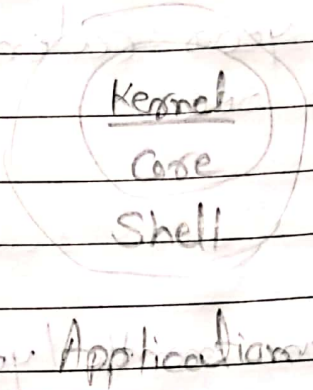
Date
Page No.

Unix

→ Unix developed in 1969 by Ken Thompson.



→ Users interact to system through application program.

→ Whenever User do command to Kernel at that time shell transfer the language to machine language.

→ Unix is a case sensitive. It is multiusers operating system.

✡ Basic Command

1 Show current system date

→ $ date

Sa 18. Dez 19:12:23 CET 2021

Date
Page No.

2 Show current month / years calendars
→ $ cal
Dezembers 2021 (with all date)

3 Show entire years calendars, which years you want to see
→ $ cal 2022
Present all months with dates

4 Show particulars month / years calendars
→ $ cal 5 2019
Mai 2019 (with dates)

5 If we show who are currently using the system / who are currently logged in
→ $ who
It give details like Username, Time, Terminal
Username :- which user is logged in
Time :- when user is logged in
Terminal :- which terminal is user used.

onworks tty7 2019-08-30 19:49 (:0)

6 To create a new directory
→ $ mkdir bca (directory name)

7 Show all files present in current directory
→ $ ls

Date
Page No.

8 From one directory to another directory (How to change the directory)
→ $ cd bca (directory name)

9 How to create another directory inside particular directory
→ bca $ mkdir by sy ty

10 How to create a folder inside another folder
→ $ mkdir by/c

11 How to create a file
→ $ cat > html1.txt (file name)
or cat > by/html/hl.txt

12 How to display content of the file (output)
→ $ cat html1.txt

13 To stop writing content of the file.
→ ^d, ^z (ctrl+d / ctrl+z)

14 How to show root directory.
→ It shows by / (slash)

15 Current directory represent as a . (single dot)
Parrent directory represent as a .. (double dot)

16 Absolute path specify at root directory.
Relative path specify at currently working directory

Date
Page No.

17 To display the message like printf.
→ $ echo "Hello World" (message written in " ")
Hello World

18 Show all possible command (help)
→ $ man cat

19 To read more content of
→ $ man cat | more
pipe sign

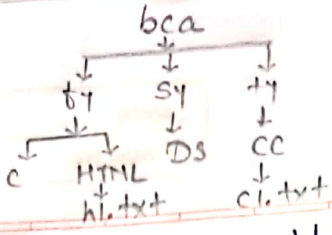
20 To change the password
→ passwd

21 Difference between cat and echo.
→ cat ⇒ display content of the file
echo ⇒ display message

22 To know terminal information
→ $tty
/dev/pts/1

23 Give information about operating system
→ $uname
Linux

24 To know version of operating system
→ $uname -r or uname -n
4.15.0-50-generic



Date
Page No.

25 To copy the file
→ $ cp f1.txt f2.txt
source file name
destination file name

26 When destination file already existed at that time override it. overwrite it. For asking before overwrite.
→ $ cp -i f1.txt f2.txt
source file
destination file

27 Copy multiple files from one directory to another directory.
→ $ cp cho1 cho2 cho3 Sy (directory name)

28 Copy entire directory
→ $ cp -R fy sy
R = Recursive Copy
fy = source copy
sy = destination copy

29 To rename the file
→ $ mv f1.txt f2.txt
source file
destination file
$ mv [also for cut and paste]

same directory = rename
different directory = cut and paste

30 Delete the directory
→ $ rmdir directoryname
if directory empty then removing possible

31 To delete the files
→ $ rm chol (filename)

32 If we delete multiple files
→ $ rm chol chol

33 Delete all the files
→ $ rm *

34 Delete entire structure (file-directory-sub directory)
→ $ rm -rf *
↑ recursive forcefull deleting

35 To show heading
→ $ who -H
↑ H for heading

Name	Line (Terminal)	Time	Comment

36 How many users are currently logged in
→ $ who -q
tybca1 tybca2 tybca3 # users=3

It will show users name in single line & at the end # users count

37 Only show about our own details
→ $ who am i [If users already logged in]

38 To get list of files or directories
→ $ ls

39 To create a file creation editor
→ vi, pico

40 Executable files = green color
Directory files = blue color
Normal files = black / white color

41 More files name are show in column
→ $ ls -x

42 Show type of the file
→ $ ls -Fx [directory will have '/' in end]

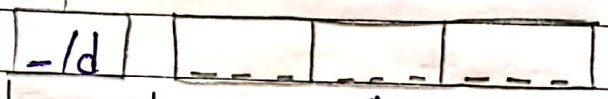
43 Show hidden files or directory
→ $ ls -a

