**LINUX / UNIX**

Ctrl + d [or] exit -to close terminal window

Ctrl + l [or] clear -to clear terminal

Note: All new creating user is created under home by default.

users -it will show current user

cat /etc/passwd (or) getent passwd -it will list all users

sudo useradd username -creating newuser &and group also created automatically with same name.

sudo passwd username -to change password for the user.

sudo userdel username -to delete user

NOTE:- when we try to delete user .it will left home directory (it didn’t delete home directory)

Sudo userdel -r username -to remove home directory&user

--------------------------------------------------------------------------------------------------------------------------------

**su -Trying to switch into root with password**

su root -Trying to switch into root with password

**su username -switching from one user to another user**

su - username -switching from one user to another user

**sudo su forcefully switching into root directory & exit**

sudo su - forcefully switching into root directory & exit

sudo su root forcefully switching into root directory & exit

------------------------------------------------------------------------------------------------------------------------------------

groups -it shows groups for current user

groups username -it shows groups for a user

cat /etc/group (or) getent group -it lists all groups

sudo useradd -g groupname username -create a new user and assign a group in on command

sudo useradd -g group1,group2,group3 username -create a new user and assign a group in on command

sudo groupadd groupname -to add new group.

getent groupname -it shows all groups for a user

sudo usermod -a -g group user1 user2 -adding user2 in user1 group(adding user to the group)

sudo usermod -a -g group1,group2,group3 user1 -add a user to multiple groups.

groups username -it will shows all groups to that user

sudo gpasswd -d username groupname - to remove the user from group.

sudo usermod -g groupname username -change user primary group

sudo groupdel groupname -to delete group

getent group -it list all groups

**NOTE**:- we can’t delete primary group. To delete that group assign another group to that user as a primary group and delete this group.

The below commands represents about the user information.

whoami -print effective user id

id -un -print effective user id

uname -print system information

id -print real and effective user and group id’s

id -u -print only the effective user value/number

**Package Management :**

Linux maintaining repository called “APT” .From that repository we can get all packages.

sudo apt-get update 🡺 To update the repository.

sudo apt-get install package\_name 🡺 To install the package

sudo apt-get remove package\_name 🡺 To remove the package

**To check the directories in tree diagram :**

Syantax : tree dirname

**How to Create Directories :**

mkdir dir0 -it will creates a empty directory

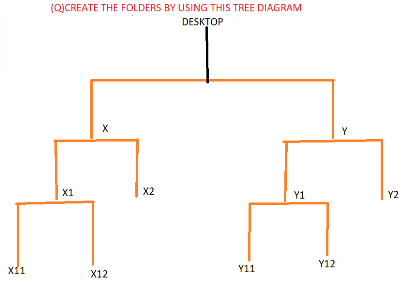
mkdir .dir1 -it will creates a hidden directories

mkdir dir1 dir2 dir3 dir4 -it will creates multiple directories at a time

mkdir dir{10..20} -it will create multiple folders at a time

mkdir -p pav1/pav2/pav3 -it will create folder in folder

**EXAMPLE 1:-**



mkdir -p x y x/x1 x/x2 x/x1/x11 x/x1/x12 y/y1 y/y2 y/y1/y11 y/y1/y12

EXAMPLE 2:-

mkdir -p heroins/{samantha,kajal,anushka,rakhul}/{jan,feb,mar,apr,may,jun,jul,aug,sep,oct,nov,dec}\_{2020,2021,2022}/schedule\_{1..31}.txt

**How to Create Empty Files :**

touch file1.txt - to create the empty file

touch .file1.txt - to create the hidden empty file

touch dir0/file.txt -without entering into folder creating textfile in that folder

touch dir{10..20}/file.txt -without entering into folders creating textfile in each folder

touch dir{10..20}/file{1..10}.txt -without entering into folders creating multiple files in each folder

Note:

We can create 2 filenames with same name but timestamp will be different.

**How to Remove Files / Directories :**

rmdir dir1 -it will remove only empty directory

rm -r dir1 -it will remove non-empty directory

rm -r \* -it will remove all directories/files (empty/non-empty)

rm file.txt -to remove file

rm -r dir1/file.txt -to remove file in particular folder.

rm \* -it will remove all text files at a time.

**How to Change directories :**

cd dir1 -to enter into directory

cd .. (or) cd - -it will goes back to one folder

cd ../../.. –it will goes back to 3 folders

cd ~ -it will directly goes to home

cd path -it will goes to that particular path

-------------------------------------------------------------------------------------------------------------------------------

**Long listing Files :**

ls -it shows all files/folders

ls -r -it shows all files in reverse order

ls -l -it shows all files/folders with permissions,date,time stamp

ls -a (or) ll -it shows all files along with hidden files

ls -l-a (or) ls -la (or) ls -al

ls -l-a-t (or) ls -lat (or) ls -alt -new to old

ls -l-a-t-r (or) ls -latr (or) ls -altr -old to new

ls –R -it shows directories & sub-directories

ls -latR

ls -lat dir1/ -it shows files/folders in particular folder

ls -f -to hide colors

ls -F -display all files with colors and ends with /\*

**How to Create Files :**

cat file.txt -to view contents of a file using cat

[or]

cat < file.txt -to view contents of a file using cat

cat dir1/file1.txt -without entering into folder how to see data

cat file1.txt file2.txt file3.txt -to display multiple files using cat

cat -n file1.txt -to display contents of file with line numbers

cat > file4.txt -how to create file& writing using cat command.ctrl + d

cat file1.txt > file3.txt -copy the contents of one file to another file

cat file1.txt >> file2.txt -it will copy & append the data

cat file1.txt file2.txt file3.txt >> file4.txt -it will copy from multiple files &append the data

cat -n file1.txt -to get line numbers

cat -b file.txt -to get line numbers but does’t count blank spaces

cat -E file1.txt -how to make cat highlight line-ends

cat -s file1.txt -how to make cat suppress repeated empty lines

cat -T file1.txt -how to make cat display tab charecters as ^I

cat -v file1.txt -how to make cat display non printing charecters.

