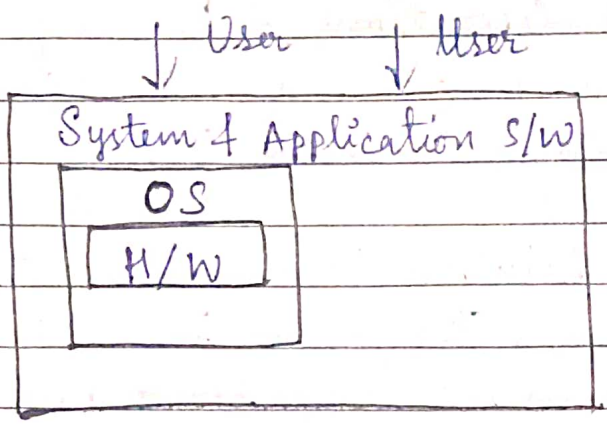
UNIT - 1

3/11/22

INTRODUCTION TO UNIX

Structure of OS :-



→ The hardware part that provide the basic computing resource that is CPU, memory and I/O device.

→ The OS control and coordinate the use of hardware among various application programs.

→ From the user point of view, the OS should give some attention to the performance.

→ The system point of view: OS act as resource manager. It requires to solve the ~~program~~ problem like CPU time, memory space, high storage space.

→ The OS decide hardware to allocate resource to do a specific job. It also control various I/O device and user program.

Function of OS :-

1. Program counter :- OS provide editor, debuggers to assist the programmers in creating program.

2. Program Execution :- No. of task require to execute a program done by OS.

3. I/O Operations :- Runing program may require input output. A user cannot execute I/O operation without OS.

4. Error Detection :- The OS detect the different type of error. The error include illegal instruction in program memory error and power failure.

5. Resource Allocation :- The OS collect all resources in the network and ~~grant~~ grant.

6. Accounting :- The OS can keep track of which user how much time and what kind of resource is used by the user.

Different type of OS :-

A. Batch System :-

- In early computer the common input devices were card reader, tapedrive, etc.
- The user prepared a job which consisted of program, data, some control information about the job and submitted to the computer operator
- The programmers would leave their program with their operator that would sort the program

into different batches with similar requirements and run it batch.

- & The output is send back to the appropriate programmer.
- In this system a major task of OS is to transfer data automatically one job to another.
- The drawback of the system is most of the time CPU sits ideal.

B. Multiprogram System :-

- In non-multiprogramming system only one program can be executed in a particular time
- This program has total control over all the resources. If the program is waiting for the I/O device then CPU sits ideal or it wait for a long time, so, it will affect the performance.
- Some resources are ideal and CPU utilization is poor. So keep all the resources busy at all time multiple program can be executed in a single time.
- In multiple programming system no. of programs are residing in a main memory at a time.
- The OS choose and begin to execute. One of the job needs to wait the CPU switch to another job

and so on.

Operation of multiprogram system :- All the jobs that enter in a system are kept in a job pool. To select job from the job pool we use regular job scheduling.

Advantages :-

→ Efficient memory utilization
→ CPU never sits ideal.

C. Multitasking System :-

- It is a logical extension of multiprogramming system. CPU execute multiple job by switching among them and switches occur so frequent that user can interact with the program while it is running.

- In the system CPU time share by all the process. So the system is called time sharing system.

* Client Server System :-

- All the computer are connected to a centralize system. This centralize system act as a server. Satisfy the request generated by client system.

- Computer server provide a interface where client send a request to perform an action and result send back to the system.

* Real Time System :-

- It is used when rigid time requirement having placed on the operation of processor. Processing of data having done with the define constant.
- A real time function is correct only if its returning correct result with its constant.

* Desktop System :- This system emphasizes on maximizing user convenience responsiveness rather than maximizing CPU and peripheral utilization

D. Multiprocessor :-

- The system have more than one processor that share the computer bus, clock, memory, peripheral devices to execute their job parallely.

Advantages :-

- Increasing through put by increasing no. of process we can get more work done in less time.
- It saves money as compare to multiple single processor because they can share all the resource of system.
- Increasing reliability one processor will not all the system slow down the system.

F. Distributed OS :-

- In distributed system the processor cannot share memory or clock.
- Each processor has its own local memory.
- The processor communicate through different communication line.