44 How to hidden the file
→ $ cat > .abc.txt
↓
write in starting

45 Long listing / Detail listing
→ $ ls -l

46 Long listing with hidden files
→ $ ls -al

47 - ← ordinary file
d ← directory
n ← network file
s ← system file
} first character (ls -l output)



type of file | Owner | group whom owner belongs | Others (not in group)

r = read
w = write
x = execute
} file permission (9 characters)

d rwx | link count | owner name | group name
① ② ③ ④ ⑤

file size | date & time | file name
⑥ ⑦ ⑧

link count = which file link out with file.
group name = group which owner belongs
file size = represented in bytes (1024)
date & time = where file created

1 row indicate one file.

48 List of files last access time
→ -u

49 Recursive Listing
→ $ ls -R (show entire structure)

50 Reverse Listing
→ $ ls -r

51 Get list of file with name starts with t.
→ $ ls t* * : any no. of character

52 File name with 4 character
→ $ ls t??? ? : represent single character

53 Find having file name
→ $ ls ch[1 2 3]
↑either one should match

54 Which type of file
→ file f1.txt (file name)

55 Basic calculator (To do basic calculation)
→ $ bc ↵
12 * 2 ↵
24
2 ^ 3 ↵
8

56 To show only current date
→ $ date +%d
31

57 To show month number only
→ $ date +%m
12

58 Month name
→ $ date +%h
Dec

59 Entire date
→ $ date +%D (only date without time)
12/31/21

60 System time
→ $ date +%T
09:49:14

61 Combine date and month
→ $ date +"%d %m"
31 12

62 Only system hours
→ $ date +%H
09

63 Only system minutes
→ $ date +%M

64 Only system seconds
→ $date +%.S (capital S)
37

65 Last modification time
→ ls -t

66 Check whether directory exists or not
→ $ ls -d $y (directory name)

* Shell script is basically program. It is group of script.

→ It's extension is .sh.

→ To create shell script we are using vi editor. Do insert mode, command mode, command line mode.

→ To write something press i or a. It means move command mode to insert mode.

→ Come out from insert mode press Esc.

→ :x = save and quit
:wq = write and quit
:q = quit (just quit without saving changes)
:wq! = it override and save the changes
:w = only save

Date
Page No.

* Display Hello World

$ vi first.sh press i
echo "Hello World" Esc
:x

⇒ Execute the command = sh first.sh
file name

$ sh first.sh
Hello World

* Display system date, current date calender.

$ vi date.sh (press i)
date
cal (Press Esc) :x

* Count no. of words, line and characters.

wc ↵ wc = word count
Hello
World
Hi press ctrl+d
3 3 12
↑ ↑ ↑
Line word characters

* wc -l (count of line)
wc -w (count of words)
wc -c (count of characters)

Date
Page No.

* Check the content are same or not.

→ $ cmp (compare)
$ comm (common)
$ diff (difference)

* Check the content

$cat > f1.txt
Hello
World
Fy (ctrl+d) press

$ cat > f2.txt
Hello
Wor
Fy (press ctrl+d)

→ $ cmp f1.txt f2.txt
Output :- f1.txt f2.txt differ: byte 10, line 2

→ $ comm f1.txt f2.txt
f1.txt f2.txt comm
Hello
Wor
Fy
World
Fy
1st part = Word in 1st file not 2nd file
2nd " = Word which are in 2nd file but not in 1st file
3rd " = Word which are common in both file

→ $ diff f1.txt f2.txt
Output :- 2c2
< World

> Wor

* In cmp command if content is same, it will not show output it. give directly command prompt. It show only diff.

