

**PRACTICAL FILE OF LINUX LAB**

**Subject Code: CSIT-505**

**B. Tech: III Year 5th Sem**

**Name of the Student : Aayush Mishra**

**Branch & Section : CSIT(CI)**

**Roll No. : 0827CI191002**

**EXPERIMENT NO: 1**

**AIM:** To Study basic & User status UNIX/Linux Commands.

**INTRODUCTION:**

**1. pwd Command**

The [pwd](https://www.javatpoint.com/linux-pwd) command is used to display the location of the current working directory.

**2. mkdir Command**

The [mkdir](https://www.javatpoint.com/linux-mkdir) command is used to create a new directory under any directory.

**3. rmdir Command**

The [rmdir](https://www.javatpoint.com/linux-rmdir) command is used to delete a directory.

**4. ls Command**

The [ls](https://www.javatpoint.com/linux-ls) command is used to display a list of content of a directory.

**5. cd Command**

The [cd](https://www.javatpoint.com/linux-cd) command is used to change the current directory.

**6. touch Command**

The [touch](https://www.javatpoint.com/linux-touch) command is used to create empty files. We can create multiple empty files by executing it once.

**7. cat Command**

It can be used to create a file,display content of the file, copy the content of one file to another file, and more.

**9. clear Command**

Linux **clear** command is used to clear the terminal screen.

**10. time Command**

The [time](https://www.javatpoint.com/linux-time) command is used to display the time to execute a command.

**11.date Command**

The [date](https://www.javatpoint.com/linux-date) command is used to display date, time, time zone, and more.

**12.cal Command**

The [cal](https://www.javatpoint.com/linux-cal) command is used to display the current month's calendar with the current date highlighted.

**[nameLinux@webminal.org ~]$cd Linux\_Lab**

**-sh: cd: Linux\_Lab: No such file or directory**

**[nameLinux@webminal.org ~]$ls Linux\_Lab**

**ls: cannot access Linux\_Lab: No such file or directory**

**[nameLinux@webminal.org ~]$ls**

**[nameLinux@webminal.org ~]$pwd**

**/home/nameLinux**

**[nameLinux@webminal.org ~]$mkdir CSIT\_dept**

**[nameLinux@webminal.org ~]$mkdir First.txt**

**[nameLinux@webminal.org ~]$ls**

**CSIT\_dept** **First.txt**

[nameLinux@webminal.org ~]$pwd

/home/nameLinux

[nameLinux@webminal.org ~]$cd CSIT\_dept

[nameLinux@webminal.org CSIT\_dept]$touch demo.py

[nameLinux@webminal.org CSIT\_dept]$ls

demo.py

[nameLinux@webminal.org CSIT\_dept]$time

user 0m0.13s

sys 0m0.03s

[nameLinux@webminal.org CSIT\_dept]$date

Mon Oct 11 12:30:07 CEST 2021

[nameLinux@webminal.org CSIT\_dept]$cal

October 2021

Su Mo Tu We Th Fr Sa

1 2

3 4 5 6 7 8 9

10 11 12 13 14 15 16

17 18 19 20 21 22 23

24 25 26 27 28 29 30

31

[nameLinux@webminal.org CSIT\_dept]$touch t1.txt

[nameLinux@webminal.org CSIT\_dept]$cat>t1.txt

CSIT - 5

CSIT - 10

CSIT - 15

CSIT - 20

^C

[nameLinux@webminal.org CSIT\_dept]$

**13. cut Command**

The [cut](https://www.javatpoint.com/linux-cut) command is used to select a specific column of a file.

**14. grep Command**

The 'grep' stands for "**global regular expression print**." It is useful for searching the content from a file. Generally, it is used with the pipe.

**15. sed command**

The [sed](https://www.javatpoint.com/linux-sed) command is also known as **stream editor**. It is used to edit files using a regular expression. It does not permanently edit files; instead, the edited content remains only on display. It does not affect the actual file.

**16. tee command**

The [tee](https://www.javatpoint.com/linux-tee) command is quite similar to the cat command. The only difference between both filters is that it puts standard input on standard output and also write them into a file.

