USMAN INSTITUTE OF

TECHNOLOGY

Affiliated with NED University of Engineering & Technology,

Karachi

Department of
Computer Science
Fall-2023

CS312 – Operating
System Complete

Lab Manual

Name of Student	**Muhammad Asfiyan**
Student ID	**21B-111-SE**
Marks Obtained	
Remarks	
Signature	

Instructor: Maham Ashraf

Lab#	Date	Objective	Obtained Marks	Signature
1		Executing some of the most frequently used Linux commands		
2		Managing Files & Directories in Linux		
3		Programming using Shell Scripting.		
4		Focusing on the usage of the test command and conditional statements.		
5		Focusing on the usage of iteration statements and functions in shell programming		
6		Understanding Process. ● Process creation in Linux using System Calls. ● Fork() method. ● Zombie and Orphan processes.		
7		Understanding Threads. ● Threads vs. Processes. ● Multithreaded Programming using Python.		
8		Simulation of FCFS CPU scheduling algorithm. Simulation of SJF CPU scheduling algorithm.		
9		Simulation of Round Robin CPU scheduling algorithm. Simulation of Priority CPU scheduling algorithm.		
10		Implementation of Semaphore Mechanism. Solving producer-consumer (Classical Problem) problem in Python using semaphores.		
11		Inter process communication (IPC) using Pipe. • Multiprocessing in Python. • Implement Pipe using *os* and *multiprocessing* module in Python.		
12		Inter process communication (IPC) using Shared Memory. • Implement Shared Memory using multiprocessing module in Python.		

13		Implementation of Deadlock Avoidance Mechanism (Banker’s Algorithm).		
14		**Open Ended Lab** **Objective:** Write a GUI based Shell script that behaves like an Operating System.		

USMAN INSTITUTE OF TECHNOLOGY

Affiliated with NED University of Engineering & Technology, Karachi

Department of Computer Science Fall-2023

CS312 – Operating System Lab #1

Objective:

Executing some of the most frequently used Linux commands

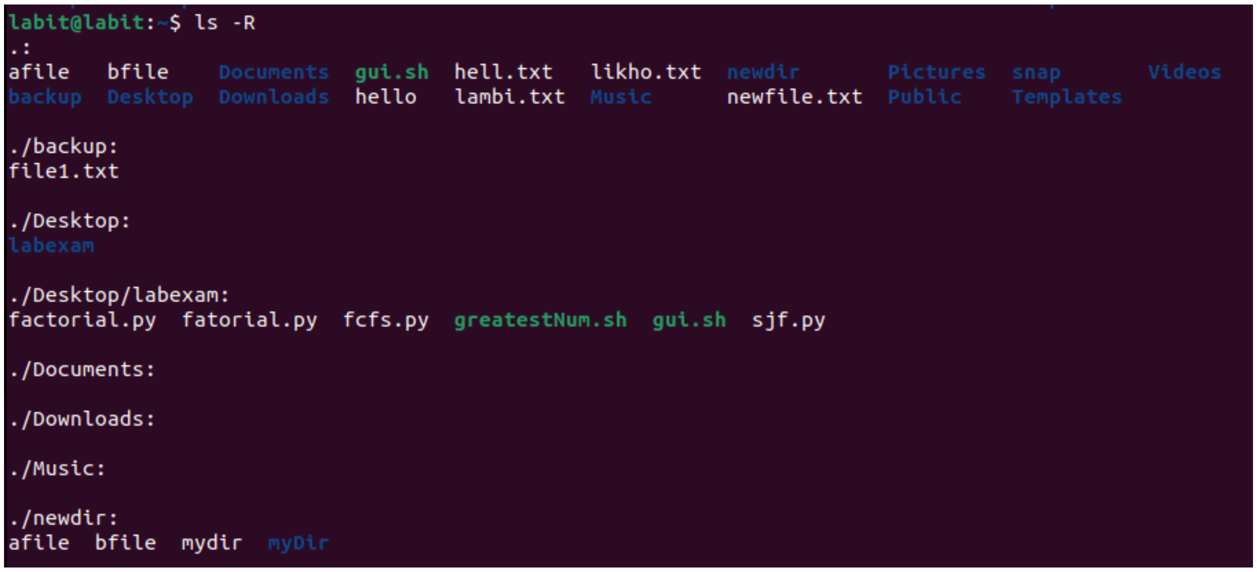
To understand and use basic utilities/Commands of UNIX/LINUX.

