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

```
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
```

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

```
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
```

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

```
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
```

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

```
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
```

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

```
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
```

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

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

```
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
```

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

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

```
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
```

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

```
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
```

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."
```

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
```

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
    exit
```

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
```

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
```

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
```

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

Bash Script

```
echo "Enter the centigrade value"
```

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
```

13. Write a shell script to calculate simple interest.

Bash Script

```
echo "Enter the principal ammount"
read p
```

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
```

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

Bash Script

```
read a
```

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
```

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

```
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
```

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
```

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
```

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++ ))
```

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

```
read -p "Enter the number of rows for the
pyramid: " rows
for (( i=1; i<=rows; i++ ))
do
```

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
```

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
```

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
```

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-- ))
```

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
```