NOTE:- suppose you created a file with some content using cat command and again you are creating the file with the same name using touch command .then data will be there only timestamp will be updated.

**How to Copy Move /Rename files**

cp file1.txt file2.txt -copying content from file1 to file2

mv pavan.txt kumar.txt -renaming the file name from pavan to kumar

mv file1.txt folder1 ---moving file from one folder to another folder

mv file1.txt file2.txt folder1 ---moving file from multiple files to another folder

mv \*.txt folder1 -----moving file from multiple folders to another folder

scp source destination ---->How to copy file

ex:-

scp file1 /home/ubuntu/file2 --->copying file and also remaing filename

scp file1 ubuntu@ipaddress:/tmp/file2 -->copying file from one env to another env

> file\_name defaults to success output

1>file\_name success output

2>file\_name error output

1>file\_name 2>file\_name both success & error files will store into a single file

1>file\_name 2>&1 save error file where the success file saved into a single file

&>file\_name both success & error files will store into a single file

**CALENDER COMMANDS :**

cal -it shows current day,date,month,year

cal 1 -it list all months from 1st year

cal 2021 -it list all months from 2021 year

cal 8 2021 -it list 8th month of that year

cal -3 -it list previous,current,next months

date --date “2years ago” -it shows day ,month,time from 2 years back

**DATE COMMANDS :**

date = Both date & time

date + %D = MM/DD/YY

date + %T = Time

date + %A = MONDAY

date + %B = JANUARY

date + %Y = 2020

date + %a = MON

date + %b = JAN

date + %y = 20 (year value)

date + %m = 05 (Month value)

date + %d = 06 (Day value)

date + %H = 24 Hours scale

date + %I = 12 Hours scale

date + %M = Minutes

date + %S = Seconds

To display date in our own format:

date +%d-%m-%y = 06-05-20

date + %Y-%b-%d = 2020-apr-06

Create an empty file where filename contains current system date?

Ex1: file1\_24\_04\_2020.log

Ans:-

touch file1\_$(date +%d\_%m\_%Y\_%H\_%M\_%S).log

Ex2: file1 24 04 2020.txt

Ans:-

touch file1 $(date +%d %m %Y %H %M %S).txt

**How to Grant /Retrieve permissions :**

**Numerical Notations:**

read = 4

write = 2

execute = 1

**Ex:**

ls -l sample.txt

---------- 1 pavankumar pavankumar 68 Nov 16 09:42 sample.txt

chmod 600 sample.txt

-rw------- 1 pavankumar pavankumar 68 Nov 16 09:42 sample.txt

chmod 660 sample.txt

-rw-rw---- 1 pavankumar pavankumar 68 Nov 16 09:42 sample.txt

chmod 666 sample.txt

-rw-rw-rw- 1 pavankumar pavankumar 68 Nov 16 09:42 sample.txt

chmod 6 sample.txt

-------rw- 1 pavankumar pavankumar 68 Nov 16 09:42 sample.txt

chmod 66 sample.txt

----rw-rw- 1 pavankumar pavankumar 68 Nov 16 09:42 sample.txt

chmod 666 sample.txt

-rw-rw-rw- 1 pavankumar pavankumar 68 Nov 16 09:42 sample.txt

Note:

If you give single digit it will give permissions to others first.

If you give double digit it will give permissions to groups and then others

**Alphabetical Notations:**

Owner 🡺 u

Group 🡺 g

Others 🡺 o

+ 🡺 To grant permission

- 🡺 To Retrieve the permission

Ex: Remove the execution permission to the owner and add the write permission to the others to the file f3.