Name of Student	**Muhammad Asfiyan**
Student ID	**21B-111-SE**
Marks Obtained	
Remarks	
Signature	

Q1:

Command : **ls -R**

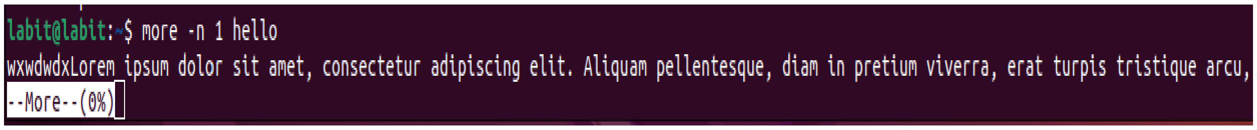
For subdirectories



```
labit@labit:~$ ls -R
.:
afile   bfile    Documents  gui.sh  hell.txt   likho.txt  newdir       Pictures  snap       Videos
backup  Desktop  Downloads  hello   lambi.txt  Music      newfile.txt  Public    Templates

./backup:
file1.txt

./Desktop:
labexam

./Desktop/labexam:
factorial.py  fatorial.py  fcfs.py  greatestNum.sh  gui.sh  sjf.py

./Documents:

./Downloads:

./Music:

./newdir:
afile  bfile  mydir  myDir
```

Q2:

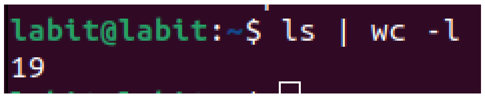
command : **more -n 1 hello**



```
labit@labit:~$ more -n 1 hello
wxwdwdxLorem ipsum dolor sit amet, consectetur adipiscing elit. Aliquam pellentesque, diam in pretium viverra, erat turpis tristique arcu,
--More--(0%)
```

Q3:

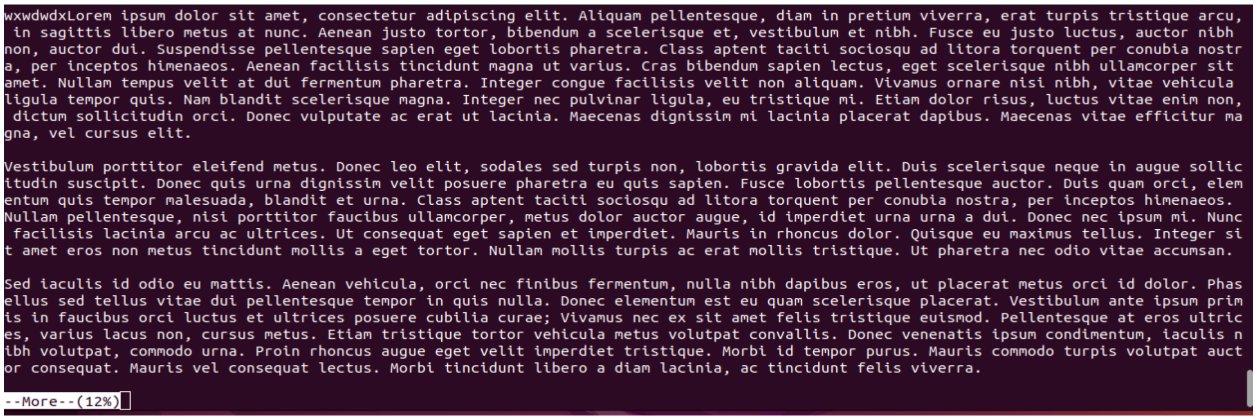
Command: **ls | wc -l**



```
labit@labit:~$ ls | wc -l
19
```

Q4:

Command: **more -c hello**



```
wxwdwdxLorem ipsum dolor sit amet, consectetur adipiscing elit. Aliquam pellentesque, diam in pretium viverra, erat turpis tristique arcu,
 in sagittis libero metus at nunc. Aenean justo tortor, bibendum a scelerisque et, vestibulum et nibh. Fusce eu justo luctus, auctor nibh
non, auctor dui. Suspendisse pellentesque sapien eget lobortis pharetra. Class aptent taciti sociosqu ad litora torquent per conubia nostr
a, per inceptos himenaeos. Aenean facilisis tincidunt magna ut varius. Cras bibendum sapien lectus, eget scelerisque nibh ullamcorper sit
amet. Nullam tempus velit at dui fermentum pharetra. Integer congue facilisis velit non aliquam. Vivamus ornare nisi nibh, vitae vehicula
ligula tempor quis. Nam blandit scelerisque magna. Integer nec pulvinar ligula, eu tristique mi. Etiam dolor risus, luctus vitae enim non,
 dictum sollicitudin orci. Donec vulputate ac erat ut lacinia. Maecenas dignissim mi lacinia placerat dapibus. Maecenas vitae efficitur ma
gna, vel cursus elit.

Vestibulum porttitor eleifend metus. Donec leo elit, sodales sed turpis non, lobortis gravida elit. Duis scelerisque neque in augue sollic
itudin suscipit. Donec quis urna dignissim velit posuere pharetra eu quis sapien. Fusce lobortis pellentesque auctor. Duis quam orci, elem
entum quis tempor malesuada, blandit et urna. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos himenaeos.
Nullam pellentesque, nisi porttitor faucibus ullamcorper, metus dolor auctor augue, id imperdiet urna urna a dui. Donec nec ipsum mi. Nunc
 facilisis lacinia arcu ac ultrices. Ut consequat eget sapien et imperdiet. Mauris in rhoncus dolor. Quisque eu maximus tellus. Integer si
t amet eros non metus tincidunt mollis a eget tortor. Nullam mollis turpis ac erat mollis tristique. Ut pharetra nec odio vitae accumsan.

Sed iaculis id odio eu mattis. Aenean vehicula, orci nec finibus fermentum, nulla nibh dapibus eros, ut placerat metus orci id dolor. Phas
ellus sed tellus vitae dui pellentesque tempor in quis nulla. Donec elementum est eu quam scelerisque placerat. Vestibulum ante ipsum prim
is in faucibus orci luctus et ultrices posuere cubilia curae; Vivamus nec ex sit amet felis tristique euismod. Pellentesque at eros ultric
es, varius lacus non, cursus metus. Etiam tristique tortor vehicula metus volutpat convallis. Donec venenatis ipsum condimentum, iaculis n
ibh volutpat, commodo urna. Proin rhoncus augue eget velit imperdiet tristique. Morbi id tempor purus. Mauris commodo turpis volutpat auct
or consequat. Mauris vel consequat lectus. Morbi tincidunt libero a diam lacinia, ac tincidunt felis viverra.

--More--(12%)
```