MOS

Unix :-

- It is the OS which is capable to handle activities from multiple user at the same time
- The unix OS is a set of program that act as a interface between system and user.
- This computer program that allocate the system resources and co-ordinate all the detail of the computer internal is called as OS or Kernel.
- User communicate with the kernel through program kernel as the shell.
- The shell is the command line interpreter which act as a translator between user & system.
- It translate command enter by the user and conver

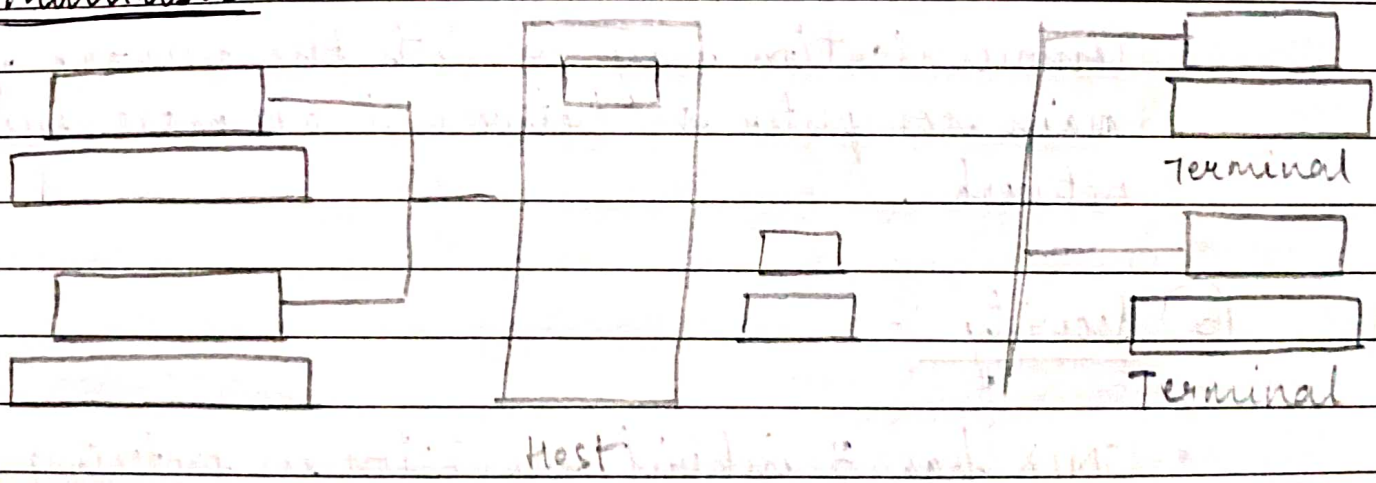
them.

Features of Unix :-

① Portable :-

- Unix is found more between platform than any other OS.
- It widespread use can be directly trace to the decision to develop it using C language between C program can be easily more from one between environment to another.

② Multiuser :-



- In a multiuser system unix installation is the host machine known as server.
- The number of terminal connected to the host machine depend on the number of card.
- Four port controller card on host machine can support four terminal.

(3) Multi-tasking :-

- It is capable of carrying out more than one job at same time. This happen by dividing CPU time between all the process being carried out.
- Depending on the priority of task the OS a lot time to each background and forground process.

(4) Communication :-

- Unix are excellent provision to communicate with the ~~foll~~ fellow user.
- Communication may be with the network of single main computer or between two or more such computer network.

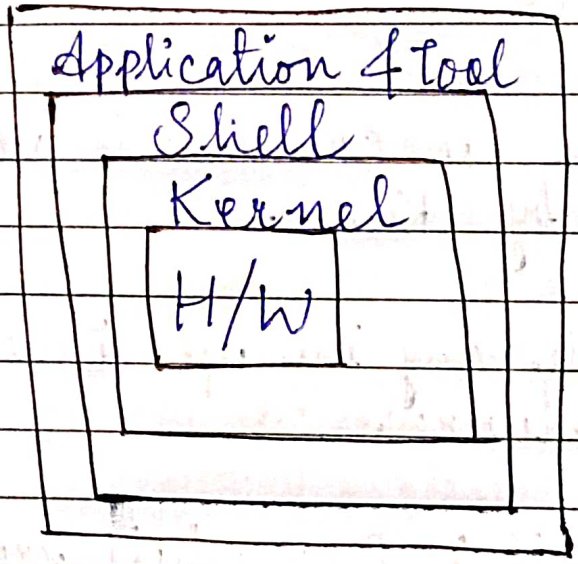
(5) Security :-

- UNIX has 3 inherit provision for protecting data.
- Provide by assigning a password and user name to individual user ensuring that not anybody come and access that.
- At the file label there are read, write and execute permission to each file, which decide who can access a particular file who can modify or execute it.