* Condition

i, a, o = to insert
~ = not usuable
a = next characters
o = cursers above line
O = cursers below line
dd = delete entire line

* Unix 2 variable
Local Variable
Environment Variable
  - HOME
  - SHELL
  - PS1
  - USER
  - PWD

Access values of variables
$HOME

$ HONE = Home directory
$ SHELL = shell you are using
$ PS1 = Primary Command Prompt (C$ sign)
$ USER = Current User Detail
$ PWD = Command, also environment variable
↓
Current working directory (command)
$ PATH = show path where command is store

→ You do not specify any data type. It get by own.

→ $ is used when fetch the value or use the value.

→ read is work like 'scanf'.
read x (variable name)

Ex. vi read.sh
echo "Enter value of x"
read x
echo "Value of x : $x"

Output :-
sh read.sh
Enter value of x
9
Value of x : 9

> -gt
< -lt
>= -ge
<= -le
== -eq
!= -nq

* How to write it.

```
echo "Enter x" read x echo "Enter y" read y
if test $x -gt $y
or
if [ $x -gt $y ] or if [ $x -gt $y ]; then
then
    echo "$x is maximum"
else
    echo "$y is maximum"
fi Esc, :x
```

Output:- sh max.sh
Enter value of x
10
Enter value of y
20
20 is maximum

→ else if written as `elif`

Date
Page No.

How to write elif.

```
vi mrks.sh ↵ Press i
mrks=75
if [ $mrks -gt 80 ]
then
    echo "Grade A"
elif [ $mrks -gt 60 ]
then
    echo "Grade B"
elif [ $mrks -gt 40 ]
then
    echo "Grade C"
else
    echo "Grade F"
fi  Esc  :x
```

Output :- sh mrks.sh
Grade B

chmod ⇒ to change the file permission

change mode
u = user
o = others
g = group
a = all (user, other, group)

permission
r, w, x

→ To show the permissions.
ls -l

+ give permission
- remove permission

1 Execute permission to users
→ chmod u+x test.txt

2 Execute permission to group
→ chmod g+wx test.txt

3 Remove from group
→ chmod g-wx test.txt

4 Write to all
→ chmod a+w test.txt

5 Execute to all
→ chmod a+x test.txt

6 Users and Group do read, write, execute
→ chmod ug+rwx test.txt

→ You can change file permission which are created by your own.
You cannot change file permission which are created by other.

→ Permissions are overwritten.

* Other way to give permission is binary value convert then change into 3 bit octal values.

Date
Page No.

0 = 000
1 = 001
2 = 010
3 = 011
4 = 100

1 rw- value = 6
110
to give permission :- chmod 110 test.txt

2 rwx value = 7
111

3 rwx r-x r--
111 101 100
chmod 754 test.txt

4 rw- r-- r--
110 100 100
chmod 644 test.txt

5 rw- --- x ---
110 001 000
chmod 610 test.txt

6 rw- --x -wx
110 001 011
chmod 613 test.txt

Date
Page No.

→ Go to next line in shell file
press j

→ Go to above line in shell file
press k

→ Go to end of the line in shell file
press b

→ Delete entire line
press dd

→ script = group of instructions

→ ` = command substitution

* ; "echo current working directory `pwd`"
run file
`pwd` = whatever is written in "pwd" ' '
that has to be executed first then put
output of the command

file manage system = all thing considers as
file

→ :wq!, :x! = Quit

Date
Page No.

expr command

→ We cannot perform Arithmetic operation directly so we can use expr command.
expr = expression evalution

→ expr $x * $y are not work so we can write
expr $x * $y

Ex. x=10
echo $x
10
y=20
echo $y
20
expr $x + $y
30
expr $x * $y
200

? = one character
* = multiple characters

1 vi max.sh press i

```
echo "Enter two numbers"
read no1
read no2

if [ $no1 -gt $no2 ]
```

Date
Page No.

```
then
    echo "$no1 is maximum"
else
    echo "$no2 is maximum"
fi press esc, :x
```

Output :- sh max.sh
Enter two numbers
9
4
9 is maximum

2 Find even or odd

vi evenodd.sh press i

```
echo "Enter Numbers"
read no
rem = `expr $no.%.2`

if test $rem -eq 0; then
    echo "$no is even"
else
    echo "$no is odd"
fi press esc, :x
```