**17. comm Command**

The ['comm'](https://www.javatpoint.com/linux-comm) command is used to compare two files or streams. By default, it displays three columns, first displays non-matching items of the first file, second indicates the non-matching item of the second file, and the third column displays the matching items of both files.

16. **mv:** mv stands for move. mv is used to move one or more files or directories    from one place to another in file system like UNIX. It has two distinct functions:

* It renames a file or folder.
* It moves group of files to different directory.

17.**more:** This command is used to view the text files in the command prompt, displaying one screen at a time in case the file is large.

18. **cp:** This command is used to copy files or group of files or directory.

19. **ps:** This command reports information on current running processes, outputting to standard output.

[nameLinux@webminal.org CSIT\_dept]$cat t1.txt

CSIT - 5

CSIT - 10

CSIT - 15

CSIT - 20

[nameLinux@webminal.org CSIT\_dept]$cut -d - -f1 t1.txt

CSIT

CSIT

CSIT

CSIT

[nameLinux@webminal.org CSIT\_dept]$cut -d - -f2 t1.txt

5

10

15

20

[nameLinux@webminal.org CSIT\_dept]$cat t1.txt|grep CS

**CS**IT - 5

**CS**IT - 10

**CS**IT - 15

**CS**IT - 20

[nameLinux@webminal.org CSIT\_dept]$

**EXPERIMENT NO: 2**

**AIM:** Study & use of commands for performing arithmetic operations with Unix/Linux.

**INTRODUCTION:**

| **+** | Addition, measures addition of numbers (operands) |
| --- | --- |

| **-** | Subtraction, measures subtraction of second operand from first |
| --- | --- |

| \* | Multiplication, measures the multiplication of operands. |
| --- | --- |

| **/** | Division, measures the division of first operand by second operand and and return quotient. |
| --- | --- |

| \*\* | Exponentiation, measures the result of second operand raised to the power of first operand. |
| --- | --- |

| += | Increment Variable by Constant- used to increment the value of first operand by the constant provided. |
| --- | --- |

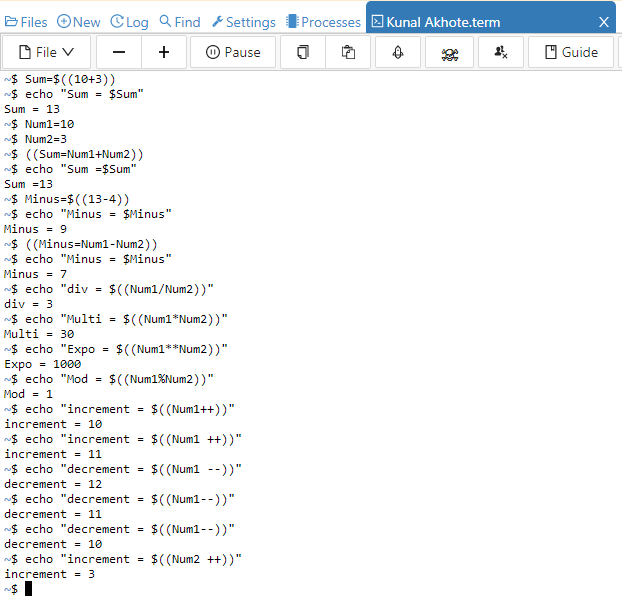
| **-=** | Decrement Variable by Constant- used to decrement the value of first operand by the constant provided. |
| --- | --- |

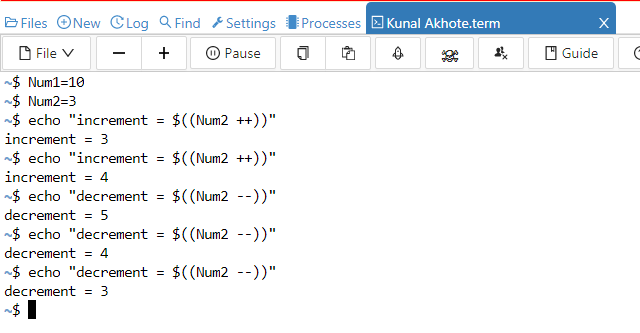
| \*= | Multiply Variable by Constant- used to multiply the value of the first operand by the constant provided. |
| --- | --- |

