UNICS 🡺 Uniplexed Information computing system

Unix is open source.

Linus 🡺 Linux

Flavour & Version 🡺

|  |  |  |
| --- | --- | --- |
| Flavour | Version | Version |
| Sun Solaris | 5.8 | 5.9 |
| IBM AIX |  |  |
| HP UX |  |  |
| Linux | Fedora | Ubuntu |

* Unix os is multi tasking , multi user, multi programming operating system.

**Architecture**:

User

Shell

Kernel

Hardware

**kernel**:

* developed in C
* It does cpu management.
* It interacts with i/o and o/p devices.
* Memory management
* Interface between shell and hardware

**shell**:

it interpret the command pass by user. It is called as command line interpreter.

**Types of shell:**

* Bourne shell(sh)
* Bourn again shell (bash)
* c shell
* korn shell (ksh)
* secure shell(ssh)

**PORT:**

* 21-ftp
* 22-ssh
* 23-Telnet

In unix 3 types of objects are there.

* directory
* file
* link

Diagram

Description automatically generated**File system:**

It starts with root(/)

temp :

etc : admin related files

bin : binary executable files

home : user home directory

dev : device related file

usr : user information and bin

lib : library files

var :

**Booting**

* Boot strap loader
* kernel loads into memory
* it called first programme init.
* initab or init programme open active ports /etc/initab
* getty programme starts. It prompts the login. /etc/getty

login successfully and comes to the shell prompt.

$ : default prompt for normal user

# : default prompt for admin user

Unix is case sensitive.

**command : set of instructions.**

**command [option] [argument]**

**Types of command**

**Internal command**: internal commands are inbuilt into shell.

**external command** :external commands are available in some of directories in the system.

$echo : displays the message or display the content of the variable.

$type ==> types of command

type echo

type mkdir

**Shell Variables:**

**echo $SHELL** ==> it will show the shell name

**echo $LOGNAME** ==>

**echo $PATH** ==> it is a system defined or shell variable which contains selected directory separated by colon.

**echo $HOME**

**echo $USER** ==> it will show the user name

**uname ==> displays the OS name**

uname -n shows the version or host name

uname -r version of OS

uname -m microprocessor family

uname -a display all the data

date displays date

**Alias & Function**

**alias h="echo hi"**

h

**function h { echo hello; }**

h

**Exact sequence of unix command**

priority level: alias==>user defind function ==> internal command ==>external command

primary prompt : $

if the command is not complete it will go to secondary prompt. That is >

**ps1 🡺 this will change the primary prompt.**

ps1='\*'

ps2 it will change the secondary prompt.

ps2='\*'

