

No	AIM	Date	Sign
1	Study of Basic commands of Linux/UNIX.		
2	Study of Advance commands and filters of Linux/UNIX.		
3	Write a shell script to generate mark sheet of a student. Take 3 subjects, calculate and display total marks, percentage and Class obtained by the student.		
4	Write a shell script to find factorial of given number n.		
5	Write a shell script which will accept a number b and display first n prime numbers as output.		
6	Write a shell script which will generate first n Fibonacci numbers like: 1, 1, 2, 3, 5, 13,...		
7	Display calendar of current month Display today's date and time		
8	Display usernames those are currently logged in the system Display your name at given x, y position.		
9	Write a shell script to read n numbers as command arguments and sort them in descending order.		
10	Write a shell script to check entered string is palindrome or not..		
11	Write a shell script to validate the entered date. (eg. Date format is : dd-mm-yyyy)		
12	The distance between two cities (in km.) is input through the keyboard. Write a shell script to convert and print distance in meters, feet, inches and		

	centimeter.		
13	Write a shell script to input two no's from the user and perform addition, subtraction, multiplication, and division.		
14	Any integer is input through the keyboard. Write a shell script to find out whether it is an odd number or even number.		
15	Write a shell script which receives any year form the keyboard and determines whether the year is a leap year or not. If no argument is supplied the current year should be assumed.		
16	Write a shell script to print the series 1, 3, 5, 7, 9,, N.		
17	Write a program to print all prime no's from 1 to 300. (Hint – Use Nested Loops, break and continue)		

Practical 1 : Study of Basic commands of Linux/UNIX.

pwd : Print name of the “current working directory”.

Output : /home/yourname/whatever

ls : List contents of the current working directory

ls -l - long listing, with dates, owners, etc.

ls -lrt - above, but sorted by time

ls -lrt /home/yourname/something
- long-list a different directory

cd : Change the current working directory

cd /tmp/yourname/

- go to your temporary directory

cd - - go back to where you just were

cd - no arguments, go back “home”
“home” is where your login starts

mkdir : Create a new directory.

mkdir ./something - make it

cd ./something - go there

ls - check its is empty

man :

Display the manual for a given program

man ls - see manual for the “ls” command

man tcsh - learn about the C shell

man bash - learn about that other shell

man man - read the manual for the manual

to return to the command prompt, type “q”

mv :

Move or rename a file. If you think about it, these are the same thing.

mv stupidname.txt bettername.txt

- change name

mv stupidplace/file.txt ../betterplace/file.txt

- same name, different directory

mv stupidname_*.img bettername_*.img

Will not work! Never ever do this!

cp :

Copy a file. This is just like “mv” except it does not delete the original.

cp stupidname.txt bettername.txt
- change name, keep original
rm stupidname.txt
- now this is the same as “mv”

chmod :

Change the “permission” of a file.

chmod a+r filename.txt

- make it so everyone can read it

chmod u+rw filename.txt

- make it you can read/write/execute it

chmod -R u+rw /some/random/place

- make it so you can read/write everything under
a directory

rm :

Remove a file forever. There is no “trash” or “undelete” in unix.

rm unwanted_file.txt

- delete file with that name

rm -f /tmp/yourname/*

- forcefully remove everything in your temporary directory.
Will not prompt for confirmation!

more :

Display the contents of a text file, page by page

more filename.txt - display contents

less filename.txt - many installs now have a replacement for “more” called “less” which has
nicer search features.

to return to the command prompt, type “q”

df du :

Check how much space is left on disks

df - look at space left on all disks

df . - look at space left in the current working directory

du -sk . | sort -g

- add up space taken up by all files and subdirectories, list biggest hog last

date :

- Guess what :-)
- Displays dates in various formats
- % date
- % date -u
 - in GMT

- % man date

cal :

- % cal current month
- % cal 2 2000 Feb 2000, leap year
- % cal 2 2100 not a leap year
- % cal 2 2400 leap year
- % cal 9 1752 11 days skipped
- % cal 0 error
- % cal 2002 whole year

clear :

- Clears the screen
- There's an alias for it: Ctrl+L
- Example sequence:
 - % cal
 - % clear
 - % cal
 - Ctrl+L

cat :

- Display and concatenate files.
- % cat
 - Will read from STDIN and print to STDOUT every line you enter.
- % cat file1 [file2] ...
 - Will concatenate all files in one and print them to STDOUT
- % cat > filename
 - Will take whatever you type from STDIN and will put it into the file filename
- To exit cat or cat > filename type Ctrl+D to indicate EOF (End of File).

touch :

- By *touching* a file you either create it if it did not exist (with 0 length).
- Or you update its last modification and access times.
- There are options to override the default behavior.
- % touch file
- % man touch

grep :

- Searches its input for a pattern.
- The pattern can be a simple substring or a complex regular expression.
- If a line matches, it's directed to STDOUT; otherwise, it's discarded.
- % echo "blah-foo" | grep blah

- Will print the matching line
- % echo “blah-foo” | grep zee
 - Will not.
- See a separate grep tutorial.

