SHELL COMMAND

1. ‘ who ‘ command

The main usage of **who** command in Linux without command-line parameter is to show the name of the users who are logged in currently.

```
anithalder@kali:~$ who
anithalder tty7         2022-09-10 09:47 (:0)
```

2. ‘ whoami ‘ command

The **whoami** command allows Linux users to see the currently logged-in user. The output displays the username of the effective user in the current shell.Additionally, **whoami** is useful in bash scripting to show who is running the script .

```
anithalder@kali:~$ whoami
anithalder
```

3. ‘ cat ‘ command

The **cat** (concatenate) command is one of the most frequently used commands in Linux/Unix-like operating systems. **cat** command allows us to create single or multiple files, view content of a file, concatenate files and redirect output in terminal or files.

```
anithalder@kali:~$ whoami
anithalder
```

```
anithalder@kali:~/Desktop$ cat>>anit.txt
I am basicaly writen in linux operating system

anithalder@kali:~/Desktop$ cat anit.txt
Hello World
I am a Shell program
My programmer name is Anit Halder
I am basicaly writen in linux operating system

anithalder@kali:~/Desktop$ cat anit.txt halder.txt >> anithalder.txt

anithalder@kali:~/Desktop$ cat anithalder.txt
Hello World
I am a Shell program
My programmer name is Anit Halder
I am basicaly writen in linux operating system
Hi My name is Anit Halder
I am a collage student
```

4. ‘ mkdir ‘ command

The mkdir command in Linux allows users to create or make new directories. **mkdir** stands for ―make directory.‖ With **mkdir** , you can also set permissions, create multiple directories (folders) at once, and much more.

```
anithalder@kali:~/Desktop$ ls
anithalder.txt  anit.txt  halder.txt

anithalder@kali:~/Desktop$ mkdir anit

anithalder@kali:~/Desktop$ ls
anit  anithalder.txt  anit.txt  halder.txt
```

5. ‘ rm ‘ command

The **rm** command removes the entries for a specified file, group of files, or certain select files from a list within a directory. User confirmation, read permission, and write permission are not required before a file is removed when you use the **rm** command.

-f **:** Forces the removal of all files or directories.

-i **:** Prompts for confirmation before removing.

-I **:** Prompts once before removing more than three files or when removing recursively.

-r **:** Removes directories and their content recursively.

-d **:** Removes empty directories.

-v **:** Provides a verbose output.

```
anithalder@kali:~/Desktop$ ls
anit  anithalder.txt  anit.txt  a.txt  halder.txt

anithalder@kali:~/Desktop$ rm a.txt

anithalder@kali:~/Desktop$ ls
anit  anithalder.txt  anit.txt  halder.txt
```

```
anithalder@kali:~/Desktop$ ls
anit  anithalder.txt  anit.txt  a.txt  halder.txt

anithalder@kali:~/Desktop$ rm -i a.txt
rm: remove regular empty file 'a.txt'? yes

anithalder@kali:~/Desktop$ ls
anit  anithalder.txt  anit.txt  halder.txt
```

6. ‘ rmdir ‘ command

rmdir command removes empty directories specified on the command line. These directories are removed from the filesystem in the same order as they are specified on the command line, i.e., from left to right. If the directory is not empty you get an error message.

```
anithalder@kali:~/Desktop$ ls
anit  anithalder.txt  anit.txt  halder.txt  linux

anithalder@kali:~/Desktop$ rmdir linux

anithalder@kali:~/Desktop$ ls
anit  anithalder.txt  anit.txt  halder.txt
```

7. ‘ mv ‘ command

The **mv** command to move files and directories from one directory to another or to rename a file or directory. If you move a file or directory to a new directory without specifying a new name, it retains its original name.

```
anithalder@kali:~/Desktop$ ls
anit  anithalder.txt  anit.txt  halder.txt

anithalder@kali:~/Desktop$ mv anit.txt anit

anithalder@kali:~/Desktop$ ls
anit  anithalder.txt  halder.txt

anithalder@kali:~/Desktop$ cd anit

anithalder@kali:~/Desktop/anit$ ls
anit.txt
```