Output :- sh evenodd.sh
Enter Numbers
9
9 is odd

* Whenever assign = sign don't put space between variable.

and = -a
or = -o
not = -n

3. Show if value between 10 and 20 then good otherwise poor.

vi check.sh press i
echo "Enter Numbers"
read x

if [$x -ge 10 -a $x -le 20]
then
    echo "Good"
else
    echo "Poor"
fi press esc, :wq

Output :- sh check.sh
Enter Numbers
19
Good

4. Write a shell script to find maximum out of 3 numbers using and and without using cmd.
vi Max.sh
with using and (comment)

echo "Enter 3 Numbers"
read no1
read no2
read no3

```
if [ $no1 -ge $no2 -a $no1 -ge $no3 ]
then
    echo "$no1 is max"
elif [ $no2 -ge $no1 -a $no2 -ge $no3 ]
then
    echo "$no2 is max"
else
    echo "$no3 is max"
fi
```

Output :- sh Max.sh
Enter 3 Numbers
10
20
15
20 is max

```
# without using and
vi maximum.sh
echo "Enter 3 Numbers"
read no1
read no2
read no3
```

} or read no1 no2 no3

```
if [ $no1 -gt $no2 ]
then
    if [ $no1 -gt $no3 ]
    then
        echo "$no1 is max"
    else
        echo "$no3 is max"
```

```
        fi
else
        if [ $no2 -gt $no3 ]
        then
            echo "$no2 is max"
        else
            echo "$no3 is max"
        fi
fi
```

Output :- sh maximum.sh
Enter 3 Numbers
12 34 56
56 is max

✱ While Loop

→ While loop start with do, end with done.

1. `#i=`expr $i+1` or i=$(($i+1))`

```
i=1
while [ $i -le 5 ]
do
    echo "$i"
    i=`expr $i +1` (increamate the value using expr)
done
```

Output :- 1
2
3
4
5

Date
Page No.

2. While loop

```
i=10
while [ $i -le 20 ]
do
   echo "$i"
   i=`expr $i+1`  or  i=$((i+1))
done
```

☆ For Loop

Syntax:- for var in values
1 2 3 4 5

```
for var in 1 2 3 4 5
do
   echo $var
done
# values are separated by spaces. You also
  give values like a b c d e
```

starting value

```
for i in `seq 1 10`   <- ending value
do
   echo $i
done
# seq command generate sequence
# without ` ` it only considers as value
# For creating range you need to write
  within ` `
# for loop use for list of files.
```

Date
Page No.

* for i in {1..10} # not workable
do
  echo $i
done

Output :- {1..10}

* Display content of file in to cmd.

echo "List of files in my current working directory"
ls -l # for long listing
ls for list

* Only show txt files.
ls *.txt

* $ shell :- what environment variable you use

* Display content of the *.txt files with using for loop.

for f in *.txt
do
  echo $f
  cat $f
done
take one one file and display its content

Date
Page No.

* Display the details of that file with using for loop.

```
for f in *.txt
do
    echo ls -l $f
done
```

Output :-

```
ls -l f1.txt
ls - l f2.txt
ls - l test.txt
```

* Only txt file executable

```
for f in *.txt
do
    chmod 766 $f
    ls -l $f
done
```

Output :-

```
rwx rw- rw-
rw x rw - rw -
rw x rw - rw -
```

* How to write switch case

```
echo "1 - Date In 2 - Calendar In 3 - Current
      Working Directory"
```

```
read ch
case $ch in
        1) date ;;
        2) cal ;;
        3) pwd ;;
        *) echo "invalid" ;;
esac
```

→ Whenever \n not work at that time using -e like echo -e "1-Date"

→ case separated by ;;
;; is end of the case

→) are compulsory in switch case and after) put one space is must.

→ * represented as a default in switch case

☆ Write a shell script to display good morning, good afternoon and good evening depending on system date.

```
vi gm.sh press i
hr=`date +%H`
echo $hr

if [ $hr -lt 12 ]
then
        echo "Good Morning"
elif [ $hr -lt 16 ]
then
        echo "Good Afternoon"
```

```
elif [ $hr -lt 20 ]
then
        echo "Good Evening"
else
        echo "Good Night"
fi
```

Output:- sh gm.sh

13

Good Afternoon

OR

```
d = $(date +%H)

if [ $d -gt 6 -a $d -le 12 ]
then
        echo "Good Morning"
elif [ $d -gt 12 -a $d -le 16 ]
then
        echo "Good Afternoon"
elif [ $d -gt 16 -a $d -le 20 ]
then
        echo "Good Evening"
else
        echo "Good Night"
fi
```

Date
Page No.