chmod u-x,o+w f3

EX: Add the read permission to the others to the file f3

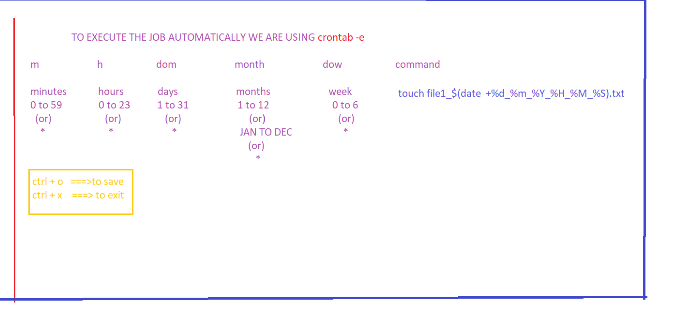
chmod o+r f3

EX: Remove the read permission to the others to the file f3

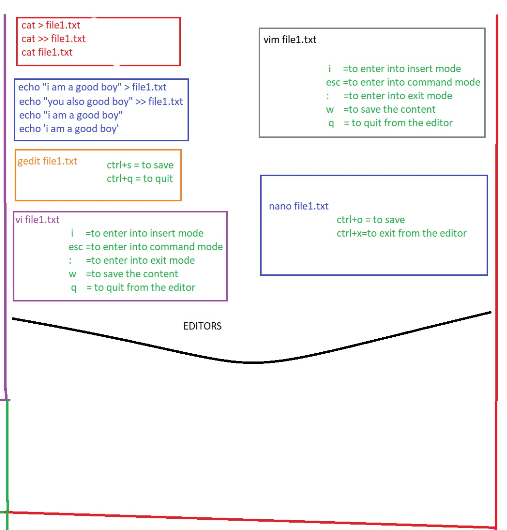
chmod o-r f3

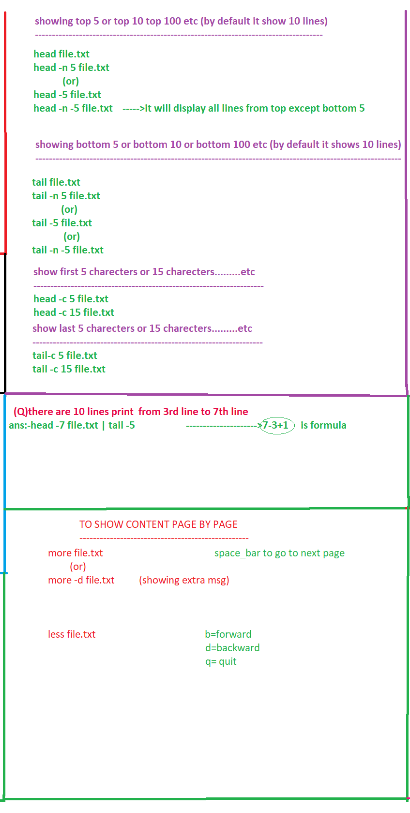
EX: Add the read & write permission to the others to the file f3

chmod o+wr f3



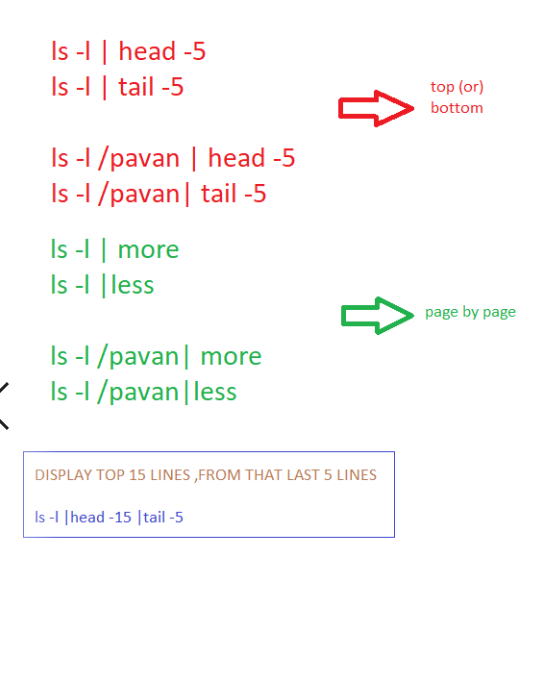
What are the different types of editors ? ctrl + d-🡪to come out from cat file.



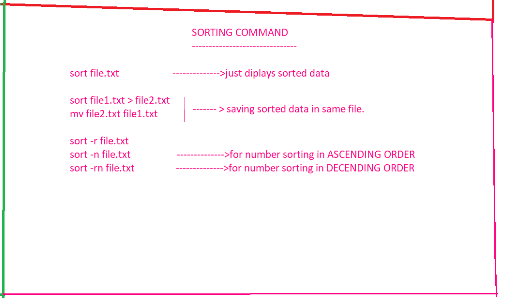


If listing is very huge then how can we see only particular number of line?

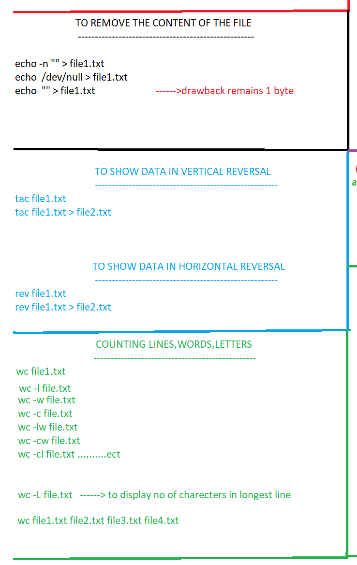
Ans:-



------------------------------------------------------------------------------------------------------------------------------------------



cat filename.txt >/dev/null ----to nullify the txt file.



Man commandname -it shows more commands related to that & Q

Whatis commandname -it shows the one line description about that command.

**SECTION: 4**

**grep**:- cowinv ABC rlh

you can give path or filename

grep “above” text.txt

grep “above” text1.txt text2.txt text3.txt

grep “above” \* -it print string with whole line & with filename

grep -c “above” text.txt -it counts the word.

grep -o “above” text.txt -it prints only the word.it does not print lines.

grep -w “above” text.txt -it will extract excat word.

grep -i “above” text.txt -to ignore the case sensitivity

grep -n “above” text.txt -it prints string with whole line with line number

grep -v “above” text.txt -to display the lines which are not having giving string/text

grep -A 2 “above” text.txt -to display 2 lines after match

grep -B 2 “above” text.txt -to display 2 lines before match

grep -C 2 “above” text.txt -to display 2 lines before and after match

grep -r “above” \* -to search string in current directory and sub directory.

grep -l “above” \* -it display only the filenames ,if the word is matched

grep -h “above” \* -it will display matching content with hiding file names.

grep -nh “above” \*

grep -cw “above” \*

ADV grep:- feE

grep -f filename text.txt -in that filename string should be there before search.

grep –e “line” –e “above” –e “python” text.txt -to search multiple strings at a time.

grep –E “line|above|python” text.txt -to search multiple strings at a time.