- We may ~~reserve~~ reserve read or write permission and have other permission free for the network or any two communication.

(6) File Encryption :-
It encode file into all readable format so that even if some succeed to opening it your secret or data are safe. When the owner want to see he/she can decrypt the file.

UNIX Structure



(i) The functioning of UNIX has 3 level on the outermost reside the application program and other utilises which is our language.

(ii) The heart of the UNIX is Kernel which interact with the actual hardware in machine language.

(iii) The stream line of this two modes of communication is done by the middle layer called shell

(iv) Shell is wrapper. It translates user input to a

form that Kernel can understand by calling suitable program.

(v.) Shell is the mediator and command interpreter which interpret the command that we give and convey them to the Kernel which ultimately execute them.

(vi.) Kernel has various funition. It manages file, carries out all data transfer between all file system and the hardware and manages the memory.

(vii.) Allocation of CPU time to all running program. deciding by Kernel.

(viii.) It handles any interupt. issue as it is direct dealing with hardware.

11/22

1. Bourneshell :- Bourneshell created by Steve Bourne in 1977. A Bourne Shell can interprete and execute user-defined command.
   - A Bourne Shell is ~~execute~~ enable writing and executing of shell of script which provide basic program control flow and control over I/O file.
   Symbols :- .sh, $

2. C Shell :- C Shell created by Bill Joy in 1978.
   - It has two advantage over one shell :-
   (i.) It allows alice of command that mean we can device by what name we want to call a command.

(iii.) This is very useful when lengthy commands are there and we can rename instead of writing entire command by using alice name ~~that means~~.

- If we want to save even more in typing work C shell has a command history feature for previously typed command can be recall since C shell keep track on all command issued in the command line.

  Symbol :- %, ~~.ch~~ .csh (for saving)

3. Korn Shell (.Ksh)(.$) :- It is very powerful and super set of Bourne Shell. It is designed by David Korn in 1983 at AT & T Lab.

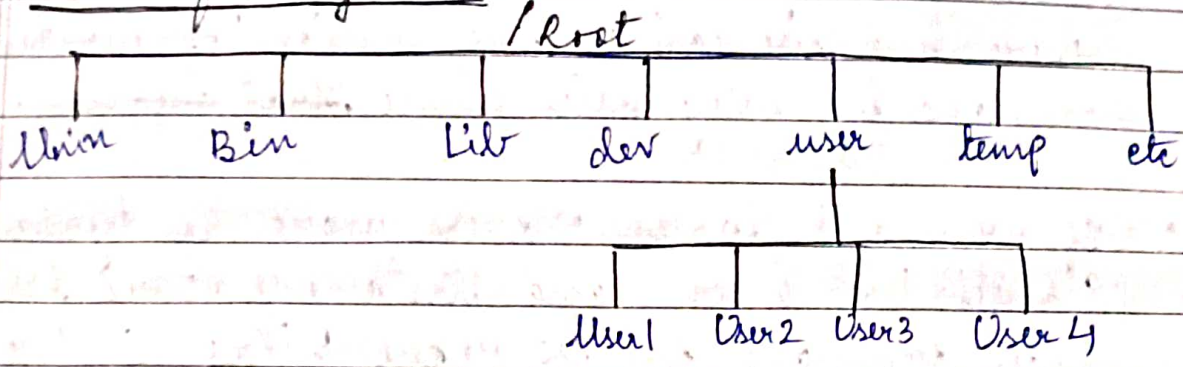
Functions :-

- The Kernel Program usually stored in a file called UNIX.
- Whereas the shell program reside in a file sh

Kernel is predefined file.
wherever kernel is present it will be said as OS.