8. ‘ cp ‘ command

We use the **cp** command for copying files from one location to another. This command can also copy directories (folders).

```
anithalder@kali:~/Desktop$ ls -l
total 16
drwxr-xr-x 2 anithalder anithalder 4096 Sep 10 11:05 anit
-rw-r--r-- 1 anithalder anithalder  163 Sep 10 10:23 anithalder.txt
-rw-r--r-- 1 anithalder anithalder  114 Sep 10 10:18 anit.txt
-rw-r--r-- 1 anithalder anithalder   49 Sep 10 10:20 halder.txt

anithalder@kali:~/Desktop$ chmod 777 anit.txt

anithalder@kali:~/Desktop$ ls -l
total 16
drwxr-xr-x 2 anithalder anithalder 4096 Sep 10 11:05 anit
-rw-r--r-- 1 anithalder anithalder  163 Sep 10 10:23 anithalder.txt
-rwxrwxrwx 1 anithalder anithalder  114 Sep 10 10:18 anit.txt
-rw-r--r-- 1 anithalder anithalder   49 Sep 10 10:20 halder.txt
```

9. ‘ ls ‘ command

The **ls** command is used to list files or directories in Linux and other Unix-based operating systems. Just like you navigate in your File explorer or Finder with a GUI, the **ls** command allows you to list all files or directories in the current directory by default, and further interact with them via the command line.

```
anithalder@kali:~/Desktop$ ls
anit  anithalder.txt  anit.txt  halder.txt

anithalder@kali:~/Desktop$ ls -l
total 16
drwxr-xr-x 2 anithalder anithalder 4096 Sep 10 10:29 anit
-rw-r--r-- 1 anithalder anithalder  163 Sep 10 10:23 anithalder.txt
-rw-r--r-- 1 anithalder anithalder  114 Sep 10 10:18 anit.txt
-rw-r--r-- 1 anithalder anithalder   49 Sep 10 10:20 halder.txt
```

10. ‘ wc ‘ command

The **wc** command in Linux is a command line utility for printing newline, word and byte counts for files. It can return the number of lines in a file, the number of characters in a file and the number of words in a file. It can also be combine with pipes for general counting operations.

```
anithalder@kali:~/Desktop$ wc anit.txt
  4  21 114 anit.txt

anithalder@kali:~/Desktop$ wc -c anit.txt
114 anit.txt

anithalder@kali:~/Desktop$ wc -l anit.txt
4 anit.txt

anithalder@kali:~/Desktop$ wc -w anit.txt
21 anit.txt
```

11. ‘ pwd ‘ command

The **pwd** command writes to standard output the full path name of your current directory (from the root directory). All directories are separated by a / (slash). The root directory is represented by the first /, and the last directory named is your current directory.

```
anithalder@kali:~/Desktop$ pwd
/home/kali/Desktop
```

12. ‘ chmod ‘ command

The **chmod** (short for change mode) command is used to manage file system access permissions on Linux and Unix-like systems. There are three basic file system permissions, or modes, to files and directories: read (r) write (w).

- chmod +rwx filename to add permissions.
- chmod -rwx directoryname to remove permissions.
- chmod +x filename to allow executable permissions.
- chmod -wx filename to take out write and executable permissions.

```
anithalder@kali:~/Desktop$ ls -l
total 16
drwxr-xr-x 2 anithalder anithalder 4096 Sep 10 11:05 anit
-rw-r--r-- 1 anithalder anithalder  163 Sep 10 10:23 anithalder.txt
-rw-r--r-- 1 anithalder anithalder  114 Sep 10 10:18 anit.txt
-rw-r--r-- 1 anithalder anithalder   49 Sep 10 10:20 halder.txt

anithalder@kali:~/Desktop$ chmod 777 anit.txt

anithalder@kali:~/Desktop$ ls -l
total 16
drwxr-xr-x 2 anithalder anithalder 4096 Sep 10 11:05 anit
-rw-r--r-- 1 anithalder anithalder  163 Sep 10 10:23 anithalder.txt
-rwxrwxrwx 1 anithalder anithalder  114 Sep 10 10:18 anit.txt
-rw-r--r-- 1 anithalder anithalder   49 Sep 10 10:20 halder.txt
```