RULES TO CREATE PATTERN:

grep –E “line|above|python” text.txt -it search for strings like (line or above or python).

Grep –E “^line” text.txt -matches for the lines which are starting with “line”.

Grep –E “line^” text.txt -matches for the lines which are ending with “line”.

Grep –E “^$” text.txt -matches for the lines which are empty

Grep –E “\^” text.txt -matches for the lines which are having special charecters

Grep –E “\$” text.txt -matches for the lines which are having special charecters

Grep –E “\.” text.txt -matches for the lines which are having special charecters

Grep –E “t..s” text.txt -matches anyone character.

Grep –E “line\b” text.txt -it will print the lines, which are having space at edge of that string.

Grep –E “\bline” text.txt -it will print the lines, which are having space at before of that string.

Grep –E “\bline\b” text.txt -it will print the lines, which are having space at before\edge of that string.

[or]

Grep -Ew “line” text.txt -it will print the lines, which are having space at before\edge of that string.

Grep –E “yf?” text.txt -the f may be 0 times or 1 times .

Grep –E “yf\*” text.txt -the f may be 0 times or more times.

Grep –E “yf+” text.txt -the f should min 1 times or max more times.

Grep –E “[tTfy]” text.txt -it search for single charecters like (t or T or f or y)

Grep –E “t|T||f|y” text.txt -it search for single charecters like (t or T or f or y)

Grep –E “[a..d]” text.txt -it search for single charecters like (a or b or c or d)

Grep –E “[a..dp-r]” text.txt -it search for single charecters like (a or b or c or d or p or q or r).

Grep –E “^[xy]” text.txt -it search for single charecters like (x or y) from starting line

Grep –E “[^xy]” text.txt -it search for single charecters other than (x or y) .

Grep –E “xf” text.txt -it search for single charecters like (x or f) .

Grep –E “xffff” text.txt -it search for single charecters like (x or ffff) .

Grep –E “xf{4}” text.txt -it search for single charecters like (x or f 4times) .

Grep –E “xf{3,4}” text.txt -it search for single charecters like (x or f 3 or 4 times).

Grep –E “xf{2,}” text.txt -it search for single charecters like (x or f 2 times min but max any no.of times).

Grep –E “xf{3,4}\b” text.txt -it search for single charecters like (x or f 3 or 4 times there must be space then only it prints).

Grep –E “[[:digit:]]” text.txt -it will print the lines which are having digits

Grep –E “[[:upper:]]” text.txt -it will print the lines which are having upper case

Grep –E “[[:lower:]]” text.txt -it will print the lines which are having lower case

Grep –E “[[:alnum:]]” text.txt -alphanumeric charecters

Grep –E “[[:alpha:]]” text.txt -alphabetic charecters

Grep –E “[[:blank:]]” text.txt -blank charecters space and tab

Grep –E “[[:space:]]” text.txt -space charecters :tab,newline,vertical tab,form feed,carriage return,and space.

(1)Write a pattern to match only directories?

Ls -lrt |grep –E “^d”

(2)Write a pattern to match only files?

Ls -lrt |grep –E “^-”

(3)find the servers ipv4 info from a file? FYU 4 parts untai each part lo 1 or 2 or 3 digits untai less than 1 untey Danni ipv4 anam?

Cat servers\_info.txt | grep -E “\b[0-9][0-9][0-9]\.[0-9][0-9][0-9]\.[0-9][0-9][0-9]\.[0-9][0-9][0-9]\b”

[or]

Cat servers\_info.txt | grep -E “\b[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\b”

------------------------------------------------------------------------------------------------------------------------------------

CUT:-

**-f,-sf,-d ------------🡪are used for the field separators**

**Based on charecters:-**

cut -c 1 file1.txt 🡪it prints 1st letter

cut -b 1 file1.txt 🡪it prints 1st letter

cut -c 4 file1.txt 🡪it prints 4th letter

cut -c 4,9 file1.txt 🡪it prints 4th & 9th letters

cut -c 4-9 file1.txt 🡪it prints 4 to 9 letters

cut -c 4-9,12 file1.txt 🡪 it prints 4 to 9 & 12th letters

cut -c -10 file1.txt 🡪 it prints from 1st character to 10th charecter

cut -c 10- file1.txt 🡪 it prints from 10th character upto last.

**Based on fields:**

**[1] If we use -f then Tab as a by default delemeter,if we use other than the tab it considered as a single field**

one two three

four five six

seven eight nine

Cut –f 2 file2.txt it prints 2nd column

Cut –f 1,3 file2.txt it prints 1st & 3rd columns

Cut –f -1 file2.txt it prints from 1st field to last filed

Cut –f 2- file2.txt it prints from 2st field to last filed

Cut –f -2 file2.txt it prints from 1st field & 2nd filed

Cut --f 2 file2.txt it print other than 2nd filed

**[2] For suppose if the file consists delimiters as tab & space then we need to use ‘s’**

one two three

four five six

seven eight nine

pavan is a good boy he got job

he went to hospital

he went to movies

Cut -sf 1 file3.txt it skip space data.

**one**

**four**

**seven**

Cut –d ’ ‘ -sf 1 file3.txt it skips tab data.