11/11/22

Unix file system :



- The root directory is denoted by slash (/).
- Branching from the root there are ~~other~~ several other directories :- bin, Lib, device, user, temp, etc.
- The root directory contains a file called Unix which is UNIX kernel itself.
- The main reason behind to create the directory is to keep related file together and separate them from other related file.

1. **Bin Directory :-** (Binary Executable file)

- It contain binary executable file UNIX command can be either C program or shell program.

2. **Lib & Directory :-** It contain library functions from the program. The programs written under UNIX make use of this library function in the ~~to~~ Lib Directory. (It keeps predefined commands and stored in shell programs)

By default library functions

3. Device Directory : It contain file that control input, output devices like printer, terminals, disk drive, etc.

4. User Directory :- User directory contain several directory each associated with a particular user. These directory are created by System Administrat when he create the account for the different user. Each user allowed to work with his directory called home Directory and can organise his directory by creating some directory.

5. Temp Directory :- It contains temporary file created by User or UNIX.
- Since the file present in it create for the temporary purpose the file is automatically deleted when we shut down the system or restart.

6. Etc Directory :- It contain binary executable files required for the system administration.

2/11/22

7. Creating files :-
There are two commands to create files :-
(i) Touch Command
(ii) Cat command.

(i) Touch Command :- A file will be created using Touch command. Touch command size is 0 byte that's why this command doesn't allow anything to store within.

- When we want to create several empty files ~~we can use~~ quickly we can use Touch command

(ii) Cat Command ($ cat filename ↵) : When we want to store anything in a file we use Cat command by pressing Enter Key we find the cursor position move to the next line waiting for type anything by the user. After writing the content we press control + d which indicate end of the file. In UNIX when we press control + d cat command recognise that end of file character then save the text in the given file.

- For displaying the content we use Cat command itself.

re-directional operator

Ex: $ cat (>) Damilis ↵

To create filename : $ Cat > filename ↵

To display filename : $ cat filename ↵

To save : ctrl + d

- Cat command ~~concatenate~~ concatenate of two files and store the content into another file.

Syntax :- $ cat file1 file22 > file3

~~Ex :- $ cat Sneha Data > Metadata~~

Ex :- $ cat file1
Punam
cat file2
Soni
file3
Punam Soni

} To create

Linux is very difficult

Date
Page

$cat file1 file2 (>>) file3 → Appendize-directional operator (output)

- Suppose new file content has something we want this content keep in that file and content of file1 and file2 not should get appendent to it then we Append re-directional operator.

15/11/22

1. Copy Command :-

$cp file1 file2

This will copy content of file1 with file2. If file2 is not exist, it will be created.

Ex:- $cat > test1

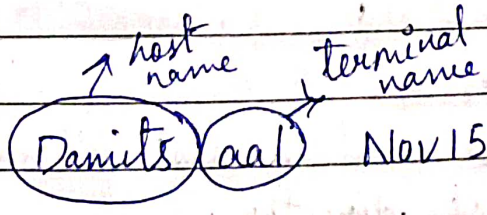
DAMITS

Ctrl + D

$cat file1 file2

$cat test1 test2

2. $who am i command (or) $who :-



3. Remove command (rm) :-

The command to remove the given file. There are 4 types of remove command.

(i) $rm filename

option/switch.

$rm -i filename.

option/switch

$rm -i filename :-

↳ It will remove the file interactively.

(ii.) $rm -r file name :- This command recursively remove all content of the directory itself.

(iii.) $rm -f filename :- It removes the file forcibly.

4. Rename Command :- It is written as (mv) → use to move the file & directory.
Syntax :- $ mv old file new file.
By executing this command the old file

Syntax :- $ mv oldfile newfile
$ mv DAMITS Ambedkar.

By executing this command the old file and directory is no longer exist.

List file and directory :-

It is written as ls.
This command is used to list files and directory stored in current directory

$ ls↵

ls command list the files and directory in alphabetical order.

cat > Suchi
cat > Test
cat > .Danish
$ ls ↵
Suchi
Test

.Danish → is hidden because of (.)

ls -a ↵
Danish (listing the hide files)

Meta Character :-

*, [ ], ?, ^

By the use of meta characters we can list all the files.

A* (If '*' is present after a character, then it will show the files ~~with~~ which starts with 'A')

A (If '' is present before a character then it will show the files which ends with 'A').

? is for searching single character.