13. ‘ expr ‘ command

expr is a command line Linux utility which evaluates an expression and outputs the corresponding value. **expr** evaluates integer or string expressions, including pattern matching regular expressions.

```
anithalder@kali:~$ expr 5 + 7
12

anithalder@kali:~$ expr 5 - 7
-2

anithalder@kali:~$ expr 5 \* 7
35

anithalder@kali:~$ expr 5 / 7
0

anithalder@kali:~$ expr 5 % 7
5

anithalder@kali:~$ echo 7 / 5 | bc
1
```

14. ‘ echo ‘ command

echo command in linux is used to display line of text/string that are passed as an argument . This is a built in command that is mostly used in shell scripts and batch files to output status text to the screen or a file. In above example, text after \c is not printed and omitted trailing new line.

```
anithalder@kali:~/Desktop$ echo "Even Death I am The Hero (EDITH)"
Even Death I am The Hero (EDITH)
```

1. Write a shell program to perform the addition of two numbers.

Bash Script

```
echo "Enter the 1st number"
read a
echo "Enter the 2nd number"
read b
S=`expr $a + $b`
echo "The result is:"$S
```

Output

```
pralay@pralay-mint:/media/pralay/PROG/Programs/SH/anit$ sh 1.sh
Enter the 1st number
56
Enter the 2nd number
87
The result is:143
pralay@pralay-mint:/media/pralay/PROG/Programs/SH/anit$ sh 1.sh
Enter the 1st number
-8
Enter the 2nd number
-4
The result is:-12
pralay@pralay-mint:/media/pralay/PROG/Programs/SH/anit$ sh 1.sh
Enter the 1st number
-9
Enter the 2nd number
6
The result is:-3
```

2. Write shell script to show the all-natural numbers from 1 to n (n is taken from the user).

Bash Script

```
read -p "Enter the limit of the series:" n
i=1
echo -n "\nThe realnumber series is:"
while [ $i -le $n ]
do
echo -n " " $i
i=`expr $i + 1`
done
```

Output

```
pralay@pralay-mint:/media/pralay/PROG/Programs/SH/anit$ sh 2.sh
Enter the limit of the series:10

The realnumber series is:  1  2  3  4  5  6  7  8  9  10

pralay@pralay-mint:/media/pralay/PROG/Programs/SH/anit$ sh 2.sh
Enter the limit of the series:5

The realnumber series is:  1  2  3  4  5
```

3. Write a shell program to find the maximum number between two number

Bash Script

```
echo "Ente the 1st number"
read a
echo "Ente the 2nd number"
read b
if [ $a -gt $b ]
then
echo "The greater number is:"$a
else
echo "The greater number is:"$b
fi
```

Output

```
pralay@pralay-mint:/media/pralay/PROG/Programs/SH/anit$ sh 3.sh
Ente the 1st number
24
Ente the 2nd number
65
The greater number is:65
pralay@pralay-mint:/media/pralay/PROG/Programs/SH/anit$ sh 3.sh
Ente the 1st number
-8
Ente the 2nd number
-5
The greater number is:-5
pralay@pralay-mint:/media/pralay/PROG/Programs/SH/anit$ sh 3.sh
Ente the 1st number
9
Ente the 2nd number
-5
The greater number is:9
```

4. Write a shell script program to calculate the what is the greater number between three number.

Bash Script

```
echo "Enter the numbers"
read a b c
if [ $a -gt $b ]
then
        if [ $a -gt $c ]
        then
                echo "The greater number is:"$a
        else
                echo "The greater number is:"$c
        fi
elif [ $a -eq $b -a $b -eq $c ]
then
        echo "The number are equal"
else
        if [ $b -gt $c ]
        then
                 echo "The greater number is:"$b
        else
                 echo "The greater number is:"$c
        fi
fi
```

