 111 | rwx |

1. For a file test set the owner permision with all the permision, for group set only read and write and for others deny all the permissions.
rwxrw----.
$chmod 760 test.

## Changing the file ownership:

A file has two owners:
1. owner of the file → creator or user
2. Group owner → A group to which an user belongs to.




u1 - g1 and g2
u2 - g2.

Two commands: chown, chgrp

Scanned by CamScanner

1. chown : Can change both the owner of the file and group owner of the file.
2. chgrp : Can change only group owner.

chown : This command should be executed by the super user only.

syntax : chown owner[:group] filename(s)

$su ↵
password :- *** ↵
#chown u2 f1 : f1 → u2.
group → g1.

#chown ~~u1[:g2]~~
u1:g2 f1. f1 → u1.
group - g2.

exit ↵

chgrp : Command can be executed by the super user and some systems allow the execution to be done by the ~~own~~ owner of the file.
- When changing the group, the user has to be added to the existing group to which a change is required.

syntax : chgrp group-name filename(s).

u1 → g1 & g2
u2 → g2.
u1 → f3 → g1
f4 → u2 → g2.

- change the group ~~none~~ owner. file f3 to g2.

chgrp g2 f3

Scanned by CamScanner

change the group owner of file f1 as g1.
- Not possible as u2 is not under g1.

## File System and inode:

- Each and every file is associated with a table called inode table.
An inode table has a unique id called as inode number.
An inode number of a file can be displayed using "ls -i" command.

## Entries inside the inode table:

1. File type.
2. File permision
3. Number of links. (Hard links)
4. user id (uid)
5. Group id (gid)
6. Size in bytes.
7. time of last modification.
8. time of last access.
9. time of last inode change.
10. Array of pointers pointing to the file blocks.

## Links:

Links are the names associated to a particular file.
Links are of two types.
1. Hard links.
2. Symbolic links (Soft links)

### 1. Hard links:

Hard links are the aliases created for a single

Scanned by CamScanner

copy of file that is existing.
All those aliases can be used to access a same copy of existing file and those aliases are called as Hard links.

If an original copy is deleted, the aliases will restore the contents of the file.

Creation of hard link:

Syntax:

1. ln filename1 filename2
   - filename1 → Existing file.
   - filename2 → Should not exist. Hard link file.

filename1 and filename2 have same attributes, They are two names for a single file.

Ex: ln f1 f2

2. To link multiple files into directory:

ln filename1 filename2 ... filenamen directoryname

- filename1 filename2 ... filenamen → Existing ordinary files.
- directoryname → Existing.

Ex: ln f2 f3 ldir

⇒ ldir

f2 f3 → links to original files f2 and f3.

Applications of hard links:

1. To reorganize the directory structure
2. Provides a protection against the accidental deletion of the file
3. Two files with different extension but same contents can act as link files.

Scanned by CamScanner

Disadvantages of hard links:

1. It can not link two directories.
2. It can not link the files existing in different file systems.

2. Symbolic links (Soft links):

Symbolic link is basically a seperate file that is pointing the file it is linked to. (pointer)

A symbolic link file is a seperate type of file denoted by 'l'.

When an original file of the symbolic link file is deleted, the symbolic link becomes a dangling link.

Creation of symbolic link:

syntax:

1. ln -s filename1 filename2. (file type → l)

filename1 → should be existing

filename2 → non existing

ordinary files.

filename2 will be filename2 → filename1 (attributes)

2. Symbolic link to link directories.

ln -s directory A directory B.

directory A → Existing

directory B → May or maynot exist.

Ex: A. ln -s A B → B → A.

f1 f2.

A B ls -s A B →

f1 f2

## Modification and Access time:

Listing time parameters:

1. To list the last modification time : ls -l.
2. To list the last access time : ls -lu.
3. To list the last inode change time : ls -lc.

To list the parameters in a sorted fashion.

1. To sort the list based on the modification time
   ls -lt.
2. To sort the list based on access time.
   ls -lut.
3. To sort the list based on inode change time.
   ls -lct.

Changing the time stamps:

- touch is the command used to change the time stamp value of last modification time and last access time.

Syntax: touch options expression filename(s)

→ options: -m : to change modification time
-a : to change access time.

→ Expression: [YY[YY]] MM DD hh mm
(24 hr format.)

1. 'Touch' without options and expressions.
   - touch filename(s).
   - When used this way, touch alters both modification and access time and sets it to current date and time.
   - Filename may or may not exist.

Eg: 1. Consider a file f1 not existing
$touch f1.

Scanned by CamScanner

2. Consider f2 was created last year.
$touch f2.

2. 'touch' without option and with expression.
- When used without option and with expression it will change both modification time and last access time to the time specified in expression.

Ex: touch 201603030830 f1. ⇒ 03-03-2016 8:30.
or
touch 1603030830 f1
∴ (-t) - to execute.