USMAN INSTITUTE OF TECHNOLOGY

Affiliated with NED University of Engineering & Technology, Karachi

Department of Computer Science Fall-2023

CS312 – Operating System Lab #2

Objective:

Executing some of the most frequently used Linux commands

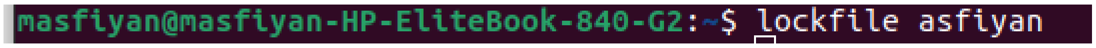
To understand and use basic utilities/Commands of UNIX/LINUX.

Name of Student	**Muhammad Asfiyan**
Student ID	**21B-111-SE**
Marks Obtained	
Remarks	
Signature	

Question 1:

1. LockFile

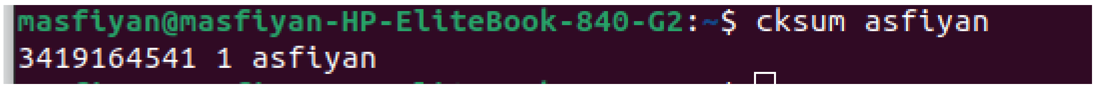
Lockfile is a powerful tool that can help you to prevent conditions and protect your data.



```
masfiyan@masfiyan-HP-EliteBook-840-G2:~$ lockfile asfiyan
```

2. cksum

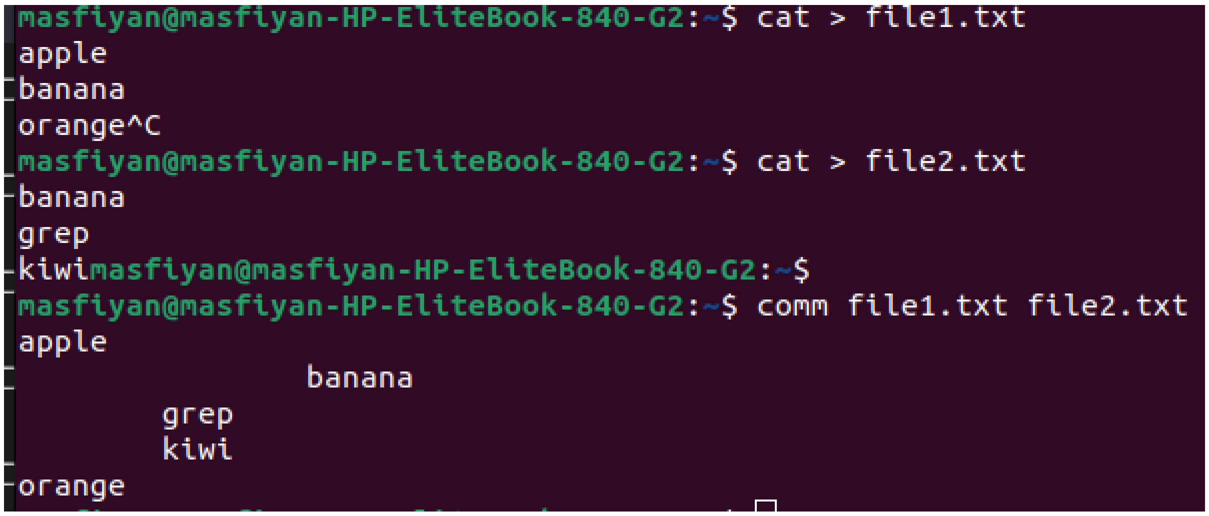
the `cksum` command is used to generate a checksum for files. The checksum is a short fixed-size numerical value calculated from the contents of the file. It is often used to verify the integrity of files and detect errors or corruption.



```
masfiyan@masfiyan-HP-EliteBook-840-G2:~$ cksum asfiyan
3419164541 1 asfiyan
```