* Make a Menu driven program with choice 0 or 1 it should display currently logged in users, choice 2 operating system name, 3 display list of txt files, 4 calendar of current years, Any other choice it should display invalid.

```
echo "Press 0 - Currently logged in users\n1 - Currently logged in users\n2 - Operating system name\n3 - List of txt files\n4 - Calendars of current years"

read ch
case $ch in
        0|1) who;;
        2) uname;;
        3) ls *.txt;;
        4) cal 2022;;
        *) echo "Invalid Choice"
esac
```

pipe sign

1 = pipe sign use for 0 and 1

Date
Page No.

☆ Command Line Argument $0 variable

```
vi cmdarg.sh
echo "$0"
echo "$First Argument - $1"
echo "Second Argument - $2"
echo "List of Arguments - $*"
echo "List of Arguments - $@"
echo "Total no. of Arguments - $#"
```

command line argument ←

Output :- sh cmdarg.sh hello world how are you

```
cmdarg.sh
First Argument - hello
Second Argument - world
List of Arguments - hello world how are you
List of Arguments - hello world how are you
Total no. of Arguments - 5
```

or

```
for i in $*  [Iterative through for loop]
do
    echo $i
done
```

Output :- sh cmdarg.sh hello world how are you

```
hello
world
how
are
you
```

Date
Page No.

Make f.txt

* Display the content of the file whose name are given command line argument, if argument not given display proper message.

dis.sh

```
if [ $# -le 0 ]
then
        echo "Insufficient Arguments"
else
        for f in $*
        do
            cat $f
        done
fi
```

Output :- sh dis.sh f.txt cmdarg.sh

* head :- show content of the file from the top of the file

* tail :- show content of the file from end of the file

* head -n5 f.txt

```
hello
hi
how
are
you
```

Date
Page No.

* head $.txt
Output :-
hello
hi
how
are
you
world
www
uname
who

* tail -n5 $.txt
Output :-
world
www
uname
who

* tail -n2 $.txt
Output :-
who

* tail -3 $.txt
uname
who

Date
Page No.

* Show only line no 3 and 4.
(particular no. to end of file)

→ tail +5 f.txt
you
world
www
uname
who

+5 :: it means start with starting line to till end

* head -n4 f.txt | tail +3 (this is input so don't add file name)
how
are

↑ Pipe sign

| = pipe sign = Output of the previous cmd input for next cmd

redirection operation :- >, >> used for appending data in existing file
> = repeated
>> = append

cat > f1.txt
hello

cat f1.txt
hello

cat >> f1.txt
hi
world

command cannot change the content of the file

Date
Page No.

```
cat f1.txt
hello
hi
world
```

* head -n4 f.txt | tail -2
```
how
are
```

* Total file count
ls -l | wc -l
8
pipe sign

* Total line of the file
wc -l < f.txt
10

Jumber
chardharay
AA Jana
Makhra

repeated & append
redirection operation :- >, >> used for appending data in existing

| → ~~Input~~ Output of the previous cmd input ~~for~~ for next cmd

head -n4 f.txt | tail +3 (this is input so don't add file name)

total file count
ls -l | wc -l
8

sort the content of the file
sort f.txt

29/1/'22

vi emp.txt
pipe sign = separator
1 abc | g.m. | sales | 12/12/20 | 6000

copy line in vi editor
Y, P - paste

sort emp.txt
s

Sorting numeric value
→ sort -n emp.txt

Sorting according disegnitation
→ sort -t "|" -k 3 emp.txt
which column you want to sort
(for particular column)

→ Same name then sort according designation
sort -t "|" -k 2,2 -k 4,4 emp.txt
(delimenator; start, end)

consecutive column -k 2,3
(start, end)

→ sort -t "|" -k 2,2 -k 4,4 -k6,6 emp.txt

12000 = String wise + first

* Reverse Sort
sort -t "|" -k 2 -r emp.txt

* Sorting on years
sort -t "|" -k 5.7,8 -n emp.txt
(colum no., start char, end char)

→ You use whatever sign in separator.