Output

```
pralay@pralay-mint:/media/pralay/PROG/Programs/SH/anit$ sh 4.sh
Enter the numbers
12 5 10
The greater number is:12
pralay@pralay-mint:/media/pralay/PROG/Programs/SH/anit$ sh 4.sh
Enter the numbers
-5 -8 6
The greater number is:6
pralay@pralay-mint:/media/pralay/PROG/Programs/SH/anit$ sh 4.sh
Enter the numbers
-9 -4 -12
The greater number is:-4
```

5. Write a shell script program to find the number is even or odd.

Bash Script

```
echo "Enter the number"
read n
s=`expr $n % 2`
if [ $s -eq 0 ]
then
        echo "The number is even"
else
        echo "The number is odd"
fi
```

Output

```
pralay@pralay-mint:/media/pralay/PROG/Programs/SH/anit$ sh 5.sh
Enter the number
25
The number is odd
pralay@pralay-mint:/media/pralay/PROG/Programs/SH/anit$ sh 5.sh
Enter the number
20
The number is even
```

6. Write a shell script to check whether a year is leapyear or not.

Bash Script

```
read -p "Enter the year: " year

if ((year % 400 == 0 && year % 100 == 0 ||
year % 4 == 0)); then
   echo -e "\n$year year is leap year\n"
else
   echo -e "\nThe year is not leap year\n"
fi
```

Output

```
pralay@pralay-mint:/media/pralay/PROG/Programs/SH/anit$ chmod +x 7.sh
pralay@pralay-mint:/media/pralay/PROG/Programs/SH/anit$ ./7.sh
Enter the year: 2024

2024 year is leap year

pralay@pralay-mint:/media/pralay/PROG/Programs/SH/anit$ ./7.sh
Enter the year: 2020

2020 year is leap year
```

7. Write a shell script to print the factorial of a user given number.

Bash Script

```
echo -n "Enter a number: "
read num
fact=1
for (( i=1; i<=num; i++ ))
do
  fact=$((fact * i))
done

echo "Factorial of $num is $fact"
```

Output

```
pralay@pralay-mint:/media/pralay/PROG/Programs/SH/anit$ chmod +x 6.sh
pralay@pralay-mint:/media/pralay/PROG/Programs/SH/anit$ ./6.sh
Enter a number: 5
Factorial of 5 is 120
pralay@pralay-mint:/media/pralay/PROG/Programs/SH/anit$ ./6.sh
Enter a number: 8
Factorial of 8 is 40320
```

8. Write a shell scrip to print the Fibonacci series:

0, 1, 1, 2, 3, 5, ….Nth term.

Bash Script

```
read -p "Enter the limit:" f
i=0
a=0
b=1
echo -n "The series is:" $a $b
while [ $i -le `expr $f - 2` ]
do
c=`expr $a + $b`
a=$b
b=$c
echo -n " "$c
i=`expr $i + 1`
done
```

Output

```
pralay@pralay-mint:/media/pralay/PROG/Programs/SH/anit$ sh 8.sh
Enter the limit:5
The series is: 0 1 1 2 3 5

pralay@pralay-mint:/media/pralay/PROG/Programs/SH/anit$ sh 8.sh
Enter the limit:10
The series is: 0 1 1 2 3 5 8 13 21 34 55
```

9. Write shell script to check the number is prime or not.

Bash Script

```
echo -n "Enter a number: "
read num
if [ $num -lt 2 ]; then
   echo "$num is not a prime number."
   exit 0
fi
is_prime=1
for ((i = 2; i <= $num; i++)); do
   if [ $((num % i)) -eq 0 ]; then
      is_prime=0
      break
   fi
done
if [ $is_prime -eq 1 ]; then
   echo "$num is a prime number."
else
   echo "$num is not a prime number."
fi
```