**cal 🡺 It will show the current month calendar**

cal nov 2011

cal 2011

**Absolute and relative path**

**Absolute : starts from root /**

/home/raj/b

/tesco/podata/CL/recv

**Relative path : starts from current directory**

./../raj

**cd 🡺 change directory**

**pwd** 🡺present work directory

cd ./a/d

**cd ~ 🡺** goes to home directory

**cd -** 🡺 it will toggle between two directory

**mkdir**  🡺create a directory

mkdir /home/raj/a

mkdir ./a

mkdir a

**mkdir /home/raj/a /home/raj/a/d**  🡺use space for for multiple directory

**mkdir -v** /home/raj  🡺verbose.it will show the message.

**mkdir -p** /home/raj 🡺preserve mode.It wont give any message.

**mkdir -pv**  🡺it will show message for newly directory

**mkdir -m 753** /home/raj/a 🡺it will create directory with permission

**Date**

date day of the week

date +"%a" day of the week

+"%A" full name of the day

+"%b" month

+"%B" fullname of month

+"%c" date in diffrent format

+"%d" date 04

+"%e" only date 4

+"%p" AM/PM

+"%V" no of week.44th week

+"%Z" zone name

+"%z" time diffrence between GMT

+"%H" only hour

+"%M" only minute

+"%S" only seconds

+"%T" time h:m:s

+"%m" month in number 05

+"%y" year 2 digit. 17

+"%Y" year full digit. 2017

-d yesterday yesterdays date

-d tomorrow tomorrow date

-d "15 Aug 1947" Friday

-d "15 Aug 1947 nextday" saturday

+"%d/%m/%y" 22/05/1989

**ls**

**ls** 🡺list out all object directory, file,link

**ls -1** 🡺display object in one single column

**ls -x 🡺** multi column output

**ls -a 🡺** display objects along with hidden objects . and ..

**ls -A** 🡺display all objects without . and ..

**ls -r** 🡺display in reverse order.

**ls -ar** 🡺display all objects with reverse mode

**ls -l** 🡺display in long listing format

**ls -F** 🡺symbolic format

**ls -R**  🡺 Recursive mode. Display all object along with child object

**ls -d**  🡺displays . always use arguments. without using argumnet shows only (.)

**ls -t**  🡺 displays objects with modification of time ad recently modified time in the top.

**ls -ltu**  🡺display objects respected to used time,accessed time

**ls -S**  🡺size in decreasing order

**ls -n**  🡺same as l as group id

**ls -m**  🡺displays objects separated by (,)

**ls -of**  🡺open files

**/ directory**

**@ link**

**\* executable file**

**long listing format**

* column 1: 10 fields **-rwx\_wx\_\_x** : type of object and permission
* column 2: link
* column 3:username
* column 4:group name
* column 5:size
* column 6: date and time
* column 7: object name

-rwxrwxrwx

-file

d directory

l link

s socket

c charachter device

b block device

**Redirection operator:**

**>**  o/p redirection operator

**<**  input redirector operator

**>>**  append operator

**<<** here document

**>&** associate operator

**cat a.txt 🡺** display the content of file

**cat > a.txt 🡺 i**t creates the file with input mode

**ctrl + d to save and exit**

**cat a.txt>b.txt** 🡺create b.txt. copy a.txt to b.txt. if the file b.txt already exists, it will truncate b.txt first then copy a.txt to it.

**cat a.txt>>b.txt** 🡺it will redirect the a.txt to b.txt without deleting any data.

**cat a.txt b.txt >> c.txt** 🡺a.txt copied to c.txt and then b.txt copied to c.txt

**cat a.txt > a.txt** 🡺create a.txt. if it already exists, truncate a.txt. empty file will created.

**cat -n a.txt** 🡺display the content of files along with the line number and blank line

**cat -b a.txt** 🡺display the content with line number excluding blank line.

**cat nl a.txt** 🡺same as -b

**cat -e a.txt** 🡺append $ at end of every line

* **ctrl z halt**
* **ctrl d save and exit**
* **ctrl c interrupt**

**field descriptor**

* **0 standard i/p device**
* **1 standard o/p device**
* **2 error**

**permission**

2 types:

* absolute
* relative

**Absolute permission: represent the permission in octal format**

r 4

w 2

x 1

directory default value: 755: drwx r\_x r\_x

file default permission: 644: -rw\_r\_\_r\_\_

link default value: 777: @rwxrwxrwx

umask used to set the default permission for file and directory

umask default value : 022

777-022=755

**Relative permission:**

u: user

g: group

o: other

**+ give permission**

**- revoke permission**

**chmod u+x a.txt**

**chmod ugo+x**

**chmod u+x,g+r,o+r**

ugo=a

**chmod a+x**

**chmod +x by default a**

**chmod u-x,g+w,o+r a.txt**

**chmod ug+rwx,o+w**

**chmod u-rw,g+rw,o+x**

chmod -R u+x /raj give permission to all files of raj

mkdir -m 755 /raj create directory with specified permission

**rm and rmdir**

**rmdir** 🡺delete the directory,if it is empty

rm 🡺a.txt delete the file

rm -f a.txt 🡺delete forecfully

rm -r /home/raj 🡺all files of raj

rm -ri /home/raj 🡺interactive

**\* for all**

**? single character**

**[ ] character class**

**! navigation**

**ls \*.txt**  shows all txt files

**ls [0-9]\***  shows all data starting from any number

**ls [a-z]\*2.sh** starting from a to z and ending with 2.sh

ls \*.log[!a-z A-Z 0-9] .logA1 or .logB4

**rm -f \*.[0-9 a-z A-Z][0-9 a-z A-Z][0-9 a-z A-Z] !.log** 🡺 delete all 3 extension files except .log

**CP & MV**

**cp source destination**

**cp b.txt ./../raj** 🡺copy b.txt in raj directory

**cp -r /home/raj /home/raj/b** 🡺copy all files of directory

**cp -i a.txt /home/nihar**  🡺interactive mode

**mv a.txt b.txt** 🡺rename or move the object

**mv /home/a/a.txt /home/b/c.txt**

**cp -p a.txt b.txt 🡺 copy without changing the time stamp**

**cp VS mv**

---

* in case of cp, inode number is different but in case of mv inode is same.
* In case of cp the modification time will be different but in case of mv its same.

**filter |**

**wc a.txt** 🡺 number of lines, words, charcters

filter acts as inter process communication

**cat a.txt|wc -w**

**wc -l a.txt 🡺** number of lines

**wc -w** a.txt 🡺number of words

**wc -c** a.txt 🡺number of character

**head**

**by default top 10 lines**

**head a.txt**  🡺top 10 lines

**head -5 a.txt**  🡺top 5 lines

**head -n3 a.txt**  🡺top 3 line along with line number

**head +5 a.txt**  🡺from 5th line to top

**cat a.txt|head -3 🡺** top 3 lines

**Tail**

last 10 line by default

**tail -3 a.txt** 🡺 last 3 lines

**tail -f a.log**  🡺status of the log file. continues flow

**tail +5 a.txt** 🡺 from 5th line to end end

who ,finger

who displays all users who has log to the system or host

**who -H**  🡺it displas all the users in the header

**who -u**  🡺user details in more column

**who -uH** 🡺with header

**who -u <user>** 🡺 particular user name

who am I it displays the complete details of your login

**w 🡺** it displays all the details who has logged in to the system along details like load average

uptime since when system is not shutdown or how long system is in used. system used time

load average : it displays last 5,10,15 in load in cpu

top all running process with cpu utilisation, memory utilisation. Its like task manager. Load average monitoring

**topas** 🡺same as top works only in AIX.

**prstart** 🡺same as top works in solaris

**finger -l 🡺** finger information like login name, shell name default directory, mail, plan, office address, office phone number

**vi** **editor**

**vi a.txt**

**3 types of mode in vi editor**

* command mode 🡺 **esc key**
* insert mode 🡺 **I, a, I, A**
* colon mode 🡺 **esc :**