* ~~Display unique~~
Check file is sorted or not
sort -c emp.txt

sort f1.txt | sort -c

Special variable in cmd line argu -
echo $?
which show status of previous cmd.

failure ⇒ 1
not failure ⇒ 0

Use
remove of file compare it is same or not

sort -u fl.txt (unique values)

uniq fl.txt (Not use each & every shell script)

sort -o fl sort.txt fl.txt
ls ↑ sorted
storing the output into a file

this file made automatically.

tr = transmitting

No changes in existing file content; change in sort, uniq, tr, head, tail. display

tr 'a' 'A'
to change the value

tr '[a-z]' '[A-Z]' < emp.txt
↑ ↑
group of characters from the file

* convert 1 to ~
tr '[a-z]1' '[A-Z]~' < emp.txt

* remove some characters
tr -d '1' < emp.txt

* only put 1
tr -cd '1' emp.txt

remove space

```
tr -s ' ' < emp.txt
```

squeezing = multiple characters

Date 12/2/'22 Page No.

Unix

cut, paste fetching data vertical (column)

emp.txt

emp-id	name	designation	DOJ	Sal
1	ABC	Clerk	12/05/03	7000
2	PQR	Director	9/06/01	12000

Expect only emp-id and name use cut

① field, column

paste - put the file

character as column emp

1st column, 2nd, 3rd

id = 1st field, name - 2nd field

-c : character cutting

-f : cut by field

cut entire field do spe seperator - -d

cut -c 1-7, 9-14 emp.txt

emp-id	name
1	ABC
2	PQR

save cut value

emp.txt > t1.txt

↑

redirection operator ⇒ >, >>, <, |

data from the file : <

data store into the file : >

" " " " override the data : >>

< : read > store data

screen ← | → file output

VISION Date Page No.

cut -d ":" -f 3 emp.txt
save output in the file and see it on screen
tee f1.txt

who → show 5 column output
who | cut -d " " -f 1

* Find the total no. of users.
* in built directories.
/bin
/etc = configuration file
/dev = devices
/lib = library files
/user/bin = all command store in this
/etc/passwd = 1 entry per user, password = encrypted format

→ cut /etc/passwd | wc -l

grep = Global Regular Expression (MIMP)
awk = Aaward
grep → use (search) pattern into file.

grep "date" *.sh
Lines which is containing date in output

How many clerk
grep "clerk" emp.txt

→ count
grep "clerk" emp.txt | wc -l
tee (to show output)

-v = invers

* different

ignoring the case
grep -i "clerk" emp.txt | wc -l

→ Data of employee who are not clerk
grep -v "clerk" emp.txt

→ Line no. along with data
grep -n "clerk" emp.txt

→ counting
grep -c "clerk" emp.txt

→ list of file names which have that pattern
grep -l "date" ~~emp.txt~~ *.sh

→ spelling mistake
grep -e 'clerk' -e 'clark' -e 'celrk' emp.txt

-e = use search multiple pattern

[le] = either l or e then match
regular expression
grep 'c[le][eal]rk' emp.txt

→ any no. of character and any character
grep 'c[le][eal].*' emp.txt
↑
0 ~~and~~ or any no. of characters

→ 2 characters after 3 character
grep 'c[le][eal]..' emp.txt
↓
only 2

^ = Starting of the line

→ Row which has 1st characters e.
grep '^e' emp.txt

→ Find a line which has last character l.
grep 'l$' emp.txt
$ - means last

blank line = grep '^$' emp.txt

→ Find a line word 'hello
grep '^hello$' emp.txt

→ last 3rd characters s and then any character
grep 's..$' emp.txt

→ Find out the line which has not 1st characters e.
grep '^[^e]' emp.txt
not

grep -v '^e' emp.txt

* Write a shell script to get list of currently logged in users whose name starts with T

* Write a shell script to display the content of the file which are having pattern hello.

* Write a shell script which display the employee who are getting salary in range of 7000.