3. 'touch' to change only modification time.

Ex: Change the modification time of the file f2 to Feb 02 -- 2022 around 4 AM.

touch -m 202202020400 f2.

4. 'touch' to change the access time.
Ex: 1. touch -a f3.
To change the access time of the file f3 to the current date and time.
2. touch -a (-at) 12211630 f4.
To change the access time of the file f4 to Dec 21 4:30 in the evening.

## umask (user mask)

umask is a variable that holds the octal value of the permissions which defines the default permission for the files when created.
By default for any unix system, the permissions are:

Scanned by CamScanner

1. file : 666 → rw-rw-rw-
2. directory : 777 → rwxrwxrwx

Whenever a file or directory is created the permission is assigned after subtracting umask value from default permission.

Ex: If umask=002.

$$\begin{array}{r} 666 \\ -002 \\ \hline 664 \end{array}$$

⇒ new file will have 664 permission

Setting the umask value :

syntax : umask 0102o3

Ex1: $umask 022.

$$\begin{array}{r} 666 \\ 022 \\ \hline 644. \end{array}$$

$touch x1

x1 → rw-r--r--

$mkdir x

x → rwxr-xr-x

Ex2: $umask 001

$$\begin{array}{r} 666 \\ 001 \\ \hline 665. \end{array}$$

$touch f1.

f1 - 666

rw-rw-rw-

→ This sets x for others.

Umask value can not result in setting any permission, removing any permision is possible. If it results in setting any of the permision, system assigns the default permission to the file.

Command to save sessions :

$script --timing=classtime.txt class1

:

$exit

$ scriptreplay -timing classtime class1

Scanned by CamScanner

## find

find is a command that is used to recursively search a given path based on a particular criteria and carries out the required action on the located files.

General syntax:

find pathlist selection-criteria action.

Selection criteria:

Ex: -name filename.

Action: -print. (print the location of the file from the specified path list.

Ex: * find . -name abc -print.
(-name abc: selection-criteria; -print: Action)

⇒ ./A/abc. ⇒ There are two files with name
./B/abc. 'abc', one inside directory A and other inside 'B'.

* To locate and print all the files inside the current directory with the extension .c.

find -name "*.c" -print.
(* → used to identify wildcar

Selection criteria.

1. -name.

2. -inum inode-number.
   - Selects the file based on the inode number.

3. -type x.
   x can be f → d, l. (ordinary-f, directory-d, link-l)
   - Selects the files of specified type.

Scanned by CamScanner

4. -perm. 0102,03
octal values.

- Selects the files based on the permission.

5. -mtime +int - (i)
-mtime -int - (ii).

It locates the file based on the modification time.

i> locates files which is more than 'int' days.
ii> locates files which is less than 'int' days.

Ex: 1. find . -mtime +2 -print.
2. To locate all the files which are more than 5 days and less than 10 days.
find . -mtime +5 -mtime -10 -print.

6. -atime : locates the file based on the access time.
-atime +int → locates files more than 'int' days.
-atime -int → locates files less than 'int' days.

→ 1. Write a command to find all the ordinary files in the current directory with the permission '444' and which is being not accessed for more than 365 days.

find . -type f -perm 444 -atime +365 -print

Action commands

1. -print

2. -ls : lists out all the attributes of the located file along with the inode number.

3. -exec : syntax: -exec cmd {} \;

Scanned by CamScanner

Executes a particular command 'cmd' taking located file as an argument.

Ex: Write a command to find all the symbolic link files in the current directory and on locating each file, remove all the files.

```
find . -type l -exec rm {} \;
```

H. -ok :

syntax: -ok cmd {} \;

Executes a particular command on the located file interactively.

⇒ y|Y - executes the command

n|N - does not execute the command.

Ex:

```
find . -type l -ok rm {} \;
```

Ex: Find all the files with an extension .java and print their path list along with the display of all the attributes.

```
find . -name "*.java" -ls -print
```

Print all the files in the current directory that is not having the extension .c.

! - operator can be used with any selection criteria.

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

Locate all the files with the extension .c or .txt inside current directory.

Scanned by CamScanner

multiple selection criterias seperated by spaces will be by default 'and'.

find . \( -name "*.c" -o -name "*.txt" \) -print

or.

## Shell Variables:

### Initialize a variable.

variable_name = value  ex: x = 10.

Scope of the variable remains till the shell is terminated.

### Initialize a variable with another variable:

y = $x

### Unsetting the variable:

It will unset the value of set variable.

Syntax: unset variable_name.

Ex:
```
a=5.
echo $a.
-5.
unset a.
echo $a.
```
(doesnot print anything).

### readonly:

It is a keyword that will not allow the variable to be rewritten or reinitialized.

Ex:
```
b=20
echo $b
20.
b=30
echo $b
30.
```

```
readonly b.
echo $b.
30
b=40.
⇒ error message.
```

Scanned by CamScanner