3. comm

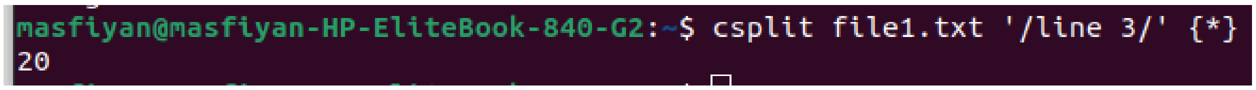
The `comm` command is used to compare two sorted files line by line



```
masfiyan@masfiyan-HP-EliteBook-840-G2:~$ cat > file1.txt
apple
banana
orange^C
masfiyan@masfiyan-HP-EliteBook-840-G2:~$ cat > file2.txt
banana
grep
kiwimasfiyan@masfiyan-HP-EliteBook-840-G2:~$
masfiyan@masfiyan-HP-EliteBook-840-G2:~$ comm file1.txt file2.txt
apple
                banana
        grep
        kiwi
orange
```

4. Csplite

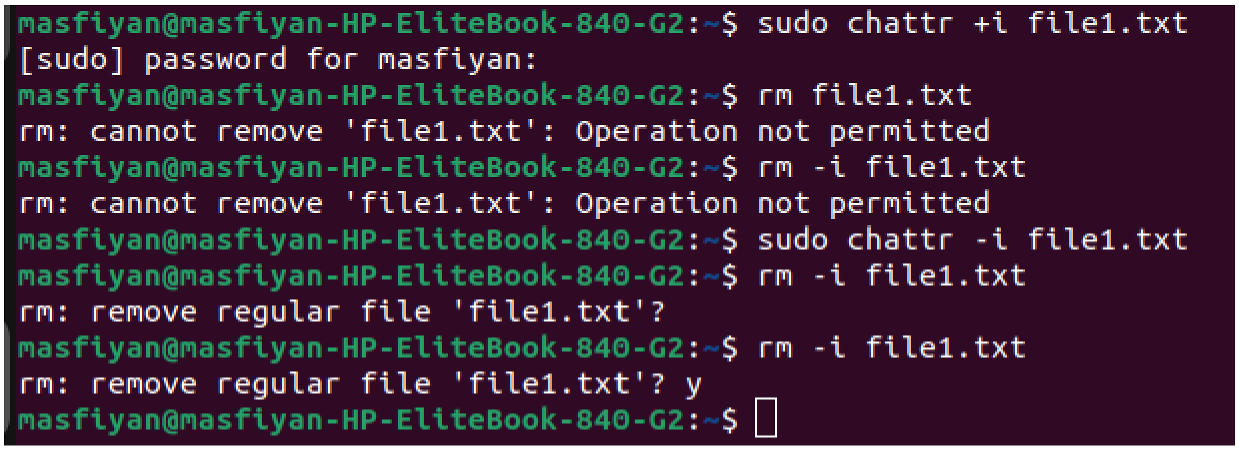
The `csplit` command is used to split a file into sections based on context lines or regular expressions. It is particularly useful when you want to divide a file into smaller parts based on specific patterns or conditions.



```
masfiyan@masfiyan-HP-EliteBook-840-G2:~$ csplit file1.txt '/line 3/' {*}
20
```

5. Chattr

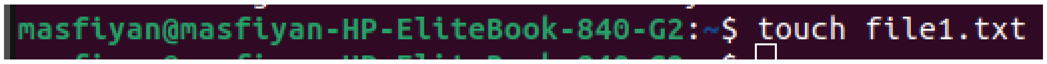
The `chattr` command is used in Linux and Unix-like operating systems to change the attributes of a file on an ext2/ext3/ext4 file system. These attributes provide additional control over how files are stored and accessed. The `chattr` command is often used with the root (superuser) privileges.



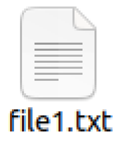
```
masfiyan@masfiyan-HP-EliteBook-840-G2:~$ sudo chattr +i file1.txt
[sudo] password for masfiyan:
masfiyan@masfiyan-HP-EliteBook-840-G2:~$ rm file1.txt
rm: cannot remove 'file1.txt': Operation not permitted
masfiyan@masfiyan-HP-EliteBook-840-G2:~$ rm -i file1.txt
rm: cannot remove 'file1.txt': Operation not permitted
masfiyan@masfiyan-HP-EliteBook-840-G2:~$ sudo chattr -i file1.txt
masfiyan@masfiyan-HP-EliteBook-840-G2:~$ rm -i file1.txt
rm: remove regular file 'file1.txt'?
masfiyan@masfiyan-HP-EliteBook-840-G2:~$ rm -i file1.txt
rm: remove regular file 'file1.txt'? y
masfiyan@masfiyan-HP-EliteBook-840-G2:~$
```

6. Touch:

The touch command is used to create a new empty file or update the timestamp (access and modification time) of an existing file. It&#39;s a simple way to create files without any Content.