Output

```
pralay@pralay-mint:/media/pralay/PROG/Programs/SH/anit$ chmod +x 9.sh
pralay@pralay-mint:/media/pralay/PROG/Programs/SH/anit$ ./9.sh
Enter a number: 25
25 is not a prime number.
pralay@pralay-mint:/media/pralay/PROG/Programs/SH/anit$ ./9.sh
Enter a number: 7
7 is not a prime number.
pralay@pralay-mint:/media/pralay/PROG/Programs/SH/anit$ ./9.sh
Enter a number: 1
1 is not a prime number.
```

10. Write a shell script to display all prime numbers from 1 to N.

Bash Script

```
read -p "Enter the limit: " limit
if ((limit <= 0)); then
   echo -e "\n\tInvalid input\n"
   exit
fi
echo -n "The prime numbers up to $limit are: "
for ((i = 1; i <= limit; i++)); do
   count=0
   for ((j = 2; j <= i / 2; j++)); do
      if ((i % j == 0)); then
         ((count++))
      fi
   done
   if ((!count)); then
      echo -n "$i "
   fi
done
echo
```

Output

```
pralay@pralay-mint:/media/pralay/PROG/Programs/SH/anit$ chmod +x 10.sh
pralay@pralay-mint:/media/pralay/PROG/Programs/SH/anit$ ./10.sh
Enter the limit: 10
The prime numbers up to 10 are: 1 2 3 5 7
pralay@pralay-mint:/media/pralay/PROG/Programs/SH/anit$ ./10.sh
Enter the limit: 30
The prime numbers up to 30 are: 1 2 3 5 7 11 13 17 19 23 29
```

11. Write a shell script to print the GCD & LCM of two numbers.

Bash Script

```
read -p "Enter the 1st number:" n1
read -p "Enter the 1st number:" n2
if [ $n1 -gt $n2 ]
then
num=$n1
den=$n2
else
num=$n2
den=$n1
fi
rem=`expr $num % $den`
while [ $rem -ne 0 ]
do
num=$den
den=$rem
rem=`expr $num % $den`
done
gcd=$den
lcm=`expr $n1 \* $n2 / $gcd`
echo "The GCD is:" $gcd
echo "The LCM is:" $lcm
```

Output

```
pralay@pralay-mint:/media/pralay/PROG/Programs/SH/anit$ sh 11.sh
Enter the 1st number:24
Enter the 1st number:36
The GCD is: 12
The LCM is: 72
pralay@pralay-mint:/media/pralay/PROG/Programs/SH/anit$ sh 11.sh
Enter the 1st number:25
Enter the 1st number:35
The GCD is: 5
The LCM is: 175
```

12. Write a shell program to convert Centigrade to Fahrenheit.

Bash Script

```
echo "Enter the centigrade value"
read c
F=`expr $c \* 9 / 5 + 32`
echo "The Fahrenheit value is:"$F
```

Output

```
pralay@pralay-mint:/media/pralay/PROG/Programs/SH/anit$ sh 12.sh
Enter the centigrade value
37
The Fahrenheit value is:98
pralay@pralay-mint:/media/pralay/PROG/Programs/SH/anit$ sh 12.sh
Enter the centigrade value
-5
The Fahrenheit value is:23
```

13. Write a shell script to calculate simple interest.

Bash Script

```
echo "Ente the principal ammount"
read p
echo "Enter the time period"
read t
echo "Enter the rate of interest"
read r
I=`expr $p \* $t \* $r / 100`
echo "The simple interest is:"$I
```

Output

```
pralay@pralay-mint:/media/pralay/PROG/Programs/SH/anit$ sh 13.sh
Ente the principal ammount
12000
Enter the time period
2
Enter the rate of interest
2
The simple interest is:480
pralay@pralay-mint:/media/pralay/PROG/Programs/SH/anit$ sh 13.sh
Ente the principal ammount
100000
Enter the time period
3
Enter the rate of interest
5
The simple interest is:15000
```

14. Write a shell script to swapping of two numbers.

Bash Script

```
echo "Enter the 1st number"
read a
echo "Enter the 2nd number"
read b
echo "\nThe values before swap\n" $a $b
c=$a
a=$b
b=$c
echo "\nThe values after swap\n" $a $b
```