**pavan**

**he**

**he**

-----------------------------------------------------------------------------------------------------------------------------------

**[3] For suppose if the fields are separated with other than tab**

ubuntu@ip-172-31-36-28:~/practise$ cat file3.txt

root:x:0:0:root:/root:/bin/bash

daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin

bin:x:2:2:bin:/bin:/usr/sbin/nologin

sys:x:3:3:sys:/dev:/usr/sbin/nologin

sync:x:4:65534:sync:/bin:/bin/sync

cut -f 1 file3.txt 🡪it is the wrong way hence those are separated with “: “so we need to use -d ‘:’

cut -d ‘:’ -f 1 file3.txt 🡪it is correct way.it prints 1st field good practise

cut -d ‘:’ -f 1,4 file3.txt 🡪it prints 1st & 4th fields

**root:0**

**daemon:1**

**bin:2**

**sys:3**

cut -d ‘:’ -f 1,4 --output-delimiter=” “ file3.txt 🡪it prints 1st & 4th fields [our expected delimiter]

**root 0**

**daemon 1**

**bin 2**

**sys 3**

**……………………………………………………………………………………………………**

**[4] PRINT ONLY 2.4.41**

server version: Apache/2.4.41()

server built: oct 22 2019 22:59:042

cat file4.txt | cut -d '/' -f 2 file4.txt | cut –d ‘ ‘ –f 1 file4.txt

**2.4.41**

**server built: oct 22 2019 22:59:04**  ---unwanted line also printed to void that use grep for better o/p

**MOSTLY WE ARE USING in this way**

cat file4.txt | grep -E "Apache" | cut -d '/' -f 2 | cut -d ' ' -f 1 -----good practise

**2.4.41**

**Storing entire script in one variable**

a=$(cat file4.txt | grep -E "Apache" | cut -d '/' -f 2 | cut -d ' ' -f 1)

echo “$a”

**SECTION: 6**

AWK :- [ AWK =grep + cut ]

**-F ------------🡪is used for the field separators**

NOTE: By default it consider space as a filed separator. Tab will also consider as a space.

awk '{print}' file3.txt

onetwothree

pavan is a good boy hegot job.

he went to hoslppital

he wnt to movies

[or]Both will fetch entire data

awk '{print $0}' file3.txt

onetwothree

pavan is a good boy hegot job.

he went to hoslppital

he wnt to movies

one two three

pavan is a good boy hegot job.

he went to hoslppital

he wnt to movies

[1] print only the 1st filed

awk '{ print $1}' file3.txt

one

pavan

he

he

print only the 1st & 2nd fileds

awk '{print $1,$2}' file3.txt

one two

pavan is

he went

he wnt

we can reverse the 1st & 2nd fileds

awk '{print $2,$1}' file3.txt

two one

is pavan

went he

wnt he

To prints 1st & 2nd fields [our expected delimiter]

awk 'BEGIN {OFS=" \_"} {print $1,$2}' file3.txt

one \_two

pavan \_is

he \_went

he \_wnt

to print only the line/record/row numbers of the file

awk '{print NR}' file3.txt

1

2

3

4

to print only the line numbers & data of the file

awk '{print NR,$0}' file3.txt

1 one two three

2 pavan is a good boy hegot job.

3 he went to hoslppital

4 he wnt to movies

to print only the line numbers, data& the total no of fiels at end of the file

awk '{print NR,$0,NF}' file3.txt

1 one two three 3

2 pavan is a good boy hegot job. 7

3 he went to hoslppital 4

4 he wnt to movies 4

To print last field from the file

awk '{print NR,$NF}' file3.txt 🡪NF is replacing with 3,7,4,4.hence $3,$7,$4,$4

1 three

2 job.

3 hoslppital

4 movies

To print 1st record from 1st field ,2nd record from 2nd field ………..

awk '{print $NR}' file3.txt

one

is

to

movies

server version: Apache/2.4.41 ()

server built: oct 22 2019 22:59:04

[2] print only the first line

cat file4.txt |awk '/version/{print}' --searching for the line which has version word

server version: Apache/2.4.41 ()

[3] print only the 1st field --at a time searching (grep) & getting (cut) is possible.

cat file4.txt |awk '/version/{print $1}'

server

[4] print only the 2nd field

cat file4.txt |awk '/version/{print $2}'

version:

[5] print only the 3rd field

cat file4.txt |awk '/version/{print $3}'

Apache/2.4.41

[6] print only the 4th field

cat file4.txt |awk '/version/{print $4}'

()

**[Q] PRINT ONLY 2.4.41**

server version: Apache/2.4.41 ()

server built: oct 22 2019 22:59:04

cat file4.txt | awk -F '[ /]' '/version/ {print $4}' -- using space or / as a field separator. PATTERN

2.4.41

cat file4.txt | awk -F '[ /]' ' NR==1 {print $4}' -- using CONDITION

2.4.41

Explanation:

In awk command we can more than 1 character for filed separating.

In the above example we are using space or / as a field separators.

**DIFFERENCES BETWEEN CUT & AWK:**

In cut we can use only 1 character as a field separator at a time

In awk we can use use more than 1 character as a field separator at a time

Using cut command we can fetch charecters & fields

Using awk we can fetch only fields.

In cut command ,By default it consider Tab as a filed separator

In awk command ,By default it consider Space as a filed separator. Tab will also consider as a space.

