1. iCal
2. Man date

Scroll down see and to exit q

1. Cal
2. Cal 8 2018
3. Cal 2018
4. date
5. -sh-3.2$ date "+%d %m %y"
6. -sh-3.2$ date "+%d %m %y %T %R"
7. -sh-3.2$ date "+%A %d %m %y %T %R"
8. --sh-3.2$ nl employees.dat
9. -sh-3.2$ nl -v3 employees.dat ( line no starts with 3)
10. nl -v3 -i2 employees.dat ( listing begins with 3 jncrements by 2)
11. -sh-3.2$ nl -v3 -i2 -w4 employees.dat (width of 4 from starting point..shifts left )
12. Tty
13. Who (all users)
14. -sh-3.2$ who am i ( my name)
15. Sdf

Chapter 2

1. pwd ( current directory)
2. cd .. (goes one level up)

pwd

/home

1. cd ..

pwd ( goes to root directory i.e. /)

/

1. ls ( list of directories)
2. cd dev ( go to dev directory)

ls

1. Cd .. ( goes one level up that is root directory)

Cd home ( goes to home directory)

Cd testuser1 ( goes to testuser1 directory)

Pwd

1. Ls -l ( list files .. those starting with – are regular files with .sh )
2. Cd project ( blue is directory
3. Cd / ( root directory)
4. Logname
5. Ls ( lists the directories)
6. Ls -f ( will mark directories with / and executables with \*)
7. Ls -r (reverse)
8. Ls -l (longlisting)
9. Ls -d ( contents inside the directory too)
10. Ls -a ( hiden files )
11. Ls -R ( subdirectories and files)
12. Ls -I ( files with inode no)
13. Ls -s ( blocks/space used by file)
14. ls -l test1.txt ( full file details in 7 columns)
15. cat abc.txt( display file contents
16. cat abc.txt xyz.txt ( display contents of 2 files)
17. cat > friends ( create file named friends. whatever is written in cat is sent to friends)

Roseline

Alphy

Mary

ancy

ctrl d (to save file )

1. Cat < abc ( same as cat abc… from file to cat)
2. -cat <abc > abc1 ( take input from abc and output it to file abc1)

-cat abc1 (display abc1)..output is in file abc1 and not on monitor)

1. cat <abc >>abc1 ( append abc contents to abc1)

- cat abc1

1. cat abc > pqr >2errfile.txt (file xyz will be written to pqr)

cat xyz > pqr >2errfile.txt ( file xyz does not exist.Write err mesaage to 2errfile.txt)

1. -sh-3.2$ cp friends friends1 ( friends file cp -i friends friends1
2. cp -i friends friends1 ( for interactive)
3. cp -r project project1( copying directory project)

ls project\*

1. rm friends1 ( file is deleted/removed)

cat friends

1. rm -i friends( interactive removal)
2. -sh-3.2$ cp friends friends1

cp friends friends2

cp friends2 friends3

ls friends\*

mv friend2 friends4 ( friend2 gets renamed to friends4 as they are in same directory/location)

mv friends1 friends3

1. -sh-3.2$ ls friend\* ( display all files in the current directory)

friends friends1 friends3 friends4

-mkdir schooldir ( create directory schooldir

mv friend\* schooldir ( to move files starting with friend\* one directory to another

ls schooldir

friends friends1 friends3 friends4

1. cat employees.dat

wc employees.dat ( no.of lines, characters, words)

1. wc -l employees.dat ( no.of lines)
2. wc -c employees.dat ( no.of chars)
3. wc -w employees.dat ( no.of words)
4. -wc employees.dat( no.of words and chacters)
5. wc cfile1.txt cfile2.txt
6. wc (enter)

Welcome to shell scripting (ctrl D)

1. cat file1.txt

cat file2.txt

cmp file1.txt file2.txt (compares files)

1. cat cfile1.txt

cat cfile1.txt

comm cfile1.txt cfile2.txt ( comapres and gives3 columns in output ….. unique to file1, unique to file2,same in both files)

1. comm -1 cfile1.txt cfile2.txt ( apart from column1 everything will be displayed)
2. comm -12 cfile1.txt cfile2.txt ( apart from column1 and 2)
3. diff cfile1.txt cfile2.txt ( what changes need to be done in file 1 so that it becomes like file2)
4. diff cfile2.txt cfile1.txt ( what changes need to be done in file 2 so that it becomes like file1)
5. tr "[a-z]" "[A-Z]" < file1.txt ( translate to upper case)
6. cat trdemo.txt

tr -s " " <trdemo.txt ( translate by removing extra spaces)