Practical 2 : Study of Advance commands and filters of Linux/UNIX.

- In this practical, we will cover the basics of the Linux/UNIX file system.
 - We will also cover the commands that are used to work with the file system such as ls, touch, mkdir, rm, cp, mv, rmdir, man, cd, history, pwd, clear, head, tail, cat, wc, date, timedatectl, su, sudo, chage, etc.
1. ls – List directory contents
 2. touch – Create an empty file or update file timestamps
 3. mkdir – Create a new directory
 4. rm – Remove files or directories
 5. cp – Copy files or directories
 6. mv – Move or rename files or directories
 7. rmdir – Remove a directory
 8. man – Display the manual pages for a command
 9. cd – Change the current working directory
 10. history – View a list of recently executed commands
 11. pwd – Print the current working directory
 12. clear – Clear the terminal screen
 13. head – Display the first few lines of a file
 14. tail – Display the last few lines of a file
 15. cat – Concatenate and display files
 16. wc – Count the number of lines, words, and characters in a file
 17. date – Print or set the system date and time
 18. timedatectl – Control the system time and date settings
 19. su – Switch user account to become a different user
 20. sudo – Execute a command with superuser privileges
 21. chage – Change the password expiration settings for a user account

Practical 3 : Write a shell script to generate mark sheet of a student. Take 3 subjects, calculate and display total marks, percentage and Class obtained by the student.

```
echo "Enter student name:"
read name
echo "Enter marks for subject 1:"
read sub1
echo "Enter marks for subject 2:"
read sub2
echo "Enter marks for subject 3:"
read sub3
total=$((sub1+sub2+sub3))
percentage=$(echo "scale=2; $total/3" | bc)
if (( percentage >= 90 ))
then
    class="Distinction"
elif (( percentage >= 60 ))
then
    class="First Class"
elif (( percentage >= 40 ))
then
    class="Pass Class"
else
    class="Fail"
fi
echo "_____ "
echo "    MARK SHEET    "
echo "_____ "
```



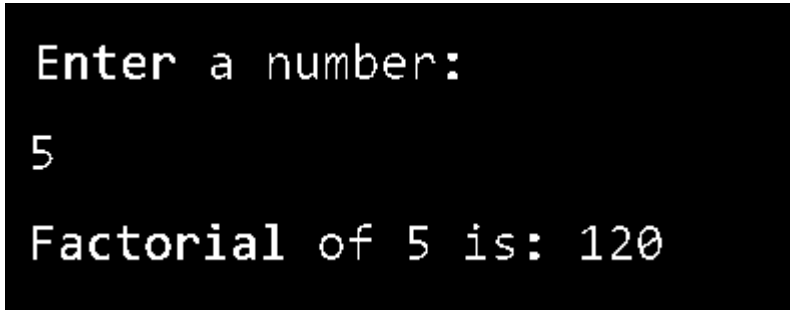
```
echo "Name: $name"
echo "....."
echo "Subject 1 Marks: $sub1"
echo "Subject 2 Marks: $sub2"
echo "Subject 3 Marks: $sub3"
echo "....."
echo "Total Marks: $total"
echo "Percentage: $percentage%"
echo "Class Obtained: $class"
echo "....."
```

Output :

```
Enter student name:
John Wick
Enter marks for subject 1:
75
Enter marks for subject 2:
85
Enter marks for subject 3:
90
-----
MARK SHEET
-----
Name: John Doe
-----
Subject 1 Marks: 75
Subject 2 Marks: 85
Subject 3 Marks: 90
-----
Total Marks: 250
Percentage: 83.33%
Class Obtained: First Class
-----
```