?ain
- contain ✓
- gain
- rain ✓

*[ain] → contain [] is for more characters

20-12-22

Process in Unix :-

- Unix known as multi-user, multitasking OS.
- A process is defined as instance of an executing program.
- In OS there is a program called scheduler which decide which process should get CPU attention and when....
- At any given moment when a program is executed the scheduler submit process to a queue called process queue. At this instance the process is called submit state.
- Once submitted the process wait in queue for sometime. At this stage the process is said to held state.
- As process advance in queue it would become next one in queue to recieve CPU attention. At the stage it is at ready stage.
- Finally the process get attention of CPU and start getting executed it is called ruen ~~date~~ state.
- Some process might required to do this input output. Since, I/O is a slow operation the CPU can't be ideal till the time I/O is over. Such process are put in wait state until their I/O is over.
- A process whose execution count on end goes into complete state and remove from process queue.
- When user want to say which process are running just type PS.

```
$PS↵
```

- Unix assign a unique number to every process this number is called process ID ? (PID).
- PID start with 0 and ruen upto 32767 when maximum

number is reached it start - counting all over again from 0 onwards.
- PS designed to display only the process running at the terminal.
- PS command with option a ($PS - a)
  - a stand for all the process of user.
- If we ~~stand~~ want to see what a particular user is doing we use $PS - u Login name ↵
  $PS - U system ↵

Background Process :-

Most of the system process run in the background while user executed there processing for ground.

To run the program in Background unix provide '&' while executing a command if the symbol is placed at the end of command then command is executed in background. When we run processing a number is display on screen this number is nothing but the PID of process that you have executed in background.

Limitation :-

On termination of background process no success or failure is reported on screen we can search PID in output of PS to verify whether the process is still running or terminated. Output of a background process is directed.

- As too many process ~~run~~ running in the background the system performance is automatically degraded.

- If you log out while some of your process is running in background all this process would be abounded. This is natural because all process are childer & grand children / great gread grand childrens of the shell process when we logout (·sh) the shell process dies along with the childrens.

$CP f_1 f_2 $f_$ — give a PID	$PS -a → check where is ID is present or not

20-12-22

The Nohup :-

If we are to ensure that process that we have executed should not die even when we log out the nohup command is used. Using this, we can submit time consuming command in background, log out and leave the terminal and next day the output is ready nohup stands for no hang ups.

Killing a process :-

Some of reason why we should like to terminate a process in the middle of the execution

(i.) The terminal has hung.

(ii.) The program which is running has gone in an ~~iss~~ indefinite look and hence it is not getting terminated.

(iii.) The system performance gone below acceptable limit because of too many processes running in background. As result you ~~may~~ may want to terminate a few time consuming proclss.

* By through process id (pid) we can see the process is being terminated

In any of the situation we would like to kill the process to carry out killing process we must note pid of the process to be killed using ps command then we apply pid and kill command to terminate the process.

Syntax :- $ kill pid

Q. How kill command works?

A. When kill command invoke, it sends a termination signal to the process being killed. A signal is a mechanism to communicate with a process this signal have been given number.

Q. How can we employ kill if our terminal has hang?

A. Simply login once again through another terminal then seen who command and tty command to figure out the no. of terminal which has hang. Next we use ps -t tty ~~command~~ number to find out the pid.

24-02-23

Changing Process Priority :-

Priority of a process is decided by a number associated with it. This number is called NICE value of the process. Higher NICE value is equal to process lower in its priority. ~~The~~ NICE value a process range from 0 to 39. A process with NICE value would execute slower than the one with NICE value 20.

File Permission :- (Read, Write and Execute Permission)

Read Write Execute Permission –

$ls -l

total 2

-rwxr-x--x (1) link user 1 group 24 June 06 10:12 Test

-rwxr-x-wx 1 user 1 group 23 June 06 00:15 sample

Note :-

	Weight	
read - r	4	Total weight = 7
write - w	2	
execute - x	1	

6 6 4

There are 3 entities to which any combination of permission has assigned to owner, group and rest. Out of 9 character the first three character decide the permission hold by the owner of the file. The next set of 3 character specify the permission for the other users in the group to which the file owner belong. Last set of 3 character decide the permission for the user outside the group. Out of 3 characters belong to each set, first character indicate Read permission, 2nd character indicate write permission and third character indicate Execute permission. '–' this sign indicate deny of the permission. If all permission is available, total weight is 7. By default, all permission available for specific file. existing permission of the file

~~Suppose we want owner of the test file.~~

$chmod - change mode

324 - w-x--2--c

Suppose we want owner of the Test file has all the permission and of group and other have not any permission.