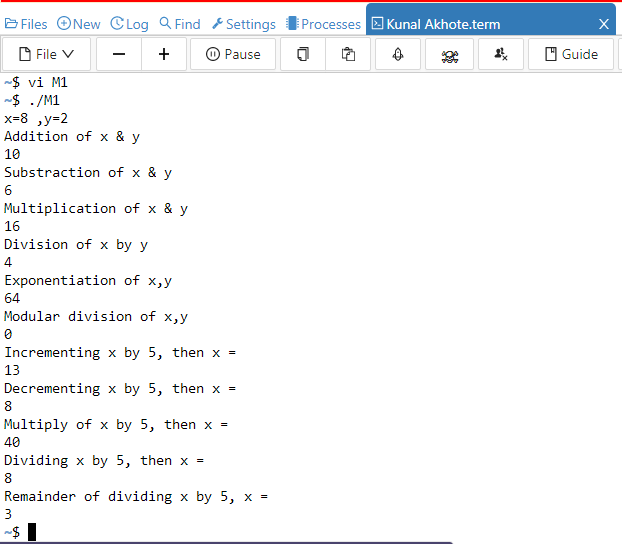
| **/=** | Divide Variable by Constant- used to calculate the value of (variable / constant) and store the result back to variable. |
| --- | --- |

| **%=** | Remainder of Dividing Variable by Constant- used to calculate the value of (variable % constant) and store the result back to variable. |
| --- | --- |

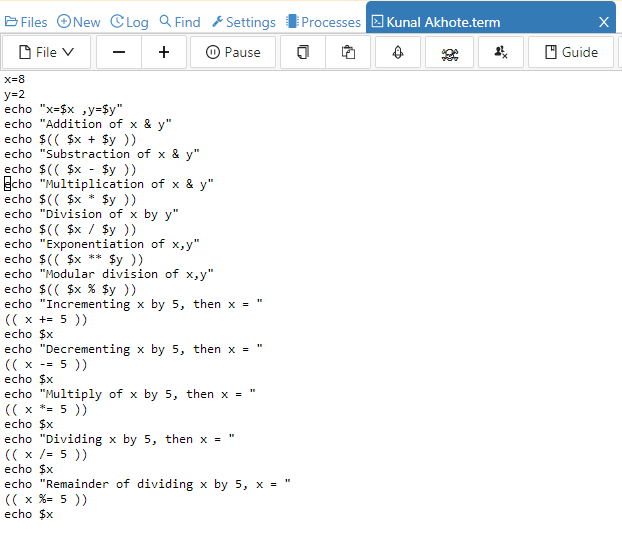
| **%** | Modulo, measures remainder when the first operand is divided by second operand. |
| --- | --- |







**VI M1 code:**



**EXPERIMENT NO: 3**

**AIM:** Create a file called wlcc.txt with some lines and display how many lines, words and characters are present in that file

**INTRODUCTION:**

**wc Command**

The [wc](https://www.javatpoint.com/linux-wc) command is used to count the lines, words, and characters in a file.

[nameLinux@webminal.org CSIT\_dept]$wc --help

Usage: wc [OPTION]... [FILE]...

or: wc [OPTION]... --files0-from=F

Print newline, word, and byte counts for each FILE, and a total line if

more than one FILE is specified. With no FILE, or when FILE is -,

read standard input. A word is a non-zero-length sequence of characters

delimited by white space.

The options below may be used to select which counts are printed, always in

the following order: newline, word, character, byte, maximum line length.

-c, --bytes print the byte counts

-m, --chars print the character counts

-l, --lines print the newline counts

--files0-from=F read input from the files specified by

NUL-terminated names in file F;

If F is - then read names from standard input

-L, --max-line-length print the length of the longest line

-w, --words print the word counts

--help display this help and exit

--version output version information and exit

For complete documentation, run: info coreutils 'wc invocation'

[nameLinux@webminal.org CSIT\_dept]$

**EXPERIMENT NO: 4**

**AIM:** Append ten more simple lines to the wlcc.txt file created above and split the appended file into 3 parts. What will be the names of these split files? Display the contents of each of these files. How many lines will be there on the last file?