Practical 4 : Write a shell script to find factorial of given number n.

```
echo "Enter a number: "  
read n  
fact=1  
if [ $n -lt 0 ]  
then  
    echo "Factorial is not defined for negative numbers."  
elif [ $n -eq 0 ]  
then  
    echo "Factorial of 0 is 1."  
else  
    for ((i=1;i<=n;i++))  
    do  
        fact=$((fact*i))  
    done  
    echo "Factorial of $n is: $fact"  
fi
```

Output :

```
Enter a number:  
5  
Factorial of 5 is: 120
```

Practical 5 : Write a shell script which will accept a number b and display first n prime numbers as output.

```
echo "Enter the value of b:"
```

```
read b
```

```
echo "Enter the value of n:"
```

```
read n
```

```
count=0
```

```
num=2
```

```
while [ $count -lt $n ]
```

```
do
```

```
    isPrime=true
```

```
    for (( i=2; i<=num/2; i++ ))
```

```
    do
```

```
        if [ $((num%i)) -eq 0 ]
```

```
        then
```

```
            isPrime=false
```

```
            break
```

```
        fi
```

```
    done
```

```
    if [ "$isPrime" = true ]
```

```
    then
```

```
        if [ $num -gt $b ]
```

```
        then
```

```
            echo "$num"
```

```
            count=$((count+1))
```

```
        fi
```

```
    fi
```

```
num=$((num+1))
```

```
done
```

Output :

```
Enter the value of b:
```

```
5
```

```
Enter the value of n:
```

```
3
```

```
7
```

```
11
```

```
13
```

Practical 6 : Write a shell script which will generate first n Fibonacci numbers like: 1, 1, 2, 3,5,13,...

```
echo "Enter the value of n: "  
read n  
a=0  
b=1  
count=1  
echo "Fibonacci series up to $n terms:"  
echo $a  
while [ $count -lt $n ]  
do  
    echo $b  
    c=$((a + b))  
    a=$b  
    b=$c  
    count=$((count + 1))  
done
```

Output :

```
Enter the value of n:  
10  
Fibonacci series up to 10 terms:  
0  
1  
1  
2  
3  
5  
8  
13  
21  
34
```

Practical 7 : Display calendar of current month , Display today's date and time.

```
echo "Calendar for the current month:"
```

```
cal
```

```
echo "Today's date and time:"
```

```
date
```

Output :

```
Calendar for the current month:
```

```
April 2023
```

```
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
```

```
Today's date and time:
```

```
Tue Apr 4 21:08:38 UTC 2023
```

Practical 8 : Display usernames those are currently logged in the system Display your name at given x, y position

```
echo "Currently logged in users:"  
who | awk '{print $1}'  
echo -e "\033[10;20HMy name is Alice."
```

Output :

```
Currently logged in users:  
alice  
bob  
  
My name is Alice.
```

Practical 9 : Write a shell script to read n numbers as command arguments and sort them in descending order

```
arr=("$@")
len=${#arr[@]}
for ((i=0;i<len;i++))
do
    for ((j=i+1;j<len;j++))
    do
        if (( ${arr[i]} < ${arr[j]} ))
        then
            temp=${arr[i]}
            arr[i]=${arr[j]}
            arr[j]=$temp
        fi
    done
done
echo "Sorted array in descending order:"
for ((i=0;i<len;i++))
do
    echo "${arr[i]}"
done
```

Input : ./sort_numbers.sh 5 1 3

Output :

```
Sorted array in descending order:
5
3
1
```


Practical 10 : Write a shell script to check entered string is palindrome or not.

```
echo "Enter a string:"
read input_string

reverse_string=$(echo "$input_string" | rev)

if [ "$input_string" == "$reverse_string" ]
then
    echo "The input string is a palindrome."
else
    echo "The input string is not a palindrome."
fi
```

Output :

```
Enter a string:
racecar
The input string is a palindrome.
```

Practical 11 : Write a shell script to validate the entered date. (eg. Date format is : dd-mm-yyyy)

```
read -p "Enter a date (dd-mm-yyyy): " input_date
if ! [[ "$input_date" =~ ^([0-9]{2})-([0-9]{2})-([0-9]{4})$ ]]; then
    echo "Invalid date format. Please enter a date in the format of dd-mm-yyyy."
    exit 1
fi
day=${BASH_REMATCH[1]}
month=${BASH_REMATCH[2]}
year=${BASH_REMATCH[3]}
if (( day < 1 || day > 31 )); then
    echo "Invalid day. Please enter a day between 1 and 31."
    exit 1
fi
if (( month < 1 || month > 12 )); then
    echo "Invalid month. Please enter a month between 1 and 12."
    exit 1
fi
current_year=$(date +%Y)
if (( year < 1900 || year > current_year )); then
    echo "Invalid year. Please enter a year between 1900 and $current_year."
    exit 1
fi
echo "The entered date ($input_date) is valid."
```