324 = w-x--w--r

427 = --x--wx-rwx

File related commands :-

(i) WC Command :- It is a simple and useful command that counts the number of line, words, character in a specific file or files.
It comes with the option of option (-l) L option (-w) W., option (-c) C i.e. line, word character which allow the user to obtain the number of lines, words, characters individually or any desired combination.

$wc - lc file1 file2

(ii) Sort :- As the name suggest sort command used for sorting of content of file. It can merge multiple stored file and store the result into another file.

Sort command based on the comparision of first character in each line in the file.

If the first character of two lines is same then it go for second character in each line and so on

The sorting is done according to that ASCII collating sequence that means it sort the space and task first then punctuation mark followed with number then uppercase and lowercase.

$ sort filename
(It will sort the content of file name)

$ sort file1 file2 file3
(It will sort at a time 3 files.)

$ sort -o (no space) file4 file1 file2 file3
(it will sort all sorted file)

$ sort -u -o file4 file1 file2 file3
If there are repeated lines in each file and we want to show that lines only ~~use~~ once we use -u.

- **Cut Command** :-

(i) cut is also a filter like sort
(ii) It pick up given number of character or fields from the specific lines.
(iii) It only cut selected field or character.
(iv) $ cut -f 2,7 filename.
$ cut -f 2,7 sneha.
(v) $ cut -c1 -15 filename
$ cut -c1-15 Chirie.

- Grep Command :-

(i.) Grep command is used for globally search or regular expression and print it.

(ii.) This command search the specified input for a match with the supplied pattern and display.

(iii.) While forming the pattern to be search we can use the meta character.

Syntax :-

(a.) $ Grep girl file1 [To search word girl in file1]

output :- We are the girl
file 1 :- We are the girl
file 2 :- Girls are sweet.

It will display the pattern with that line.

(b.) $ Grep girl file1 file2
output :- We are the girl file1
girls are sweet file2.

(c.) $ Grep "the girl" -i -n file1
This command search pattern enclosed in the inverted comma -i makes cash intensive -n also causes number of lines in which pattern was found to be printed.

(d.) $ Grep [Gg]irl

(e.) $ Grep g??l file1

(f.) $ Grep -va*
It will show the character which does not starts

with a.
It display the lines words does not starting with a.

- $Grep va*

It will show the character starts with a.

"[abc]':-
It will search only those lines which begins with abc

'[a-e]$':-
It will search the lines which end with any character between a to e.

chmod [who][+/-/=] = [permission file]
who indicates to whom the file permission assign i.e., owner, group or rest.
o = owner
g = group
u = rest.
+ : It refers to add permission
- : It refers to remove permission.
= : It refers to add specified permission and take away all other permission.

$chmod +w file1

$chmod og -x file2
eg:- rwxrwxrwx -x rwxrwx file2
rw-r--r-x.
$ chmod go+r, go-w file3
$ chmod go=r, u=rw file1.

Masking file permission :- (Umask)

Umask in Unix is actually user mask or user file creation mask.
This is the base permission or default permission given to files or directory is created.
Default Umask value is 0022
First '0' indicate octal number.
Second '0' indicate user can access 3 permissions.
First '2' define the group only denied write permission
Second '2' define the user denied write permission.

* for file, min value is 000 and max value is 666
for directory min value is 000 and max value is 777.

To set the Umask value, use the command '$ Umask value'
If U mask value is not set, then all permission are accessable.
For directory execute permission must be present.

Directory related command :-

mk dir :- It is used to create a directory
Syntax :- mk dir directory name
(i) mk dir -p → It allows to create multiple directory simultaneously.
(ii) mk dir -m → 754 new directory name.

The Unix File System :-

⇒ To know the block size on the file system. We use $ cmchk ↵
1028.

- **Boot Block** :(a.) This represent the beginning of file system it contain a program called bootstrap loader. The program is called boot strap loader.