**INTRODUCTION:**

**Cat >> filename** command is used to append ten more simple lines to the wlcc.txt file.

[nameLinux@webminal.org CSIT\_dept]$cat>>t1.txt

CSIT - 21

CSIT - 22

CSIT - 23

CSIT 24

CSIT -25

CSIT - 26

CSIT - 27

CSIT - 28

CSIT - 29

CSIT - 30

**split <filename>** command is used to split file.

**split -l 5 <file name>**

**Split -n 3 <file name>**

The option -l specifies the number of lines per output file. Since the input file contains 7 lines, the output files contain 3, 3 and 1 respectively. The output files generated are:

**Names of these split files are:-**

Original file: - wlcc.txt

After splitting the files

1. xaa
2. xab
3. xac
4. xad

[nameLinux@webminal.org CSIT\_dept]$split -l 5 t1.txt

[nameLinux@webminal.org CSIT\_dept]$ls

demo.py t1.txt xaa xab xac xad xae

[nameLinux@webminal.org CSIT\_dept]$cat xaa

CSIT - 5

CSIT - 10

CSIT - 15

CSIT - 20

[nameLinux@webminal.org CSIT\_dept]$cat xab

^C

[nameLinux@webminal.org CSIT\_dept]$cat xac

CSIT - 21

CSIT - 22

CSIT - 23

[nameLinux@webminal.org CSIT\_dept]$cat xad

CSIT 24

CSIT -25

CSIT - 26

CSIT - 27

CSIT - 28

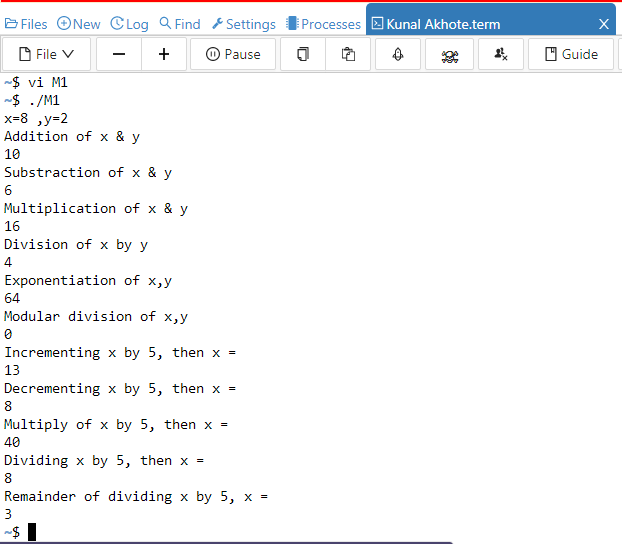
[nameLinux@webminal.org CSIT\_dept]$cat xae

CSIT - 29

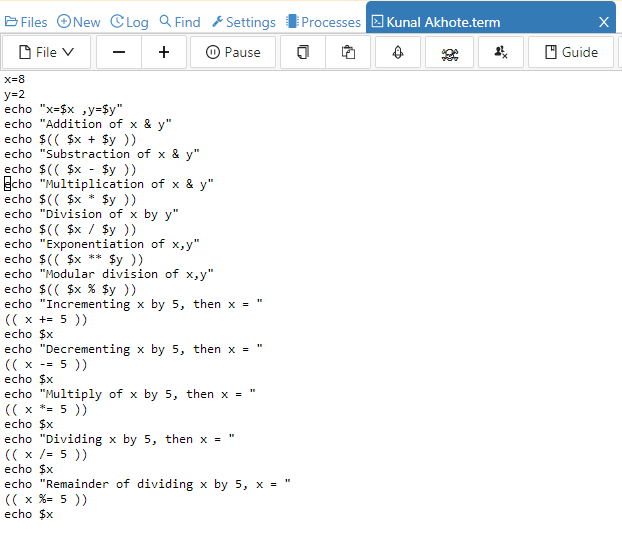
CSIT - 30

**EXPERIMENT NO: 5**

**AIM:** Execute shell commands through vi editor.

****

**VI M1 code:**

****

**EXPERIMENT NO: 6**

**AIM:** Write a Shell Script that accepts a file name, starting and ending line numbers as arguments and displays all lines between the given line numbers.