Output :

```
Enter a date (dd-mm-yyyy): 15-04-2022
The entered date (15-04-2022) is valid.
```

**Practical 12 : The distance between two cities (in km.) is input through the keyboard.
Write a shell script to convert and print distance in meters, feet, inches and centimeter.**

```
read -p "Enter the distance between two cities (in km): " distance_km
distance_m=$(echo "scale=2; $distance_km * 1000" | bc)
distance_ft=$(echo "scale=2; $distance_km * 3280.84" | bc)
distance_in=$(echo "scale=2; $distance_ft * 12" | bc)
distance_cm=$(echo "scale=2; $distance_m * 100" | bc)
echo "Distance in meters: $distance_m m"
echo "Distance in feet: $distance_ft ft"
echo "Distance in inches: $distance_in in"
echo "Distance in centimeters: $distance_cm cm"
```

Output :

```
Enter the distance between two cities (in km): 10
Distance in meters: 10000.00 m
Distance in feet: 32808.40 ft
Distance in inches: 393700.80 in
Distance in centimeters: 1000000.00 cm
```

Practical 13 : Write a shell script to input two no's from the user and perform addition, subtraction, multiplication, and division.

```
read -p "Enter the first number: " num1
read -p "Enter the second number: " num2
sum=$(echo "$num1 + $num2" | bc)
echo "Sum: $sum"
diff=$(echo "$num1 - $num2" | bc)
echo "Difference: $diff"
prod=$(echo "$num1 * $num2" | bc)
echo "Product: $prod"
if (( $(echo "$num2 == 0" | bc -l) )); then
    echo "Cannot divide by zero"
else
    quotient=$(echo "scale=2; $num1 / $num2" | bc)
    echo "Quotient: $quotient"
fi
```

Output :

```
Enter the first number: 5
Enter the second number: 2
Sum: 7
Difference: 3
Product: 10
Quotient: 2.50
```

Practical 14 : Any integer is input through the keyboard. Write a shell script to find out whether it is an odd number or even number.

```
read -p "Enter an integer: " num
if (( num % 2 == 0 )); then
    echo "$num is an even number."
else
    echo "$num is an odd number."
fi
```

Output :

```
Enter an integer: 14
14 is an even number.
```

Practical 15 : Write a shell script which receives any year form the keyboard and determines whether the year is a leap year or not. If no argument is supplied the current year should be assumed.

```
if [[ $# -eq 1 ]]; then
    year=$1
else
    year=$(date +%Y)
fi
if (( year % 4 == 0 && (year % 100 != 0 || year % 400 == 0) )); then
    echo "$year is a leap year."
else
    echo "$year is not a leap year."
fi
```

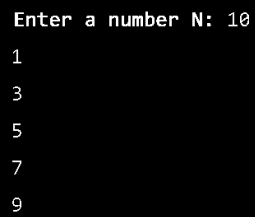
Output :

```
$ bash leap_year.sh 2024
2024 is a leap year.

$ bash leap_year.sh
2023 is not a leap year.
```

Practical 16 : Write a shell script to print the series 1, 3, 5, 7, 9, ..., N.

```
read -p "Enter a number N: " N
for (( i=1; i<=N; i+=2 ))
do
    echo "$i"
done
```

Output :

```
Enter a number N: 10
1
3
5
7
9
```

Practical 17 : Write a program to print all prime no's from 1 to 300. (Hint – Use Nested Loops, break and continue)

```
for ((i=2;i<=300;i++))
do
    is_prime=true
    for ((j=2;j<i;j++))
    do
        if (( i % j == 0 ))
        then
            is_prime=false
            break
        fi
    done
    if [ "$is_prime" = true ]
    then
        echo -ne "$i\t"
    fi
done
```

Output :

2	3	5	7	11	13	17	19	23	29	31	37	41	43	47
53	59	61	67	71	73	79	83	89	97	101	103	107	109	113
127	131	137	139	149	151	157	163	167	173	179	181	191	193	197
199	211	223	227	229	233	239	241	251	257	263	269	271	277	281
283	293													