(b.) It execute when we boot the host machine. Although only one boot block is needed to startup the system all file system contain a boot block.

- **Super Block** :-
Super Block describes the state of file system. The state of file means how large the file is how many maximum files can be accomodate or create.

- **Inode Table** :-

(a) All entities in the OS treated as file. The information related to the files is stored in the Inode table.

(b.) For each file there is an Inode entry in the table.

(c.) Each entry is made of 64 byte and contain information about the files.

(d.) The details are owner of the file, group to which

owner belong type of file, file excess permission, date and time of last access, date and time of last modification, no. of link to the file, size of file, address of block where the file currently present.

- Data Block :-
This contain actual file content. This block cannot be used for storing any other file content unless the file to which it originally belongs it deleted.

- Disk related commands :-
(i.) Main function of system administrator of UNIX installation is efficient hard disk management.
(ii.) Unix file installed on hard disk system admin regularly check the integrity of file system and available of space in the disk.
(iii.) If it is neglect the system will be crash.

- Checking disk free space :-

(i.) If we want to see how much disk is being used there is a command to check i.e. df.
(ii.) It report free space as well as well as remain disk space for file system installed in the machine.

- $df -ivt :- It will show all the values numerically.
- $du :- (disk user) :- The function is similar to df but du shows disk space used by specific file or directory.
- $du /dev :- It shows all the file store in the dev directory.
- $U limit :- (User limit)

(i) most of the file occupies extra bytes in disk space to avoid creation of such files we use ~~limit~~. u limit.

(ii) It implies user cannot create file whose size cannot be exceed than the given value

- rmdir :- By rmdir we can remove directory with -p, rmdir not only remove the specified directory but also its parent directory however rmdir only removes empty directory.

Ex :- rmdir/rmd/ command

(Here, only pwd (present working directory is removed) i.e. command.

rmdir (-p) book/unix/ command

-p is used to remove empty directory and also pwd.

If book, unix is empty directory then only it can be removed.

cd (change directory) :- $ mkdir book
$ cd subject
$ pwd
It will show that you are in current directory i.e. subject.

pwd → When a user log in, it always take the user into the present working directory. The convenient way to find in which directory user is working simply:
$ pwd
/ user / user1
/ denote root directory of unix file within root directory there is a sub directory called user.
Within user there is another sub directory called user1.
We are in user1 (pwd)

A bit of mathematics :- The calculator is called bitwise calculator (bc). It can be invoked by typing bc at the shell prompt we are in calculator mode.
If we will type quit, then again we can return back into the shell.
In bc calculator, BODMAS rule is applied.
In floating point we set the ~~available~~ variable scale to a value equal to number of digit after decimal point we want to display. It support square root ~~since~~ sine, cosine etc. Sine and cosine will work when bc is invoked with

-c.

In trignometry function the argument are in radian and not in degrees.

bc allows setting of variable, the life of variable until you exit the bc.

Expression (expr) :-

Expression command is used to evaluation of expression

expr 100 + 5

105

In floating point, expression command cannot be implemented.

Factors :- In unix when factor is invoked, it will wait for a number to be typed by user if we type in a positive number less than 2^{46}. It will factorize the number and print the factor.

Miscellaneous command :-

$logname :- It will print the username of the system

$id :- Unix know each user not only by logname but also by two number i.e. user or group id number

$date :- It will show the current date.

Piping :- A piping is a chain process where output of one command is input for the another command.

The output of the command in left of the file is send as input of the command in right of the file.

Symbol of piping : |

$ cat Text | wc
(O/P) ↳ It will count the word & character present in text file.

$ ls | wc -l
(It will count the lines present in all files)

I/O Redirection :-

(a.) '>' This symbol implies redirection output and 'c' this symbol implies redirection input

(b.) The symbol '>' redirection output ---- sends the output of a command to a file or device.

(c.) The symbol '<' redirection input takes the input needed for a command from a file rather than keyboard.

(d.) The redirection output tells unix display output on the screen instead put it somewhere else.

(e.) The redirection input says the input for this command is not coming from the keyboard this time it look for someone else.

(f.) $ cat ~~fork~~ file1 file2.
(By executing this command it will return to the prompt $)

(g.) The redirection operator declare the file to as standard output that means the output of cat normally send to screen but here it send to the file to.