Program:

echo "enter the filename"

read fname

echo "enter the starting line number"

read s

echo "enter the ending line number"

read n

sed -n $s,$n\p $fname | cat > newline

cat newline

**output:**

[root@localhost ~]# vi 1s.sh

[root@localhost ~]# ./1s.sh

bash: ./1s.sh: Permission denied

[root@localhost ~]# chmod 777 1s.sh

[root@localhost ~]# ./1s.sh

enter the filename

sales.dat

enter the starting line number

2

enter the ending line number

4

1 computers 9161

1 textbooks 21312 2 clothing 3252

**EXPERIMENT NO: 7**

**AIM:** Write a shell script that deletes all lines containing a specific word in one or more files supplied as arguments to it.

[root@localhost ~]# vi del

unix is os

dos is also os

here using unix

unix is powerful os

~

[root@localhost ~]# vi dell

windowsnt is also os

there are some difference between unix and windowsnt

but unix is great among all os

## after creation two files now we have to write sed script file name is del.sed using vi editor.

[root@localhost ~]# vi del.sed

{

/os/d

}

#### Output:

[root@localhost ~]# sed -f del.sed del dell

here using unix

there are some difference between unix and windowsnt

**EXPERIMENT NO: 8**

**AIM:** Write a shell script that displays a list of files in current directory to which the user has read, write and execute permissions.

echo "enter the directory name"

read dir

if [ -d $dir ]

then

cd $dir

ls > f

exec < f

while read line

do

if [ -f $line ]

then

if [ -r $line -a -w $line -a -x $line ]

then

echo "$line has all permissions"

else

echo "files not having all permissions"

fi

fi

done

fi

#### Output:

student@ubuntu:~$sh prg3.sh

enter the directory name

dir1

ff has all permissions

files not having permissions

**EXPERIMENT NO: 9**

**AIM:** Write a shell script that accepts one or more file name as arguments and converts all of them to uppercase, provided they exist in the current directory.

if [ $# -eq 0 ]

then

echo "pass an argument !"

else

for i in $\*

do

if [ -f $i ]

then

a=`echo $i | tr '[a-z]' '[A-Z]' `

mv $i $a

echo "New file name is: $a"

else

echo "Not a file"

fi

done

fi

**EXPERIMENT NO: 10**

**AIM:** write a shell script that computes the gross salary of an employee according to the followingrules:

i. If basic salary is &lt;1500 then HRA=10% of the basic and DA=90% of the basic

ii) If the basic salary is&gt;=1500 then HRA=500/- and DA=98% of the basic

$ vi gsalary.sh

echo "enter the basic salary:"

read bsal

if [ $bsal -lt 1500 ]

then

gsal=$((bsal+((bsal/100)\*10)+(bsal/100)\*90))

echo "The gross salary : $gsal"

fi

if [ $bsal -ge 1500 ]

then

gsal=$(((bsal+500)+(bsal/100)\*98))

echo "the gross salary : $gsal"

fi

Output :

$ sh gsalary.sh

enter the basic salary:

1200

The gross salary : 2400

$ sh gsalary.sh

enter the basic salary:

2400

the gross salary : 5252

**EXPERIMENT NO: 11**

**AIM:** Write an interactive file –handling shell program. Let it offer the user the choice of copying removing, renaming, or linking files. Once the user has made a choice, have the same program ask the user for the necessary information, such as the file name ,new name and so on.

[CSESTAFF@localhost foss]$ cat krishna

No one is waste, No one is greate

Without work nothing is possible

With hard work nothing is impossible

So do your work perfectly

by Krishna

[CSESTAFF@localhost foss]$ cat krishna\_vivekananda

Be a Hero. Always Say,

'I have no fear'.