------------------------------------------------------------------------------------------------------------------------------------------

**SECTION: 7**

tr (translate) :-

tr is useful to translate or delete given set of charecters from the input.

cat xyz.txt

Line 1

Line 2

Line 3

Line 4

Line 5

tr [:upper:] [:lower:] <xyz.txt

line 1

line 2

line 3

line 4

line 5

tr [:digit:] [3-7] <xyz.txt

Line 3

Line 4

Line 5

Line 6

Line 7

tr 'i' 'I' <xyz.txt

LIne 1

LIne 2

LIne 3

LIne 4

LIne 5

tr 'i' '\_' <xyz.txt

L\_ne 1

L\_ne 2

L\_ne 3

L\_ne 4

L\_ne 5

tr '4' '5' <xyz.txt

Line 1

Line 2

Line 3

Line 5

Line 5

tr ' ' '\_' <xyz.txt

Line\_1

Line\_2

Line\_3

Line\_4

Line\_5

cat xyz.txt | tr ' ' '\_'

Line\_1

Line\_2

Line\_3

Line\_4

Line\_5

-d -------🡪simply it deletes given set of charecters.

TO PRINT ONLY THE 19.03.12

cat docker.txt | cut -d " " -f 3

19.03.12, …...here we got extra “,” we need to remove.

cat docker.txt | cut -d " " -f 3 | tr -d ","

19.03.12

tee :-

tee command is used to display the o/p and also to store the o/p into a file. (It does both the tasks simultaneously).

It is useful to create logs for shell scripting.

Display the data and also save the data at a time.

cat file3.txt > output.txt ---it will save only the data it will not display

cat file3.txt | tee output1.txt ---it will display the data and saves the data.

onetwothree

pavan is a good boy hegot job.

he went to hoslppital

he wnt to movies

cat file3.txt | tee -a output1.txt ----it will display,saves & append the data.

onetwothree

pavan is a good boy hegot job.

he went to hoslppital

he wnt to movies

--------------------------------------------------------------------------------------------------------------------------------------

**SECTION: 8**

**Basics of Bash Shell Scripting :-**

echo command is used to display a string/message or variable value or command result.

|  |
| --- |
| Syntax:  echo string/message  echo ‘string/message’  echo “string/message” |

Three types we can execute script:

|  |
| --- |
| Syntax:  ./filename.sh  Bash filename.sh  /path/filename.sh |

Simple shell script Helloworld.sh

#!/bin/bash

echo "Hellow world"

Using Shebang we can specify the interpretor(command)which is responsible to execute the script.

Shebang line is nothing but which shell we are using to execute our script.

Afte writing script we need to give execute permissions to user.

**Intoduction to variables :**

Variables are useful to store the data in shell scripts and Later we can use them if they required.

Default value of a variable is “Empty/Nothing”

How to store the strings/values into a variable

Simple syntax:

x=value

p=string

q=’string’

r=”string”

How to print the strings/variables using echo

Simple syntax:

* echo message/string
* echo “message/string”
* echo “message/string with variable$xyz”
* echo “message/string/$variable/$(command)”

ex:-

echo $x 🡪it prints value 4 right

echo ‘$x’ 🡪it prints string($x) wrong

echo “$x” 🡪it prints value 4 right

set 🡪it displays script in which all system variables are defined.

All system variables are defined in upper case letter.

EX:-

Echo $USER

Echo $ BASH 🡪it displays bash path

Rules To Define User Defined Variables

1. Variable name should contain only a-z or A-Z, 0-9 and \_charecters.
2. Variable names length should be lessthan or equal to 20 charecters.
3. Variable names are case sensitive .means x and X are different.
4. Don’t provide space on either sides of equal symbol.
5. Use quotes for the data if data consists of spaces.
6. We can store the o/p of command into a variable as follows.

variableName=$(command)

variableName=`command`

1. We can assign one variable value/data into another

Name=”Shell Scripting”

NewNamel=$Name [or] NewName=${Name}

8.No need to declare variable data type.

EXAMPLE 1: about\_variables.sh

#!/bin/bash

s="Bash shell scripting"

echo "welcome to $s"

echo "$s is powerful in Linux Env"

echo "Now we are working with variables concept of $s"

EXAMPLE 2: practise\_on\_variables.sh

#!/bin/bash

dockerVersion=$(cat docker.txt|awk '/Version/ {print $3}'|tr -d ",")

mvnVersion=$(mvn -version|awk '/Apache/ {print $3}')

date=$(date)

echo "Docker version is : $dockerVersion"

echo "Maven Vesion is : $mvnVersion"

echo "Today date is :$date"

**Advanced Usage of echo command:**

Defining a string variable

Simple syntax:

x=value

p=string

q=’string’

r=”string”

s=$(date)

Displaying the string variable value using echo

Simple syntax:

* echo message/string
* echo “message/string”
* echo “message/string with variable$xyz”
* echo ${variable}
* echo “message/string/$variable/$(command)”

Advanced usage (to execute escape charecters)

* echo -e “string or varaible”
* Escape charecters:
* \n New Line
* \t Horizontal Tab
* \v Vertical Tab
* \b Backspace
* \r Carriage Return
* \ Escape character
* To display message in colors.
* echo -n “message/string/$variable/$(command)” 🡪it will append the data from 2 lines to 1 line

EXAMPLE 1: advanced\_echo.sh

#!/bin/bash

echo "this is the first line"

echo "this is the second line"

[or]

#!/bin/bash

echo -e "this is the first line\nthis is the second line"

more examples:

echo -e "this is the first line\tthis is the second line"

echo -e "this is the first line\vthis is the second line"

echo -e "this is the first line\bthis is the second line"

echo -e "this is the first line\rthis is the second line"