(h.) File2 is clear and refilled with new data.
(i.) The unix says that cat read from the standard input if not input file is given.
(j.) First, we type cat then press enter key thus we provided no file as input then what we type was gathered as standard input and cat pass it to standard output.
(k.) $ cat < current file > new file.
The first part < current file indicate input is to be taken from the current file and second part > new file indicate the output is to routed to the new file.

VI Editor :-
(Virtual Editor)
Vi editor usually available in all types of unix system. It requires very few resources. It is more user friendly than other editor such as. ed, ex.

Modes of Operation in vi editor :-

(i) Command mode :- In this mode all keys are pressed by the user are interpreted to be editor command. For eg :- if you hit 'h' the cursor is moved one position to the left.

In command mode the keys are that are hit are not displayed on the screen.

(ii) Insert Mode :- This mode permit insertion of new text, editing of existing text or replacement of existing text.

Each operation can be performed only after changing over from the command mode to insertion mode by using appropriate command. The insertion command is also known as input-text mode.

~~(iii) Ex Command :- This mode~~

(iii) Ex Command mode :- This permits us to give ~~com~~ command at the command line. The bottom line of vi screen is called the command line. Vi uses command line to display message and commands. All commands entered in Ex command mode are displayed in command line. The mode is so called because commands given in this mode are compatible with the command of ex editor.

To create a file when we type vi command and filename. vi clears the screen and display a window in which we can enter and edit the text.

The underscore (_) on the top of the line shows the cursor waiting for the user to enter a command. Every other line is marked with a tilde (~) symbol for an empty line.

Press 'i' key to enter the insert mode of vi editor (don't press enter key).
Now we can add text to the file

Another window will open
Type some text :—

enter key :- It is used to begin a fresh line.

The text that we type goes into the place in a memory known as buffer. When typing is finished, press escape key to leave insert mode and return to command mode. Now we can edit the text which we have created. To save the file, hold down the shift key and press 'Z' twice.
Instead of using Z, Z to save and quit we can also use :w and :q for writing and quitting of the file. Any command begins with ":" is a command to be given in the ex command mode.

'h' - move the cursor 1 character to the left
'j' - move the cursor down one line.
'k' - move the cursor up one line
'l' - move the cursor 1 character to the right

Delete :-
If we want to delete the character move the character to the cursor and press 'x'.
x = delete one character
nx = delete n no. of character.

Overwriting of text :-

If we want to overwrite the existing text, then the 1st position the cursor using h, j, k, l at the character from where you want to overwrite begin.

Next press 'R' which indicate that whatever you type next should overwriting the existing text at the current cursor position.
Press escape to finish when you finish replacing the text.

Quitting vi :-

Having made all changes in the ~~do~~ document, finally save the command then click Z, Z to quit or :wq in ex command mode. If we decide to quit but not save the changes, type :q! in ex command mode and then press enter. This means "leave vi and throw away my changes, I know what I am doing."

Parent and Child Process :-

Each unix process has 2 ID no's assigned to it. The process ID (Pid) & parent process ID (PPID) Each user process in the system has a parent process.

Zombie & Orphan Process :- When a child process is killed, the parent process is updated via SIGCHILD signal. Then the parent do some other ~~work~~ task or restart a new child

as needed. However, sometimes the parent process killed befor its child ~~parent~~ killed. In this case parent of all processes the init processes, becomes the new PPID (Parent process ID). In some case these process are called orphan processes.
When a process is killed, a PS listing may still show the process with a z state. This is a zombie or defunct process. The process is dead not being used. This process different from orphan process.

Daemon Process :- They are system related background process often run with the permission of root and services requests from other process. It waits for something to happen that it capable of working with.
Ex :- Printer Daemon waiting for Print command.

TOP Command :- It is a useful command which shows info about physical and virtual memory, CPU usage, load average
$ top - see the CPU utilization

(Q.) Write in detail about device driver
A. (1) A device driver is a software module that resides within the Digital UNIX kernel and is the software interface to a hardware device or devices.

(ii) A hardware device is a peripheral such as a disk controller, tape controller, or network controller device. In general there is one device device driver for each type of hardware device

Device Driver can be classified as :-

(a) Block device driver
(b) Character device ~~driver~~ driver
(c) Network Device Driver
(d) Pseudodevice Driver.

Date
Page