[CSESTAFF@localhost foss]$ vi 8a.sh

while true

do

echo "\*\*\*\*\*\*\*MENU\*\*\*\*\*\*\*\*\*"

echo "

1. List of files.

2. Copying files.

3. Removing files.

4. Renaming files.

5. Linking files.

press [CTRL+C] TO EXIT"

echo "enter your choice "

read ch

case "$ch" in

1 ) echo "The list of file names."

ls -l || echo "These are file";;

2) echo "Enter the old filename."

read ofile

echo "Enter the new file name."

read nfile

cp $ofile $nfile && echo "Copied sucessfully." || echo "Copied is not possible." ;;

3 ) echo "Enter the file name to remove."

read rfile

rm -f $rfile && echo "Successfully removed." ;;

4 ) echo "Enter the old file name."

read ofile

echo "Enter the new file name."

read nfile

mv $ofile $nfile && echo "The file $ofile name renamed to $nfile." || echo "You

cann't Rename the file.".;;

5 ) echo "Enter the original filename."

read ofile

echo "Enter the new filename to link a file."

read lfile

ln $ofile $lfile && echo "Creat the linking file Sccessfully." || echo "You cann't

Linking the file.";; \* )

echo "Invalid option."

Echo " Enter correct choice."

esac

done

Output:

[CSESTAFF@localhost foss]$ sh 8a.sh

\*\*\*\*\*\*\*MENU\*\*\*\*\*\*\*\*\*

1. List of files.

2. Copying files.

3. Removing files.

4. Renaming files.

5. Linking files.

press [CTRL+C] TO EXIT

enter your choice

1

The list of file names.

total 68

-rw-rw-r--. 1 CSESTAFF CSESTAFF 63 Jan 27 16:14 k3

-rw-rw-r--. 1 CSESTAFF CSESTAFF 141 Jan 27 16:06 kkk

-rw-rw-r--. 1 CSESTAFF CSESTAFF 141 Jan 31 16:59 krishna

-rw-rw-r--. 1 CSESTAFF CSESTAFF 41 Jan 31 16:59 krishna\_vivekananda

\*\*\*\*\*\*\*MENU\*\*\*\*\*\*\*\*\*

1. List of files.

2. Copying files.

3. Removing files.

4. Renaming files.

5. Linking files.

press [CTRL+C] TO EXIT

enter your choice

2

Enter the old filename.

pow.sh

Enter the new file name.

power.sh

Copied sucessfully.

\*\*\*\*\*\*\*MENU\*\*\*\*\*\*\*\*\*

1. List of files.

2. Copying files.

3. Removing files.

4. Renaming files.

5. Linking files.

press [CTRL+C] TO EXIT

enter your choice

3

Enter the file name to remove.

krishna

Successfully removed.

\*\*\*\*\*\*\*MENU\*\*\*\*\*\*\*\*\*

enter your choice

4

Enter the old file name.

krishnasai

Enter the new file name.

srisai

The file krishnasai name renamed to srisai.

\*\*\*\*\*\*\*MENU\*\*\*\*\*\*\*\*\*

enter your choice

5

Enter the original filename.

srisai

Enter the new filename to link a file.

krishnasai

Creat the linking file Sccessfully.

**EXPERIMENT NO: 12**

**AIM:** Write a shell script that accepts any number of arguments and prints them in the reverse order.

echo "input string is :$\*"

for (( i=$#;i>0;i-- ));do

echo "${!i}"

done

Output

input string is:abcd edd yht uhhh

uhhh

yht

edd

abcd

**EXPERIMENT NO: 13**

**AIM:** Write a Shell script to count the number of lines in a file that do not contain vowels.

echo "Enter your file name:"

read file

cnt=0

awk '$0!~/[aeiou]/{ cnt++ }

END{printf "The number of lines that does not contain vowels are: %d\n",count}' $f

Output :