**Insert mode:**

i 🡺insert data left to the cursor

a 🡺insert data right to the cursor

I 🡺insert data at beginning of line

A 🡺insert data at end of line

**command mode:**

**j**  🡺move cursor down

**k** 🡺move cursor up

**h** 🡺move cursor left

**l**  🡺move cursor right

**0** 🡺move cursor at beginning of line

**1**  🡺move cursor at beginning of line

**^**  🡺move cursor at beginning of line

**10|**  🡺move cursor to 10th charecter

**20|**  🡺move cursor to 20th character

**G** 🡺move cursor to the last line. begining of last line

**10G**  🡺move cursor to the 10th line

**ZZ**  🡺save and exit the file.(upper case)

**o** 🡺creates a blank space below the cursor and came to insert mode (oh)

**x** 🡺deletes the under cursor character

**w**  🡺moves the cursor to the begining of next word

**e** 🡺moves the cursor to the end of same word

**b** 🡺moves the cursor to the begining of previous word

**dd**  🡺delete the line under cursor

**u** 🡺undo the last change

**U** 🡺undo all changes in the line

**J** 🡺join the line wherever the cursor is

**2J** 🡺join 2 lines

**set nu**  🡺set the line number in a file

**set no nu**  🡺remove line number

**d$** 🡺delete from current cursor position to end of the line

**D** 🡺delete from current cursor position to end of the line

**dG** 🡺deletes from current cursor position to end of file

**d^** 🡺delete current cursor position to begining of line

**.** 🡺 dot is used to execute the previous command

**dH** 🡺delete from current cursor position to begining of the page

**ctrl+f** 🡺used to see the file page by page

**ctrl+b** 🡺scroll the page in backward direction

**yy** 🡺copy the current line

**p** 🡺paste

**3G 3yy 12G p** 🡺 copy from 3rd line to 5th line.paste it after 12th line

**Yw**  🡺 copy the word

**dd**  🡺cut and paste

**r**  🡺 replace under cursor character with single character

**R**  🡺 replace under cursor character with multiple char

**s**  🡺substitute with multiple under cursor character

**(**  🡺move the cursor to the beginning of the sentence

**)**  🡺move the cursor to the end of the sentence

**{**  🡺move to beginning of paragraph

**}**  🡺move to end of paragrah

**searching**

pattern: substring of a string

/Pattern 🡺 search in forward direction from cursor position

/hi

**n**  🡺go to next pattern

**N**  🡺reverse the search.previous pattern

**?**  🡺search the pattern in backward direction

**colon mode:**

we can execute the command in colon mode

**:1,$w test.txt**

**:1,$ w! test.txt** 🡺 forcefully

**:1,5 w a.txt** 🡺copy from one file to another

**:10,15 w a.txt**

**.w >> a.txt** 🡺current cursor to a.txt

**5,10 t 16** 🡺copy 5 to 10 line paste after 16th line

**5,10 t $** 🡺copy 5 to 10th line paste it in end

**5,10 m 16** 🡺 cut 5 to 10 line and paste after 16th line

**e! c.txt**  🡺open different file in vi editor

**1G 5 yy**  🡺copy 1 to 5th line

**e! c.txt**  🡺open c.txt

**4G p**  🡺paste the te copied line

**vi editor to prompt**

**:sh**

**ctrl + d**  🡺return to vi editor

**:!cal**

**:!echo hi**

$vi a.txt b.txt c.txt it opens a.txt. :n b.txt

:n c.txt

**substitution**

**:1,2 s/old/new**

**:1,2 s/hi/bye**  🡺substitute the pattern in first appearance of line

**:1,$ s/hi/bye**  🡺 substitute the pattern from 1st line to end

**:1,5 s/hi/bye/g**  🡺substitute globally. every occurrence

for save we must go colon mode

**:W**  🡺save the file

**:wq**  🡺save and exit

**:q!**  🡺quit forcefully without saving.

**:x**  🡺save and exit

**grep**

global regular expression and pattern

**grep option pattern file\_name**

**grep hi a.txt**  🡺it displays the content those contain hi

**grep -i hi a.txt**  🡺 used to ignore the case

**grep -n hi a.txt**  🡺shows the result along with line umber

**grep "hello hi" a.txt**

**grep -v hi a.txt**  🡺shows the line those dont contain hi

**grep -c hi a.txt**  🡺counts the pattern number as per the first appearance in line. 6 or 7 or number of times pattern present