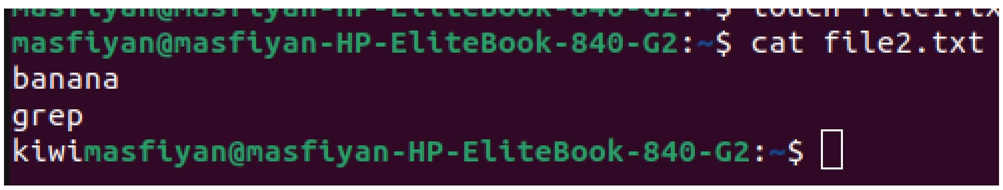
```
masfiyan@masfiyan-HP-EliteBook-840-G2:~$ touch file1.txt
```



Question 2

1. cat ch1

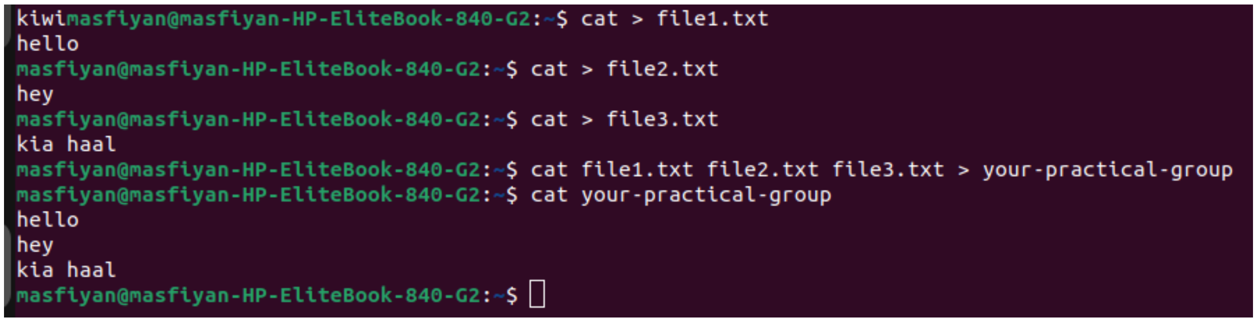
Cat ch1 is used to display the content of file in terminal (read and display the contents of text files)



```
masfiyan@masfiyan-HP-EliteBook-840-G2:~$ cat file2.txt
banana
grep
kiwimasfiyan@masfiyan-HP-EliteBook-840-G2:~$
```

2. cat ch1 ch2 ch3 > “your-practical-group”

If ch1,ch2 and ch3 are already created before so now it will be concate all these three file in one and name as “your-practical-group”



```
kiwimasfiyan@masfiyan-HP-EliteBook-840-G2:~$ cat > file1.txt
hello
masfiyan@masfiyan-HP-EliteBook-840-G2:~$ cat > file2.txt
hey
masfiyan@masfiyan-HP-EliteBook-840-G2:~$ cat > file3.txt
kia haal
masfiyan@masfiyan-HP-EliteBook-840-G2:~$ cat file1.txt file2.txt file3.txt > your-practical-group
masfiyan@masfiyan-HP-EliteBook-840-G2:~$ cat your-practical-group
hello
hey
kia haal
masfiyan@masfiyan-HP-EliteBook-840-G2:~$
```

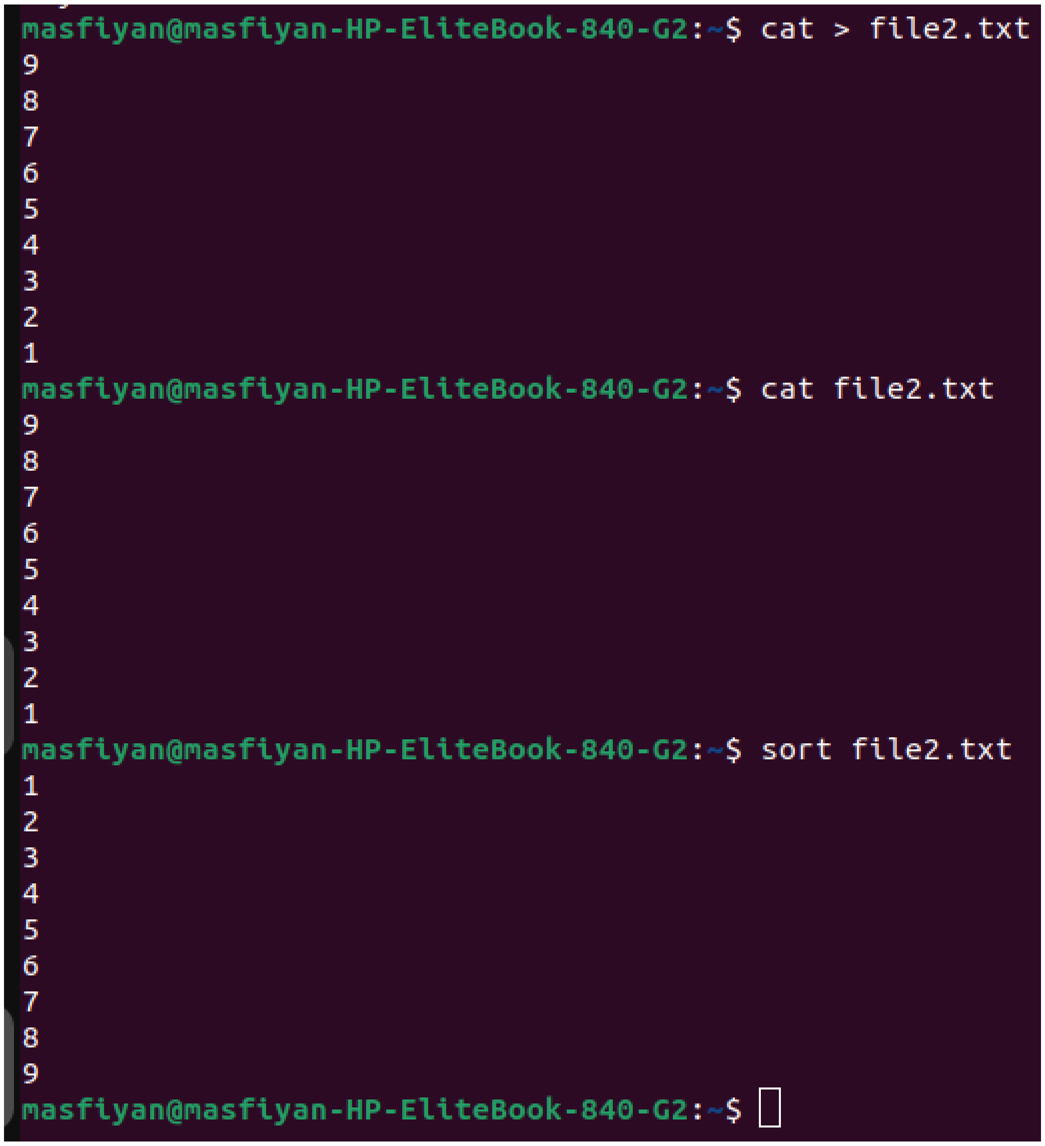
Question 3:

1. Cpio:

The `cpio` command is used for copying files into or out of a cpio or tar archive. It's often used in combination with `find` to perform complex file manipulations.

2. Sort:

The `sort` command is used for sorting lines of text files or input from a pipeline. It's a versatile tool that can be used for various text manipulation tasks.



```
masfiyan@masfiyan-HP-EliteBook-840-G2:~$ cat > file2.txt
9
8
7
6
5
4
3
2
1
masfiyan@masfiyan-HP-EliteBook-840-G2:~$ cat file2.txt
9
8
7
6
5
4
3
2
1
masfiyan@masfiyan-HP-EliteBook-840-G2:~$ sort file2.txt
1
2
3
4
5
6
7
8
9
masfiyan@masfiyan-HP-EliteBook-840-G2:~$ 
```

3. `fuser`

The `fuser` command is used to identify processes using a file or a socket. It can help you find which processes are currently accessing a specific file or socket.

4. `file`

The `file` command is used to determine the type of a file. It examines the file's contents and provides information about its type (e.g., text, executable, image)

Question 4:

ANS: both `cp` and `cpio` deal with file operations, `cp` is a simple file copy utility, and `cpio` is more focused on creating or extracting archives, often in combination with other commands for more complex file manipulation tasks.

Question 5:

ANS:

777	775
Owner has read write and execute permission	Owner has read write and execute permission
group has read write and execute permission	group has read write and execute permission
others has read write and execute permission	others has read and execute permission

USMAN INSTITUTE OF TECHNOLOGY

Affiliated with NED University of Engineering & Technology, Karachi

Department of Computer Science Fall-2023

CS312 – Operating System Lab #3

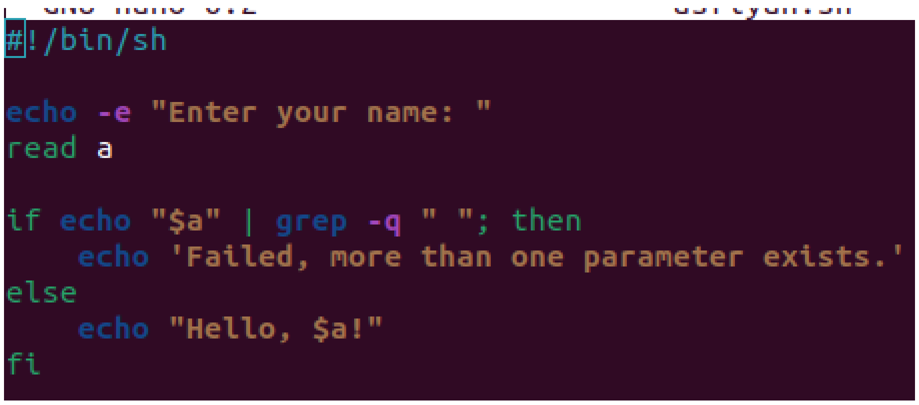
Objective:

Executing some of the most frequently used Linux commands

To understand and use basic utilities/Commands of UNIX/LINUX.