[singh@00-13-02-56-15-7c programs]$ sh lin11.sh

Enter file name

test1

The number of lines that does not contain vowels are: 3

**EXPERIMENT NO: 14**

**AIM:** Write a shell script which receives two file names as arguments. It should check whether the two file contents are same or not. If they are same then second file should be deleted.

echo \"Enter first file name \"

read file1

echo \"Enter second file name \"

read file2

cmp $file1 $file2 > error

total=\`wc -c error | cut -f 7 -d \" \"\`

echo $total

if [ $total -eq 0 ]

then

echo \"Both file\'s contents are same\"

else

echo \"Both file\'s contents are not same\"

fi

**EXPERIMENT NO: 15**

**AIM:** Develop an interactive script that ask for a word and a file name and then tells how many times that word occurred in the file.

[CSESTAFF@localhost ex9]$ cat f1.txt

day by day week by end

week by week month by end

month by month year by end

but friendship is never end

[CSESTAFF@localhost ex9]$ vi wcount.sh

echo " Enter the word to be searched"

read word

echo "Enter the filename to be used"

read flname

echo "the no. of times the word ['$word'] occured in the file."

grep -o $word $flname|wc -l

echo " the no of line that contains ['$word'] is"

grep -c $word $flname

Output:

[CSESTAFF@localhost ex9]$ sh wcount.sh

Enter the word to be searched

by

Enter the filename to be used

f1.txt

the no. of times the word ['by'] occured in the file.

6

the no of line that contains ['by'] is

3

**EXPERIMENT NO: 16**

**AIM:** Write a shell script to perform the following string operations:

i. To extract a sub-string from a given string.

ii. To find the length of a given string.

[CSESTAFF@localhost ex9]$ vi substr.sh

echo "enter the string"

read str

strlen=${#str}

echo "The length of the given string '$str' is:$strlen "

echo "enter the string possibion in main str"

read s1

echo "ending possition"

read f1

echo $str | cut -c$s1-$f1

[CSESTAFF@localhost ex9]$

Output:

[CSESTAFF@localhost ex9]$ sh substr.sh

enter the string

krishnasai

The length of the given string 'krishnasai' is:10

enter the string possibion in main str

8

ending possition

10

sai

[CSESTAFF@localhost ex9]$

**EXPERIMENT NO: 17**

**AIM:** Write a shell script that accepts two integers as its arguments and computes the value of first number raised to the power of the second number.

if [ $# -ne 2 ]

then

echo "Error : Invalid no. of arguments."

exit

fi

pwr=`echo "$1 ^ $2" | bc`

echo "$1 ^ $2 = $pwr"

Output:

$sh 11a.sh 2 3

2^3 = 8

**EXPERIMENT NO: 18**

**AIM:** Write a shell script that takes a command –line argument and reports on whether it is directory, a file, or something else.

echo " enter file"

read str

if test -f $str

then echo "file exists n it is an ordinary file"

elif test -d $str

then echo "directory file"

else

echo "not exists"

fi

(or)

$ vi filetype.sh

echo "Enter the file name: "

read file

if [ -f $file ]

then

echo $file "---> It is a ORDINARY FILE."

elif [ -d $file ]

then

echo $file "---> It is a DIRCTORY."

else

echo $file "---> It is something else."

f

**EXPERIMENT NO: 19**

**AIM:** Develop an interactive grep script that asks for a word and a file name and then tells how many lines contain that word.

[CSESTAFF@localhost ex9]$ cat f1.txt

day by day week by end

week by week month by end

month by month year by end

but friendship is never end

[CSESTAFF@localhost ex9]$ vi wcount.sh

echo " Enter the word to be searched"

read word

echo "Enter the filename to be used"

read flname

echo "the no. of times the word ['$word'] occured in the file."

grep -o $word $flname|wc -l

echo " the no of line that contains ['$word'] is"

grep -c $word $flname

Output:

[CSESTAFF@localhost ex9]$ sh wcount.sh

Enter the word to be searched

by

Enter the filename to be used

f1.txt

the no. of times the word ['by'] occured in the file.