1. cat trdemo1.txt

tr -s "a " < trdemo1.txt ( translate and remove all extra occurrence of a)

1. Ls- l (big list is displayed)
2. ls -l >dirlist.txt ( saved the big list in a filedirlist.txt

cat dirlist.txt ( display big list

more dirlist.txt ( converts the big list into pages) and on clicking space next page ,= then line no

1. /file ( all files having names file in it)
2. ls -l file1.txt ( display the rights)

chmod a+rwx friendfile.txt (change the rights of all )

ls -l friendfile.txt

1. chmod a-x friendfile.txt ( change / remove execute permission of all)

ls -l friendfile.txt ( display the rights)

1. chmod u+w ( give user write access )
2. chmod g+x,o-w friendfile.txt ( give execute access to group and remove write write access of others)
3. chmod 777 friendfile.txt ( give all access to file)read 4, 2rite 2 , execute =1
4. Chmod 761 friendfile.txt ( give all, read+write , execute access)
5. Mkdir dir1 (make directory
6. mkdir dir3 dir3/abc.txt ( make subdir and dir together)
7. cd testuser1

rmdir dir1 ( removes directory )

1. Rmdir abc and then dir(inside first)
2. echo " hello user"

**Chapter 3 Filters :**

1. cat employee.dat ( ~:^ any symbol except $ can be delimiter)
2. head employees.dat ( first 10 by default )
3. head -5 employees.dat ( first 5 )
4. tail -7 employees.dat (last 7)
5. tail -5c employees.dat (last 5 characters)
6. head -5 employees.dat | tail -1 (5th record)
7. cut -c1-4 employees.dat (cutting characters from 1 to 4)
8. cut -c1-4,31-35 employees.dat (cutting characters from 1 to 4 and 31 to 35)
9. cut -d "|" -f1,2,5 employees.dat (cutting displaying with delimiter | , fields 1,2 and 5)
10. cut -d -f1,2 employees.dat ( without delimiter then space is the delimiter)
11. cut -d "|" -f1 employees.dat > empabc1.dat

cut -d "|" -f2 employees.dat > empabc2.dat

paste empabc1.dat empabc2.dat

1. paste -d "|" empabc1.dat empabc2.dat (pasting with delimiter any delimiter can be used for pasting)
2. sort employees.dat (sort based on first field)
3. sort -r employees.dat (reverse sort)
4. sort -t "|" -k2 employees.dat ( sort based on second field)
5. sort -t "|" -r -k2 employees.dat ( reverse sort based on second field)
6. sort -t "|" -n -k5 employees.dat ( sort based on5th field which is numeric)
7. ccase sensitive -f?
8. sort -t "|" -k3,3 -k5,5n employees.dat ( sort department(3rd) based on salary field(5th) ….)
9. sort -t "|" -k3,3 -k5,5rn employees.dat (reverse on second field)
10. uniq friends ( removes duplicates)
11. uniq -u friends ( list of not repeated)
12. uniq -d friends ( list of repeated list
13. uniq -c friends ( count of repetition)
14. tee users.dat < employees.dat ( contents of employee is written on screen as well as copied to file users.dat)

cat users.dat

ls -l users.dat ( date n time of creation of file)

1. wc < file1.txt >result( words n characters of file.txt is taken as input n written to result
2. Ls -l | wc -l ( contents of ls is passed through pipe | to wc for counting words/chars
3. cat employees.dat | wc -l ( display no.oflines in employees)

cat employees.dat | wc -l >fcount ( save no.of lines in file fcount )

cat fcount

1. echo " todays date is `date` "
2. find / .profile -print (search from root directory the .profile file and print it)
3. find . .profile -print (search from current directory the .profile file and print it)
4. find .-name .profile -print (search from current directory the .profile file based on nameand print it)
5. find / -type f -name “myfile “ -print (search based on type = regular f and name= myfile from root / )
6. Cat employees.dat

Grep “Manager” employees.dat(searches for manager in the file employees)