**grep -o hi a.txt**  🡺show the pattern in a single column

**grep -o hi a.txt|wc -w**  🡺displays the no. of patterns

**grep -e hi -e bye a.txt**  🡺displays multiple pattern search

**grep -w hi a.txt**  🡺search the exact pattern

**grep -v ^ $ a.txt>b.txt**  🡺remove blank line from file a and store it in file b

**grep -l "hi" \***  🡺search all files that contains pattern h

**grep '[0-9]\{10\}' tel.txt**  🡺search the mobile numbers

\{6\} 6 times

**E GREP:**

extended global regular expression and pattern.it supports bre + ERE

+ 1 or more occurrence of previous character

? 0 or 1

| alternat

egrep "[aA]g+[ra][ar]wal" a.txt

egrep "(t|m|n)ina" a.txt 🡺search tina mina nina

**FGREP**

file grep or fixed grep

1c

hello

log

promo

zpq

a35

🡺 search all above patterns in a.txt

* copy all these patterns to a new file
* call it in fgrep

**fgrep -f a.txt hello.txt**

**fgrep -f a.txt < hello.txt**

**Process**

**ps** 🡺it displays all the running process

after login , by typing **$ps** it will give 2 o/p

PID TTY Time CMD

1234 pts/0 00:00:00 bash

5678 pts/0 00:00:00 crontab

pid 🡺process identification number

ppid 🡺 parent process identification number

**ps -f** 🡺full details of process

**ps -a** 🡺shows user defined process

**ps -ef** 🡺shows system defined and user defined process

**ps -Af** 🡺shows system defined and user defined process

**ps -u nihar** 🡺displays the process run by particular user

**ps -l** 🡺displays memory information about the process

**ps -ef|grep java** 🡺shows all java process

**ps -ef|grep -i <defunct>**  🡺displays all zombie process

tty 🡺terminal tele type

? 🡺system defined process

**orphan process:**

* init process takes care of all orphan process
* Those process whose parent process died called as orphan process
* where PPID=1 those are orphan process
* init is the parent of all orphan process
* pid of init is 1

**Zombie process**

* if the child process dies before parent process and still there in process table is called zombie process
* this process having <defunct> is called zombie process
* kernel returns the exact status of child process. Till that time the parent process waits for it through a call

**wait()**

The kernel returns the value 0 or any number

0 successful termination

Any number : unsuccessful

**Foreground and background process**

**ls -l &** 🡺 execute the process in background

ls -l > a.txt &

**jobs** 🡺displays all the back ground process

ctrl+ z 🡺halt the process

ls -l|wc c >a.txt ctrl + z

bg 🡺runs the current halt process in background

fg %3 🡺brings 3rd background job to foreground

fg 🡺last background job to foreground

* **if the session is terminated, all the background jobs will be terminated. In this case use nohup.**

sort a.txt>srt.txt &

**nohup sort a.txt >srt.txt &**

**Nice value**

* nice is inversely proportional to priority of process.
* nice command is used to set the priority of the process

**nice -5 sort a.txt &** 🡺setting priority value for a new process

* **nice value 1-19 range. by default 0**

**renice -15 sort .txt &** 🡺setting priority for a running process

**crontab**

scheduling :

* one time scheduling (at/batch)
* periodic scheduling (crontab)

If you want to do some task in a particular time automatically.

**at : schedule once**

at 10:30 🡺It will go to at prompt

at>ls -l

at>cal 1947 ctrl + d 🡺to save exit

at now+1 hour

at now+1 year

at 10:15 AM,15 Jun 2018

at -l 🡺displays all the jobs which are scheduled using at or batch command

at -r job\_id 🡺delete the jo which you want to delete those are scheduled already.

atrm job\_id1 job\_id2 job\_id3 🡺delete multiple jobs

at -c jobid 🡺details of job

**batch:**

batch < a.sh

* difference between at & batch :
* at: user gives time to schedule the job
* batch: cpu gives the time to execute the file or schedule.