Output

```
pralay@pralay-mint:/media/pralay/PROG/Programs/SH/anit$ sh 14.sh
Enter the 1st number
35
Enter the 2nd number
96

The values before swap
 35 96

The values after swap
 96 35
```

15. Write a shell script to print left pyramid pattern:

```
*
* *
* * *
```

Bash Script

```
read -p "Enter the number of line:" n
i=1
j=1
echo ""
while [ $i -le $n ]
do
    j=1
    while [ $j -le $i ]
    do
            echo -n " *"
            j=`expr $j + 1`
    done
    echo ""
    i=`expr $i + 1`
done
```

Output

```
pralay@pralay-mint:/media/pralay/PROG/Programs/SH/anit$ sh 15.sh
Enter the number of line:4

 *
 * *
 * * *
 * * * *
pralay@pralay-mint:/media/pralay/PROG/Programs/SH/anit$ sh 15.sh
Enter the number of line:3

 *
 * *
 * * *
```

16. Write a shell script to print this pattern:

```
1
1 2
1 2 3
1 2 3 4
```

Bash Script

```
read -p "Enter the number of line:" n
i=1
j=1
echo ""
while [ $i -le $n ]
do
    j=1
    while [ $j -le $i ]
    do
            echo -n " "$j
            j=`expr $j + 1`
    done
    echo ""
    i=`expr $i + 1`
done
```

Output

```
pralay@pralay-mint:/media/pralay/PROG/Programs/SH/anit$ sh 16.sh
Enter the number of line:4

 1
 1 2
 1 2 3
 1 2 3 4
pralay@pralay-mint:/media/pralay/PROG/Programs/SH/anit$ sh 16.sh
Enter the number of line:5

 1
 1 2
 1 2 3
 1 2 3 4
 1 2 3 4 5
```

17. Write a shell script to print left pyramid pattern:

```
*
* * *
* * * * *
```

Bash Script

```
read -p "Enter the number of rows for the
pyramid: " rows
for (( i=1; i<=rows; i++ ))
do
 # Print spaces
 for (( j=rows; j>i; j-- ))
 do
  echo -n " "
 done
 # Print stars
 for (( k=1; k<=((2*i-1)); k++ ))
 do
  echo -n "*"
 done
 # Move to the next line
 echo
done
```

Output

```
root@ip-172-31-14-18:/home# ./process_management.sh

Process Management Options:
1. Create a process
2. Kill a process
3. Display process scheduling information
4. Change process priority
5. Display currently running processes
6. Show background processes
7. Display all process information
8. Exit
Choose an option (1-8):1
Creating a process...
Process created with PID: 5784
```

18. Write a shell script to print left pyramid pattern:

```
*
* *
* * *
```

Bash Script

```
echo -n "Enter number of rows: "
read rows
for ((i=1; i<=rows; i++))
do
 for ((j=i; j<rows; j++))
 do
  echo -n " "
 done
 for ((k=1; k<=i; k++))
 do
  echo -n "* "
 done
 echo
done
```

Output

```
pralay@pralay-mint:/media/pralay/PROG/Programs/SH/anit$ chmod +x 18.sh
pralay@pralay-mint:/media/pralay/PROG/Programs/SH/anit$ ./18.sh
Enter number of rows: 4
   *
  * *
 * * *
* * * *
pralay@pralay-mint:/media/pralay/PROG/Programs/SH/anit$ ./18.sh
Enter number of rows: 5
    *
   * *
  * * *
 * * * *
* * * * *
```

19. Write a shell script to print left pyramid pattern:

```
1
2 3
4 5 6
7 8 9 10
```

Bash Script

```
read -p "Enter the number of rows for the
pattern: " rows
number=1
for (( i=1; i<=rows; i++ ))
do
 # Print numbers in the required pattern
 for (( j=1; j<=i; j++ ))
 do
  echo -n "$number "
  ((number++))
 done
 # Move to the next line
 echo
done
```