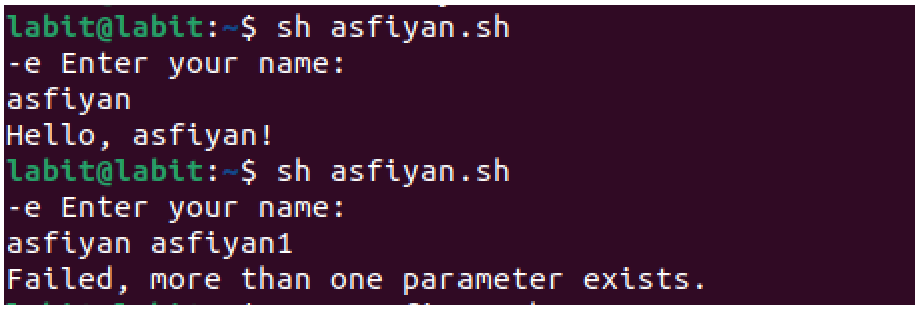
Name of Student	**Muhammad Asfiyan**
Student ID	**21B-111-SE**
Marks Obtained	
Remarks	
Signature	

Question 1:



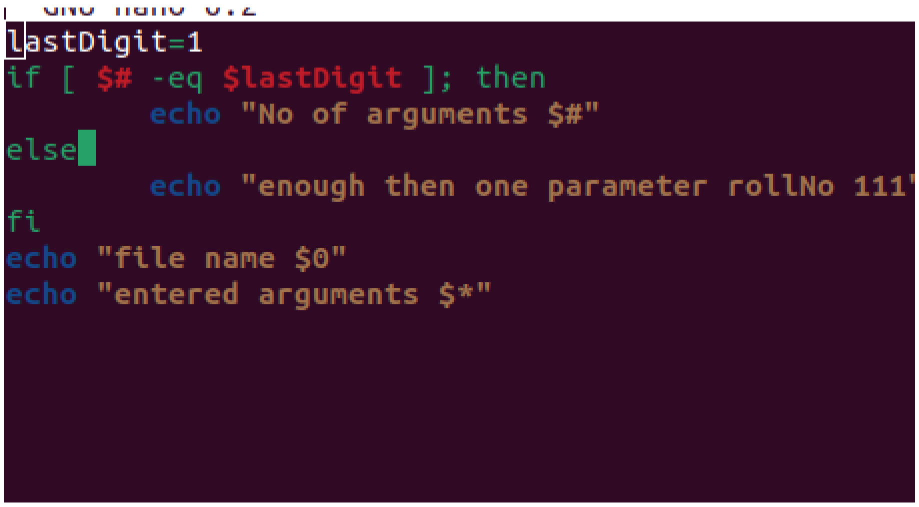
```
#!/bin/sh

echo -e "Enter your name: "
read a

if echo "$a" | grep -q " "; then
    echo 'Failed, more than one parameter exists.'
else
    echo "Hello, $a!"
fi
```

Output:



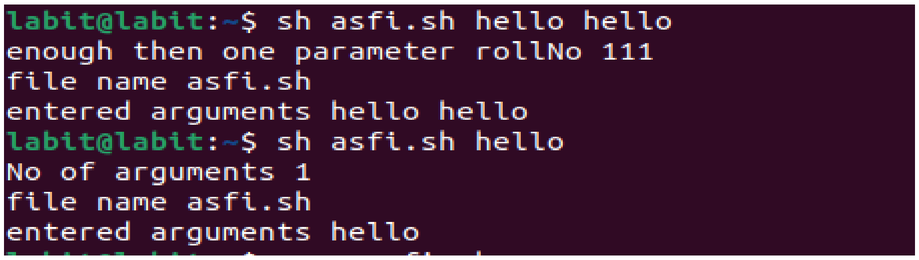
```
labit@labit:~$ sh asfiyan.sh
-e Enter your name:
asfiyan
Hello, asfiyan!
labit@labit:~$ sh asfiyan.sh
-e Enter your name:
asfiyan asfiyan1
Failed, more than one parameter exists.
```

Question 2:



```
lastDigit=1
if [ $# -eq $lastDigit ]; then
        echo "No of arguments $#"
else
        echo "enough then one parameter rollNo 111"
fi
echo "file name $0"
echo "entered arguments $*"
```

Output:



```
labit@labit:~$ sh asfi.sh hello hello
enough then one parameter rollNo 111
file name asfi.sh
entered arguments hello hello
labit@labit:~$ sh asfi.sh hello
No of arguments 1
file name asfi.sh
entered arguments hello
```

USMAN INSTITUTE OF TECHNOLOGY

Affiliated with NED University of Engineering & Technology, Karachi

Department of Computer Science Fall-2023

CS312 – Operating System Lab #4

Objective:

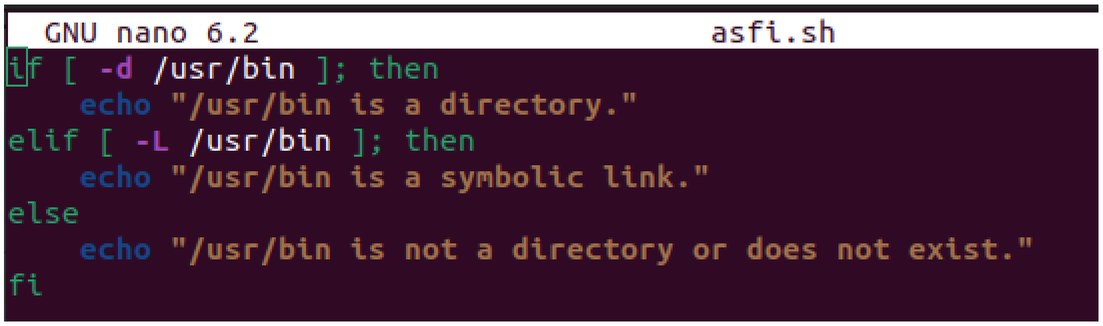
Executing some of the most frequently used Linux commands

To understand and use basic utilities/Commands of UNIX/LINUX.