6

the no of line that contains ['by'] is

3

**EXPERIMENT NO: 20**

**AIM:** Write a shell script to list all of the directory files in a directory.

# !/bin/bash

echo "enter directory name"

read dir

if[ -d $dir]

then

echo "list of files in the directory"

ls –l $dir|egrep ‘^d’

else

echo "enter proper directory name"

fi

Output:

guest-glcbIs@ubuntu:~$sh lprg6.sh

enter directory name

dir1

list of files in the directory

drwxrwxr-x 4 guest-glcbls guest-glcbls 140 2012-07-06 14:40 dir1

**EXPERIMENT NO: 21**

**AIM**: Write and execute all process control commands.

Process control commands in Unix are:

bg - put suspended process into background

Syntax :

bg [job]

fg - bring process into foreground

Syntax :

fg [ %job]

jobs - list processes

Syntax :

jobs [JOB]

**EXPERIMENT NO: 22**

**AIM**: Study &amp; Installation of SAMBA, APACHE, TOMCAT.

SAMBA:- A Samba file server enables file sharing across different operating systems over a network. It lets you access your desktop files from a laptop and share files with Windows and macOS users. For Ubuntu 16.04 LTS To install Samba, we run: sudo apt update sudo apt install samba

We can check if the installation was successful by running: where is samba

The following should be its output:

samba: /usr/sbin/samba /usr/lib/samba /etc/samba /usr/share/samba /usr/share/man/ma

n7/samba.7.gz /usr/share/man/man8/samba.8.gz

APACHE:- Apache Web Server is a software package that turns a computer into an HTTP server. That is, it sends web pages – stored as HTML files – to people on the internet who request them. It is open-source software, which means it can be used and modified freely.

Open a terminal and type: sudo apt-get update

Let the package manager finish updating.

Step 1: Install Apache

To install the Apache package on Ubuntu, use the command:

sudo apt-get install apache2

The system prompts for confirmation – do so, and allow the system to complete the installation.

Step 2: Verify Apache Installation

To verify Apache was installed correctly, open a web browser and type in the address bar:

http://local.server.ip

The web browser should open a page labeled “Apache2 Ubuntu Default Page,” as in the image

below :

**EXPERIMENT NO: 23**

**AIM**: Study &amp; installation of Firewall &amp; Proxy server.

Firewall: A firewall is a network security device, either hardware or software-based, which monitors all incoming and outgoing traffic and based on a defined set of security rules it accepts, rejects or drops that specific traffic.

Accept : allow the traffic

Reject : block the traffic but reply with an “unreachable error”

Drop : block the traffic with no reply

$ sudo systemctl start firewalld #start the service for the meantime

$ sudo systemctl enable firewalld #enable the service to auto-start at boot time

$ sudo systemctl status firewalld #view service status

After starting firewalld service, you can also check whether the daemon is running or not, using

the firewall-cmd tool (in case it’s not active, this command will output “not running”).

$ sudo firewall-cmd --state

Check Firewalld Status

If you happen to save any changes permanently, you can reload firewalld. This will reload

firewall rules and keep state information. The current permanent configuration will become new

runtime configuration.

$ sudo firewall-cmd --reload

Proxy server: A proxy server is a server that physically sits between your computer and the

internet. ... The proxy server acts as the middleman and the web page doesn&#39;t see the request

coming from your computer. You can say proxy servers give the user who is requesting the web

page anonymity and security. Setting Up Temporary Proxy for a Single User

A temporary proxy connection resets after a system reboot. To establish such a connection for

the current user, use the export command.

The syntax for establishing a temporary proxy connection is:

export HTTP\_PROXY=[username]:[password]@[proxy-web-or-IP-address]:[port-number]

export HTTPS\_PROXY=[username]:[password]@[proxy-web-or-IP-address]:[port-number]

export FTP\_PROXY=[username]:[password]@ [proxy-web-or-IP-address]:[port-number]

...

export NO\_PROXY=localhost,127.0.0.1,::1

Provide the proxy address (web or IP), followed by the port number. If the proxy server requires

authentication, add your proxy username and password as the initial values.

This is what the set of commands should look like in terminal.