Output

```
pralay@pralay-mint:/media/pralay/PROG/Programs/SH/anit$ chmod +x 19.sh
pralay@pralay-mint:/media/pralay/PROG/Programs/SH/anit$ ./19.sh
Enter the number of rows for the pattern: 4
1
2 3
4 5 6
7 8 9 10
pralay@pralay-mint:/media/pralay/PROG/Programs/SH/anit$ ./19.sh
Enter the number of rows for the pattern: 5
1
2 3
4 5 6
7 8 9 10
11 12 13 14 15
```

20. Write a shell script to print left pyramid pattern:

```
* * * *
* * *
* *
*
```

Bash Script

```
read -p "Enter the number of rows for the
pattern: " rows

for (( i=rows; i>=1; i-- ))
do
  # Print stars in decreasing order
  for (( j=1; j<=i; j++ ))
  do
    echo -n "*"
  done
  # Move to the next line
  echo
done
```

Output

```
pralay@pralay-mint:/media/pralay/PROG/Programs/SH/anit$ chmod +x 20.sh
pralay@pralay-mint:/media/pralay/PROG/Programs/SH/anit$ ./20.sh
Enter the number of rows for the pattern: 4
****
***
**
*
pralay@pralay-mint:/media/pralay/PROG/Programs/SH/anit$ ./20.sh
Enter the number of rows for the pattern: 5
*****
****
***
**
*
```

21. Write a shell script to print the series: 1! + 2! + 3! + …….. + N!

Bash Script

```
# Prompt user for input
read -p "Enter the value of N: " N
# Initialize sum variable
sum=0
# Loop through numbers from 1 to N
for ((j=1; j<=N; j++)); do
   # Calculate factorial of the current number
   fact=1
   for ((k=1; k<=j; k++)); do
      fact=$((fact * k))
   done
   # Add the factorial to the sum
   sum=$((sum + fact))
   # Print the addition step
   if [[ $j -eq 1 ]]; then
      echo -n "$fact"
   else
      echo -n " + $fact"
   fi
done
# Print the final sum
echo " = $sum"
```

Output

```
pralay@pralay-mint:/media/pralay/PROG/Programs/SH/anit$ chmod +x 21.sh
pralay@pralay-mint:/media/pralay/PROG/Programs/SH/anit$ ./21.sh
Enter the value of N: 5
1 + 2 + 6 + 24 + 120 = 153
pralay@pralay-mint:/media/pralay/PROG/Programs/SH/anit$ ./21.sh
Enter the value of N: 7
1 + 2 + 6 + 24 + 120 + 720 + 5040 = 5913
```

Q. Comprehensive Process Management in Modern Operating Systems.

1. Creating Process
2. Killing Process
3. Scheduling Process
4. changing Process priorities
5. which process is running now
6. Background process

Bash Script

```
# Function to create a process
create_process() {
   echo "Creating a process..."
   sleep 100 & # This creates a dummy process that runs in the background
   echo "Process created with PID: $!"
}
# Function to kill a process
kill_process() {
   echo "Enter the PID of the process to kill:"
   read pid
   kill $pid
   if [ $? -eq 0 ]; then
      echo "Process $pid killed successfully."
   else
      echo "Failed to kill process $pid."
   fi
}
# Function to display process scheduling information
schedule_process() {
   echo "Process scheduling information:"
   ps -eo pid,comm,pri,nice,state,etime --sort=-pri | head -n 10
}
# Function to change process priority
change_priority() {
   echo "Enter the PID of the process to change priority:"
   read pid
   echo "Enter the new priority (-20 to 19, lower is higher priority):"
   read priority
   renice $priority -p $pid
   if [ $? -eq 0 ]; then
      echo "Priority of process $pid changed to $priority."
   else
      echo "Failed to change priority of process $pid."
   fi
}

# Function to display the currently running process
current_process() {
   echo "Currently running processes:"
   ps -eo pid,comm,user,state,%cpu,%mem,etime --sort=-%cpu | head -n 10
}
```