To print as it is:

echo -e “This is about \”bash\” scripting” [or] echo “This is about \”bash\” scripting”

echo -e "this is the first line\\\nthis is the second line" [or] echo "this is the first line\nthis is the second line"

echo -e "this is the first line\\\tthis is the second line" [or] echo "this is the first line\tthis is the second line"

echo -e "this is the first line\\\vthis is the second line" [or] echo "this is the first line\vthis is the second line"

echo -e "this is the first line\\\bthis is the second line" [or] echo "this is the first line\bthis is the second line"

echo -e "this is the first line\\\rthis is the second line" [or] echo "this is the first line\rthis is the second line"

To Display message in colors:

#Reset

Color\_Off=’\033[0m’ #Text Reset

#Regular Colors

Black=’\033[0;30m’ #Black

Red=’\033[0;31m’ #Red

Green=’\033[0;32m’ #Green

Yellow=’\033[0;33m’ #Yellow

Blue=’\033[0;34m’ #Blue

Purple=’\033[0;35m’ #Purple

Cyan=’\033[0;36m’ #Cyan

White=’\033[0;37m’ #White

#Bold

BBlack=’\033[1;30m’ #Black

BRed=’\033[1;31m’ #Red

BGreen=’\033[1;32m’ #Green

BYellow=’\033[1;33m’ #Yellow

Blue=’\033[1;34m’ #Blue

Burple=’\033[1;35m’ #Purple

Cyan=’\033[1;36m’ #Cyan

White=’\033[1;37m’ #White

More examples:

echo -e "\033[0;32mHi my name is pavan kumar\033[0m"

Hi my name is pavan kumar

echo -e "\033[1;32mHi my name is pavan kumar\033[0m"

Hi my name is pavan kumar

EXAMPLE 2: advanced\_echo1.sh

#!/bin/bash

echo -n "this is the first line"

echo "this is the second line"

this is the first linethis is the second line

**MULTILINE BLOCK:**

Heredoc is very useful to write multi-lines or multiline block.

|  |
| --- |
| Syntax:  Cat << DELIMMITER  Line-1  Line-2  Line-3  DELIMITER |

Note: Here “DELIMITER” can be any string.

Heredoc is mostly used with the combination of cat command.

We can also redirect this heredoc result into a file or as a input for another command

* Cat << DELIMMITER > Demo.txt

Line-1

Line-2

Line-3

DELIMITER

* Cat << DELIMMITER | grep Line-1

Line-1

Line-2

Line-3

DELIMITER

EXAMPLE 1: usageOfHereDoc.sh

#!/bin/bash

cat << EOF

The user is : $USER

The home for thsi $USER is : $HOME

EOF

The user is : ubuntu

The home for thsi ubuntu is : /home/ubuntu

EXAMPLE 2: usageOfHereDoc.sh

#!/bin/bash

cat << Pavan

The user is : $USER

The home for thsi $USER is : $HOME

Pavan

The user is : ubuntu

The home for thsi ubuntu is : /home/ubuntu

**Here String : (Reducing echo word)**

Here String is like here document only but with one line (no need to write echo command)

|  |
| --- |
| Syntax:  Command <<< string |

EXAMPLE 1: in terms of string

echo "Hellow this is pavankumar" | tr [a-z] [A-Z] 🡪with echo command

[or]

tr [a-z] [A-Z] <<< "Hellow this is pavankumar" 🡪without echo command

EXAMPLE 2: in terms of Variable

name="Bash Shell Scripting"

echo “$name” |tr [a-z] [A-z]

[or]

tr [a-z] [A-z] <<<$name

EXAMPLE 3: in terms of command

mvn -version |tr [a-z] [A-Z]

[or]

tr [a-z] [A-Z] <<< $(mvn -version)

**Comments for Bash Shell Scripting:**

* A comment is a human-readable explanation that is written in the shell script.
* Why we need comments:
  + Adding comments to Bash Shell Scripts will save you a lot of time ad effort when you look at your code in future.
  + Comments are used to explain the code.
  + The comments also helps other developers and system administrators who may need to maintain the script to understand your code and its purpose.
* Here, we have two types of comments. They are:
  + Single line comments.
  + Multi-line comments

NOTE: Comments wont execute while running or executing your script.

|  |
| --- |
| Syntax: 3ways we can write  #Line-1  <<DELIMMITER  Line-2  Line-3  Line-3  DELIMMITER  : ‘  Line-4  Line-5  Line-6  ‘ |

EXAMPLE 1: comments.sh

#!/bin/bash

#Author: pavan

<<MYCOM

Created: may-2021

Version: 1.0

purpose: comments

MYCOM

: '

This is about inventory

and its works for linux systems

'

echo "This is a script"

This is a script

**How to make Bash Shell Scripts as Portal with Unix/Linux Systems:**

What is Shebang line?

Shebang line means : which shell we are using to execute our shell scripts.

For Ubuntu it is working good

#!/bin/bash

echo "Hi checking for portability"

For RHEL it Doesn’t work

#!/bin/bash

echo "Hi checking for portability"

This is due to portability issue. Bash is in different path

For Ubuntu 🡪/bin/bash

For RHEL 🡪/usr/bin/bash

To avoid that we are using which env in both “UBUNTU OS “ & “RHEL OS” and taking that path.