1. Find is to find a file in the directory / root directory wheras grep is to find all the lines in a file where the pattern/word is
2. Find from directory and Grep for finding in file
3. Grep -i “Manager” employees.dat ( case insensitive search)
4. Grep -v “Manager” employees.dat ( those lines that does not contain manager)
5. Grep -c “Manager” employees.dat ( count of the no.of lines where manager is present)
6. Grep -n “Manager” employees.dat ( no.of the line where manager is present)
7. Grep “The” grepex.txt (searches for all the lines that has the word The, there, these etc
8. Grep -w “The” grepex.txt (searches for all the lines that has the word The only)
9. Grep “^r” friends ( lines starting with r)
10. Grep “t$” friends ( lines ending with t)
11. Grep “.[eu]” friends ( lines containg e or u)
12. Grep “r[aeu]” friends ( lines containg r followed by aeiou)
13. Grep “^r[aeu]” friends ( lines starting with r followed by aeiou)
14. grep "ra\{1,2\}khee" temp ( matching 1 or 2 occureneces of a)
15. grep "ra\{3\}khee" temp (a must match 3 occureneces of a)
16. grep "ra\{3,\}khee" temp (a must match 3 or more occureneces of a)
17. fgrep u cannot use regular expression..u can only use pattern , multiple patterns not related to each other
18. fgrep -e ‘LND’ -e ‘Director’ employee.dat ( searches for 2 patterns lnd and director)
19. fgrep -f patternfile employee.dat ( the pattern is present in the file patternfile)
20. egrep is extended regular matching like grep and fgrep

**chapter 4 VI EDITOR**

1. vi grepex.txt
2. vi demo1.sh

i

type something

Esc

:wq

Sh demo1.sh

1. Do example 2 instead of Esc ii try esc a ,o A, R.. etc from pdf
2. Esc S will replace entire line
3. Esc s will replace the letter by the new line
4. Esc R will start replacing full sentence from the cursor point
5. :w :q! (Save and exit)
6. Create a file grepex1.txt with below lines

There is only one Sun

There is only one Moon

There are multiple stars

:wq

Vi grepex1.txt

Esc

/The ( searches ‘the’ in the forward)

n

:q!

1. Now keep the cursor on the second line ‘ There’

Esc

?the ( searches ‘ the ‘in the backward)

n

1. I

:set number ( assigns numbers to the line)

1. Esc J ( joins 2 lines)
2. Esc 4J ( joins 4 lines)

Chapter 5

1. Set ( list of variables)
2. Vi .profile

echo "Welcome to unix"

echo $HOME

echo $LOGNAME

echo "the ps name is " $PS1

read PS2

echo " ps2 is " $PS2:

1. Cat > hello.sh

Echo “ Good Morning”

Read name

Echo “ Good $name”

Sh hello.sh

1. Sh -x hello.sh ( debugging mode on )
2. Vi addnos.sh

(Esc i)

echo "Enter first Number"

read no1

echo "Enter second Number"

read no2

res=`expr $no1 + $no2`

echo "The result is $res"

(esc)

:wq

Sh addnos.sh

1. Cp addnos.sh add2.sh

Vi add2.sh

echo "Enter first Number"

read no1

echo "Enter second Number"

read no2

#res=`expr $no1 + $no2` “ # for commenting)

let res=no1+no2

echo "The result is $res"

1. Echo “the date is; date” ????
2. Var=pwd

Echo $var ( Output is pwd)

1. Var=`pwd`

Echo $var (Output is the value in pwd i.e. \home\tesuser1)

1. Vi script1.sh

(create the file)

echo Program: $0

echo Number of arguments are $#

echo arguments are $\*

grep "$1" $2 || echo "pattern not found"(searches for pattern 1 in string2, if not found error msg)

echo -e "\n End of Script"

sh script1.sh PES employees.dat ( searches for PES in employees.dat )

1. And operator if second command is run then

Or operator if command is unsuccessful then Second command is run

Sd

1. Vi ifdemo.sh

if grep "^$1" /etc/passwd 2>/dev/null ( searches for word in path etc/passd and put error to devnull file)

then

echo "pattern found"

else

echo "pattern not found"

Fi

Sh idemo.sh root

1. = for comparing strings .. so use eq
2. Sh.greater.sh
3. Sh. Numbers.sh ( greatest of 3 nos)
4. Sh filetests.sh

Test.sh ( give input )

Test ( give input)

1. Sh checkvowel.sh (;; is break)

( input alphabet)

1. Sh continue.sh ( multiples of 7 don’t print for and if loop)
2. Sh until.sh (Until executes till condition is false)
3. While executes till condition is true
4. For x in 1 2 3 ( first x becomes 1,2,3)
5. Cc (compliation)
6. .bak (creates a back up file)
7. $ \* is argument in entered in command line
8. $ a is arguments entered in command line that can be used in for
9. Array.sh
10. $? is the return code (status code) of the last command