**/etc/at.allow**

**/etc/at.deny** contains usernames those are allowed to schedule the job

**crontab:**

* for periodic scheduling we use crontab
* crontab is a file which contains 6 fields.each fields are separated by space
* if one field having multiple option then options are separated by ,
* cron is a daemon process
* it wake and sleep at every minute wheather there are any jobs to run or execute in crontab.
* /etc/cron.allow contains username those are allowed to schedule
* /etc/cron.deny

**crontab -e** schedule opens crontab file in vi editor.

**Min hour date month day\_of\_the\_week script\_name**

00-59 00-23 1-31 1-12 0-6 a.sh

**Runa a job 7 am to 9 am in every 2 mins**

\*/2 7-9 00 \* \* \* b.sh

**Run a job at 7 to 9.25 with 15 mins interval**

00,15, 7,8 \* \* \* c.ksh

30,45

00,15,25 9 \* \* \* c.sh

**Run a job with 15 mins of interval**

00,15, \* \* \* \* d.ksh

30,45

**Run a job in every 5 seconds**

#!/bin/ksh

while true

do

/home/nihar/backup.ksh

sleep 5

done

**Run a job interval of 5 min between 7:30 to 8**

30/5 7-8 \* \* \* e.ksh

**Run a job in every 2 hour**

\* \*/2 \* \* \* f.sh

**Run a job alternate date 2 pm to 5 pm every 3 min**

\*/3 14-17 \*/2 \* \* g.ksh

**alternate Sunday**

\* \* \* \* 0/2 h.ksh

**1st Sunday of every month**

\* \* 1-7 \* 0 f.ksh

**2nd Saturday**

\* \* 8-14 \* 6 j.ksh

**run a job in crontab in background**

\* \* \* \* \* k.sh &

crontab -l contains username those are allowed to schedule display all scheduled jobs

crontab -r contains username those are allowed to schedule delete the crontab file

* The crontab files are stored in /var/spool/cron/crontab/
* crontab -e -u username Root user will edit the crontab file of a user

**KILL**

**kill -15** 🡺 pid normal kill

**kill -15 2416**

**kill -9 pid**  🡺forcefully kill

**kill -9 2416**

**kill -9 p1 p2 p3** 🡺multiple process kill

**kill -9 $!**  🡺kill last background job

**kill -9 $$**  🡺kill the shell

**kill -0** 🡺 kill processes except login shell

**kill -3** 🡺 thread ump

**Kill** 🡺process without using pid:

**pkill** 🡺orca

**ptree** 🡺display hierarchy of all the process

**pstree**

**fork() & exec()**

fork() is responsible for creating a child process and creates like a clone of parent.It is a system call.

It is responsible for copying all the programme to child and execute it.

**tar**

for archiving/backup we use tar command

a.txt

+

b.txt

+

c.txt >> a.tar

**tar -cvf a.tar a.txt b.txt**  🡺creating tar file

**tar -xvf a.tar**  🡺extract the tar file

**tar -tvf a.tar**  🡺it will display files inside a tree

**tar -xvfz a.tar.gz**  🡺extracted directly

**ZIP:**

it is used to process the file

* **zip🡺unzip .z**
* **gzip🡺gunzip .gz**
* **compress🡺uncompress**

**gzip a.tar**

**gunzip a.tar.gz**

**gzip -d a.tar.gz**  🡺extract the files inside a directory

**gzip -R /home/dir1/nihar**  🡺It will zip all the content of directory

**gunzip -R dir1**

**gzip -l a.tar**  🡺it shows the compression size of the file

**zcat a.tar.gz**  🡺displays all contents of files

**zless**

**zmore a.tar.gz**

**tar -uvf a.tar p.txt m.txt**  🡺add two more files in tar file

**FTP**

file transfer protocol

local machine 🡺 server

**put 🡺 upload**

**get 🡸 download**

ftp servername

user:

password:

**or**

ftp

ftp> open servername

ftp>user:

ftp>passwd:

put a.txt /home/server/dir1

mput a.txt b.txt /home/nihar/2

**by using ftp it comes to server location. For any need in local machine we can use command followed by !**

ftp> !pwd

**put a.txt**  🡺transfer the file to the home directory of the server

**cd q**  🡺changing directory in the remote server

**lcd**  🡺changing directory in local machine

**!pwd**  🡺 in local machine

!ls 🡺in local machine

bye 🡺 exit from ftp

* for downloading from remote ftp to local machine use get
* **for multiple file transfer use mput or mget**
* by default ftp is in asci mode
* asci mode is used to send normal files
* binary mode is used to transfer binary, media files