Bash Script

```
# Function to handle background processes
background_process() {
   echo "Background processes:"
   jobs -l
}
# Function to show all information about processes
display_all_info() {
   echo "All process information:"
   ps -eo pid,ppid,comm,user,pri,nice,state,%cpu,%mem,etime,start,time --sort=-%cpu
}

# Menu to perform the actions
while true; do
   echo "\nProcess Management Options:"
   echo "1. Create a process"
   echo "2. Kill a process"
   echo "3. Display process scheduling information"
   echo "4. Change process priority"
   echo "5. Display currently running processes"
   echo "6. Show background processes"
   echo "7. Display all process information"
   echo "8. Exit"
   echo "Choose an option (1-8):"
   read choice

   case $choice in
   1)
      create_process
      ;;
   2)
      kill_process
      ;;
   3)
      schedule_process
      ;;
   4)
      change_priority
      ;;
   5)
      current_process
      ;;
   6)
      background_process
      ;;
   7)
      display_all_info
      ;;
   8)
      echo "Exiting..."
      break
      ;;
   *)
      echo "Invalid option. Please choose again."
      ;;
   esac
done
```

Outputs

```
root@ip-172-31-14-18:/home# ./process_management.sh

Process Management Options:
1. Create a process
2. Kill a process
3. Display process scheduling information
4. Change process priority
5. Display currently running processes
6. Show background processes
7. Display all process information
8. Exit
Choose an option (1-8):1
Creating a process...
Process created with PID: 5784
```

```
Process Management Options:
1. Create a process
2. Kill a process
3. Display process scheduling information
4. Change process priority
5. Display currently running processes
6. Show background processes
7. Display all process information
8. Exit
Choose an option (1-8):2
Enter the PID of the process to kill:5784
Process 5784 killed successfully.
```

```
Process Management Options:
1. Create a process
2. Kill a process
3. Display process scheduling information
4. Change process priority
5. Display currently running processes
6. Show background processes
7. Display all process information
8. Exit
Choose an option (1-8):3
Process scheduling information:
    PID COMMAND         PRI  NI S     ELAPSED
     30 khugepaged        0  19 S       27:58
     29 ksmd             14   5 S       27:58
   1024 rtkit-daemon     18   1 S       27:44
      1 systemd          19   0 S       27:58
      2 kthreadd         19   0 S       27:58
      3 pool_workqueue_  19   0 S       27:58
     13 rcu_tasks_rude_  19   0 I       27:58
     14 rcu_tasks_trace  19   0 I       27:58
     15 ksoftirqd/0      19   0 S       27:58
```

Outputs

```
Process Management Options:
1. Create a process
2. Kill a process
3. Display process scheduling information
4. Change process priority
5. Display currently running processes
6. Show background processes
7. Display all process information
8. Exit
Choose an option (1-8):4
Enter the PID of the process to change priority:5814
Enter the new priority (-20 to 19, lower is higher priority):10
5814 (process ID) old priority 0, new priority 10
Priority of process 5814 changed to 10.
```

```
Process Management Options:
1. Create a process
2. Kill a process
3. Display process scheduling information
4. Change process priority
5. Display currently running processes
6. Show background processes
7. Display all process information
8. Exit
Choose an option (1-8):6
Background processes:
[1]+  5814 Done                    sleep 100
```

```
Process Management Options:
1. Create a process
2. Kill a process
3. Display process scheduling information
4. Change process priority
5. Display currently running processes
6. Show background processes
7. Display all process information
8. Exit
Choose an option (1-8):5
Currently running processes:
    PID COMMAND         USER     S %CPU %MEM     ELAPSED
   1592 snapd           root     S  0.7  3.3       26:26
   1544 fwupd           root     S  0.1  3.9       26:29
      1 systemd         root     S  0.0  1.4       34:33
   1033 lightdm-gtk-gre lightdm  S  0.0 10.6       34:19
    765 amazon-ssm-agen root     S  0.0  1.9       34:22
    899 Xorg            root     S  0.0  6.8       34:21
    190 systemd-udevd   root     S  0.0  0.8       34:30
    125 systemd-journal root     S  0.0  1.5       34:31
    516 dbus-daemon     message+ S  0.0  0.6       34:24
```