1. who | grep '^t'

grep -l "^hello$" *.sh

2. for f in `grep -l "hello" *`
do
    cat $f
done

3. grep '7...$' emp.txt

awk = particular search like clerk or managers.

syntax:
awk option 'selection.criteria {action}' files
awk '/clerk/{print}' emp.txt
between 2 slace (/) write searching
→ only name of user not whole record
awk '/clerk/{print $2}' emp.txt

→ awk '/clerk/{print $2 $5}' emp.txt

divide the fields.
awk -F "|" '/clerk/{print $2 $5}' emp.txt
↑
separator

→ Find out record of clerk and director.
awk -F "|" '$3 == "director" || $3 == "clerk" {print}'
↑
and
emp.txt

→ ① Display employee name whose salary greater than 7000.

nw = new variable

~~awk '{print}'~~

awk -F "!" '$5 > 7500 {print $2}' emp.txt

awk -F "!" 'NR==2, NR==3 {print NR $2, $5}' emp.txt

NR = record no, line no.

* Write a shell script to change the extension of all txt file to ~~exe~~ dat

```
for f in *.txt
do
    nw = `basename $f txt`
    mv $f ${nw}dat
done
```

basename command store ~~only~~ after txt ~~name~~ fl.

* if basement doesn't work

```
nw = `echo $f | cut -d "." -f1`
mv $f ${nw}.dat
```

* Write a shell script that read 2 file name check whether that exist or not. If both the files exist append the 2nd file content to 1st.

* Write a shell script that count total no. of zero size file in current working directory.

* Write a shell script to find total no. of directories in current working directory.

* Write a shell script that accept 2 directory names from command line. If insufficient arguments are given display error message. If source directory doesn't exists give error. If destination directory doesn't exists create it. If both the directory exists copy source directory content to destination.

* Write a shell script to display permissions of particular file.

- ls -l

- echo "sh name.sh f1.txt f2.txt"

```
echo " sh name.sh f1.txt f2.txt"
read " sh name.sh f1.txt f2.txt"
if [ sh name.sh ] ; then
    echo " Insufficient arguments"
elif [ sh name.sh $ f2.txt ]
then
    echo " Create source directory "
elif [ sh name.sh f1.txt ]
then
    echo " Create destination directory "
    cat f1.txt
elif [ sh name.sh f1.txt f2.txt ]
then
    copy f1.txt f2.txt
fi
```

1.
```
echo "Enter two file names"
read f1 f2
if [! -f $f1 ]
then
    echo "$f1 does not exists"
elif [! -f $f2]
then
    echo "$f2 does not exists"
else
    cat $f1 >> $f2
fi
```

or

```
if [ -f $f1 -a -f $f2 ]
then
    cat $f1 >> $f2
else
    echo "file not exists"
fi
```

2.
```
cnt=0
for f in *
do
    if [ ! -s $f ]
    then
        cnt=`expr $cnt +1`
    fi
done
echo "Total no. of zero size files $cnt"
```

Shell Scaipt

16/2/'22

* Comprase the file
  gunzip
  gzip

* Write a shell scaipt that display all sub directories in current working directory.

* Write a shell scaipt that display home directory, date, current working directory and logged in users.

VISION
Date | Page No.

* Write a shell script that display directory information as file name, size, date, protection, owner.

* Write a shell script which checked whether your friend is currently logged in or not.

1. cd..
   -type d directory / while wild card character "*"
   in double quotes ("*.txt")
2. cd
   $ date
   $ cal
   $ who
   find use searching the files.
   ./ current directory and all sub directory
   find ./ -name "f1.txt"
   find . -type d -name "f4"
3.
1
```
for f in *
do
   if [ -d $f ]
   then
      echo "$f"
   fi
done
```

2
```
echo " 1. home directory \n 2. date \n 3. current working directory \n 4. Currently logged in users"
read ch
case:
   1) for f in *
      do
         if [-d $f]
         then
            echo "$f"
      fi
```

```
done;;
2) $ date;;
3) $ pwd;;
4) $ who;;
*) echo " invalid choice";;
esac
```

3 protection - permission

```
ls -l | cut -d " " -f1 > prot.txt
ls -l | cut -d " " -f7 > filenm.txt
                         owner.txt
paste -d" " filenm.txt prot.txt owner.txt
```

4
```
read nm
who | grep "$nm" > /dev/null
if [ $? -eq 0 ]
then
    echo "$nm is currently logged in"
fi
```