For Ubuntu it is working good

#!/usr/bin/env bash

echo "Hi checking for portability"

For RHEL it is working good

#!/usr/bin/env bash

echo "Hi checking for portability"

So, we can make our bash shell scripts as portal using: **#!/usr/bin/env bash**

**Debugging Bash Shell Scripts:**

* Debugging is determining the cause which fails the Script.
* Why scripts fails?
  + Because of some errors.
* This is because of two type of errors.
  + Syntax Errors (Before executing we will get)
  + Runtime Errors (while executing we will get)
* Syntax Errors stops script execution and run time errors don’t stop script execution.
* There are different commands for debugging and we will work with set command.
* Our bash is an interpretor. it reads scripts line by line. It will execute line by line.

We can go with set command and we have different options with set command.

|  |
| --- |
| Syntax:  set 🡪to list system defined variables.  set [options]  set -n 🡪No Execution, Purely for syntax check  set -x 🡪Prints the command before executing it in script  set –e 🡪Exit script if any command fails |

EXAMPLE 1: Debug\_script.sh

#!/usr/bin/env bash

set -n ---it we use this command, first it will fetch the errors.

pwd

date

user=$(whoami)

echo "This is about debugging a bash shell script"

echo "The user is :$user"

**EXIT STATUS OF A COMMAND:**

Each Linux command returns a status when it is executed.

We can display the exist status of a command with **echo $?**

**0**  🡪**command was success without any errors.**

**(1-255 values) 🡪 A non-zero exit status means command was failure**

**SECTION: 9**

**BASIC OPERATIONS ON STRING:**

Defining a string variable

|  |
| --- |
| Simple syntax:  x=value  p=string  q=’string’  r=”string”  s=$(date) -------🡪commands user defined variable |

Displaying the string variable value

|  |
| --- |
| Simple syntax:   * echo message/string * echo “message/string” * echo “message/string with variable$xyz” * echo ${variable}   echo “message/string/$variable/$(command)” |

Finding The length of the string

* echo ${#variable}

Concatination of Strings

X=pavan

Y=kumar

echo $X$Y

[or]

xyResult=$x$y **🡪storing command into variable**

echo “$xyResult”

**Convert Strings into lower/upper case**

echo ${X^^} [or] echo “${X^^}”

[or]

xU=${x^^} 🡪storing command into variable

echo $xU

----------------------------------------------

echo ${Y,,} [or] echo “${Y,,}”

[or]

yL=${Y,,} 🡪storing command into variable

echo $yL

**---------------------------------------------------**

**Replacing the part of the string using variable**

echo ${X/pavan/sunil} [or] echo “${X/pavan/sunil}”

[or]

newX=${X/pavan/sunil}

echo $newX

**otherwise we can use sed command**

|  |
| --- |
| echo $X|sed ‘s/pavan/sunil/’ |

**Slicing The String/sub-string**

|  |
| --- |
| ${variableName:startPosition} 🡪it will print from positions what u gave 0,1,2……..etc |

Name="pavan kumar"

echo ${Name:2}

van kumar

|  |
| --- |
| ${variableName:startPosition:length} 🡪it will print from positions what u gave upto that length |

echo ${Name:2:7}

van kum

**STRING Operations on Paths**

**1.realpath :**

|  |
| --- |
| Syntax:  realpath filename.txt/foldername |

It will gives the complete path i.e it append the current path with filename/foldername.

No need of giving pwd command and writing filename/foldername manually.

Note:

But it don’t validate the path.

realpath demo.sh

/home/ubuntu/practise/demo.sh

**2.basename :**

* it Strips directory information.
* It Strips suffixes from file names.

basename /home/ubuntu/practise/demo.sh

demo.sh

basename /home/ubuntu/practise/demo.sh .sh

demo

**3.dirname :**

It will delete any suffixes beginning with the last slash character and return the result.

dirname /home/ubuntu/practise/demo.sh

/home/ubuntu/practice

**SECTION: 10**

**Input and output commands for shell script :**

Giving input command through keyboard .

Example 1: input\_output.sh

#!/usr/bin/bash

read -p "Enter your name : " x

y=${x^^}

echo "your word in uppercase is : $y"

./input\_output.sh

Enter your name : i am king of the europe

your word in uppercase is : I AM KING OF THE EUROPE

Note:

If you don’t give any variablenames then by default variable name is “REPLY”

Example 2: input\_output.sh

#!/usr/bin/bash

read -p "Enter your name : "

y=${REPLY^^}

echo "your word in uppercase is : $y"

./input\_output.sh

Enter your name : i am king of the europe

your word in uppercase is : I AM KING OF THE EUROPE

INTERVIEW QUESTIONS:

(1)What is the default variable for read command?

Ans: REPLY

**Input with Command Line /Positional Arguments : (while running script we need to pass arguments )**

Example 1: cmd\_line\_args.sh

#!/usr/bin/bash

echo "${0}"

echo "{1}"

echo "${2}"

echo "${3}"

echo "${10}"

./cmd\_line\_args.sh pavan sunil kumar kiran a d g j h f g t

./cmd\_line\_args.sh

{1}

sunil

kumar

f

Example 2: cmd\_line\_args.sh

#!/usr/bin/bash

echo " The number of command line arguments are : $#"

echo "The command line arguments are : $@"

echo "The command line arguments are : $\* "

./cmd\_line\_args.sh pavan kumar 1 5 67 666 gfr5 mvjhgchjcjh 88900098

The number of command line arguments are : 9

The command line arguments are : pavan kumar 1 5 67 666 gfr5 mvjhgchjcjh 88900098

The command line arguments are : pavan kumar 1 5 67 666 gfr5 mvjhgchjcjh 88900098