**FIND**

find <address\_from\_where\_you\_want\_to\_search> search\_criteria\_option

/

**-name**

**-type**

**-user**

**-inum**

**-perm**

**-size**

**-mtime days**

**-atime**

**-ctime**

**-newer**

**-mmin minutes**

**-amin**

**find / -type f -name \*.txt -print** 🡺search all txt files from root

find / -name "ora" -type f -print

find / -type d -name "ora" -print search directories

**searching files having permission**

find . -type f -perm 567 -print

find / -user Raj -type f -perm 666 -print 🡺 for a particular user

find / -type f -inum 12345 -print particular inode number

**search files having size empty**

find / -type f -size 0 -print

**delete the files having size zero**

find / -type f -size 0 -delete

**display all files having size 20mb**

find / -type f -size 20480 c -print

**file greater than 20 mb**

find / -type f -size +20480 c

**files less than 20 mb**

find / -type f -size -20480 c

**search both text and .sh files**

find / -type f -name \*sh -name \*txt -print

find / -type f \(-name \*sh -o \*txt \) -print

**display all files modified 2 days before**

find / -type f -mtime +2

**modified within 2 days**

find / -type f -mtime -2

**modified exact 2 days**

find / -type f -mtime 2

**display files of user raj modifoed within 5 raj**

find / -user Raj -type f -mtime -5

**search all files modified between 5 to 10 days**

find / -type f -mtime +5 -mtime -10 -print

**modified within 2 hour**

find / -type f -mmin -120 -print

find / -type f -amin -120 -print

**one file created name a.txt before 2 days. display all files created after a.txt**

find / -type f -newer a.txt -print

**display files created after 15th jan**

touch -m 201201150000 a.txt

find / -type f -newer a.txt

**display all files created in between 15jan2012 to 15feb2012**

touch -m 201201150000 a.txt

touch -m 201202150000 b.txt

find / -type f \(-newer a.txt -a -newr b.txt\)

**3 terminologies used in find command**

**| xargs**

**- exece**

**- ok**

ok and exec are same but ok is interactive

**search pattern in all those files in entire directory structure**

find / -type f |xargs grep -i "ora"

**by default xargs takes 20 files**

find / -type f |xargs -50 grep "ora" it will take 50 lines from find output

**search all .sh files and give execute permission**

find / -type f -name \*.sh -exec chmod u+x

**search all .sh files and make tar of it**

find / -type f -name \*.sh |xargs tar -cvf a.tar

**search all logfiles older than 5 days and remove it**

find / -type f -name \*.log -mtime +5 |xargs rm -f {}\;

**send all errors to a different file.dont show errors in the result**

find / -type f -name \*.sh -exec cp {} {}.bak\; 2>/dev/null

find / -type f -name \*sh 1>a.txt

2>b.txt

**Associate operator**

>

&

find / -type f -name \*.s 1>a.txt

2>&1 🡺it will copy the error logs to the field descriptor. That means the same files contains o/p a.txt

1>/dev/null 2>&1

**Touch**

create an empty file

it sets the time stamp

touch a.txt if the file does not exist ,it will create time stamp

if it already exists, it changes the time stamp to current date

**touch -m 201201150000 a.txt**  🡺modification time

**touch -a 201201150000 a.txt** 🡺access time

**Sort**

sort is used to arrange or sort the data by default ascending order of alphabet.

**sort a.txt**

a.txt

abc

xyz

234

cdb

30-05-2022

**sort -t(;) -k3 a.txt t= terminator k=key**

k1 k2 k3

abc; 123; xyz

def; 456; sdf

fgh; 567; ghj

ghj; 789; ytk

**sort -t ';' -k2 -r a.txt** 🡺sort in reverse order

sort -n b.txt

substitute ; with |

vi emp.list

:1,$ s/;/|/g