Name of Student	**Muhammad Asfiyan**
Student ID	**21B-111-SE**
Marks Obtained	
Remarks	
Signature	

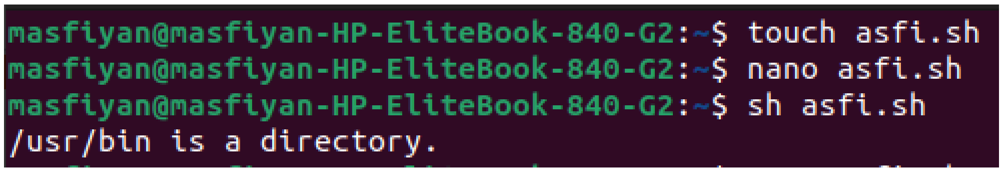
Ques: 1

Shell:



```
  GNU nano 6.2                                    asfi.sh
if [ -d /usr/bin ]; then
    echo "/usr/bin is a directory."
elif [ -L /usr/bin ]; then
    echo "/usr/bin is a symbolic link."
else
    echo "/usr/bin is not a directory or does not exist."
fi
```

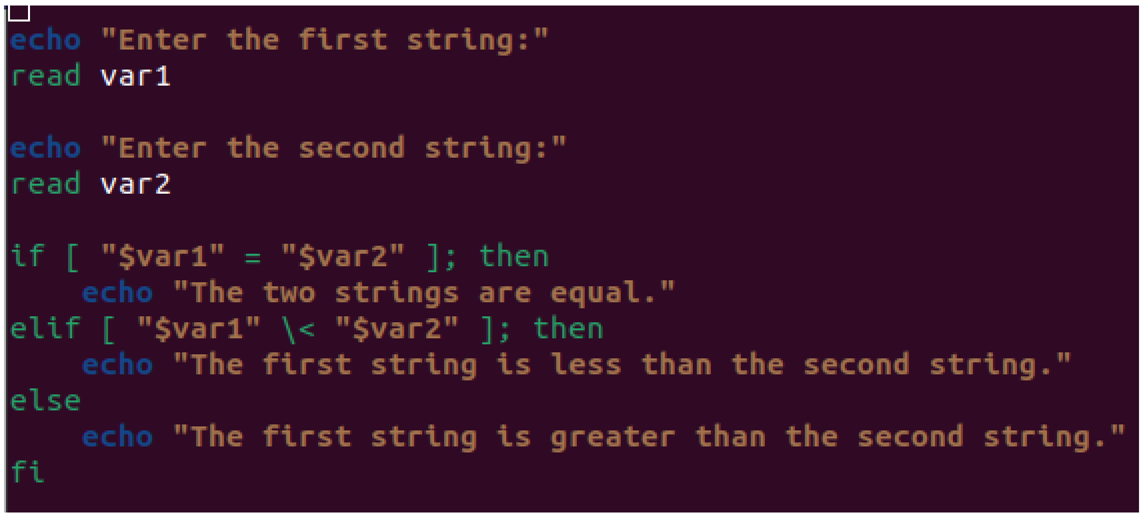
Output:



```
masfiyan@masfiyan-HP-EliteBook-840-G2:~$ touch asfi.sh
masfiyan@masfiyan-HP-EliteBook-840-G2:~$ nano asfi.sh
masfiyan@masfiyan-HP-EliteBook-840-G2:~$ sh asfi.sh
/usr/bin is a directory.
```

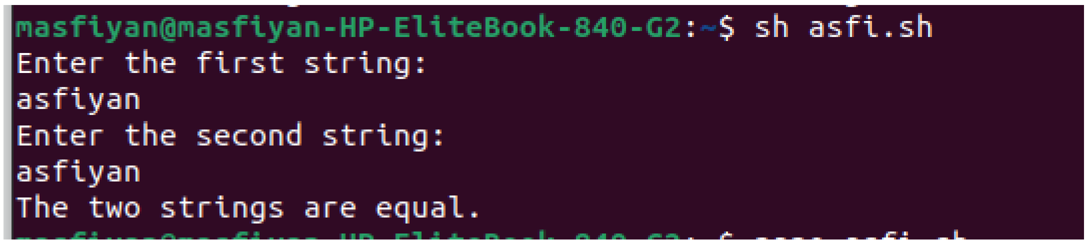
Ques: 2

Shell:



```
echo "Enter the first string:"
read var1

echo "Enter the second string:"
read var2

if [ "$var1" = "$var2" ]; then
    echo "The two strings are equal."
elif [ "$var1" \< "$var2" ]; then
    echo "The first string is less than the second string."
else
    echo "The first string is greater than the second string."
fi
```

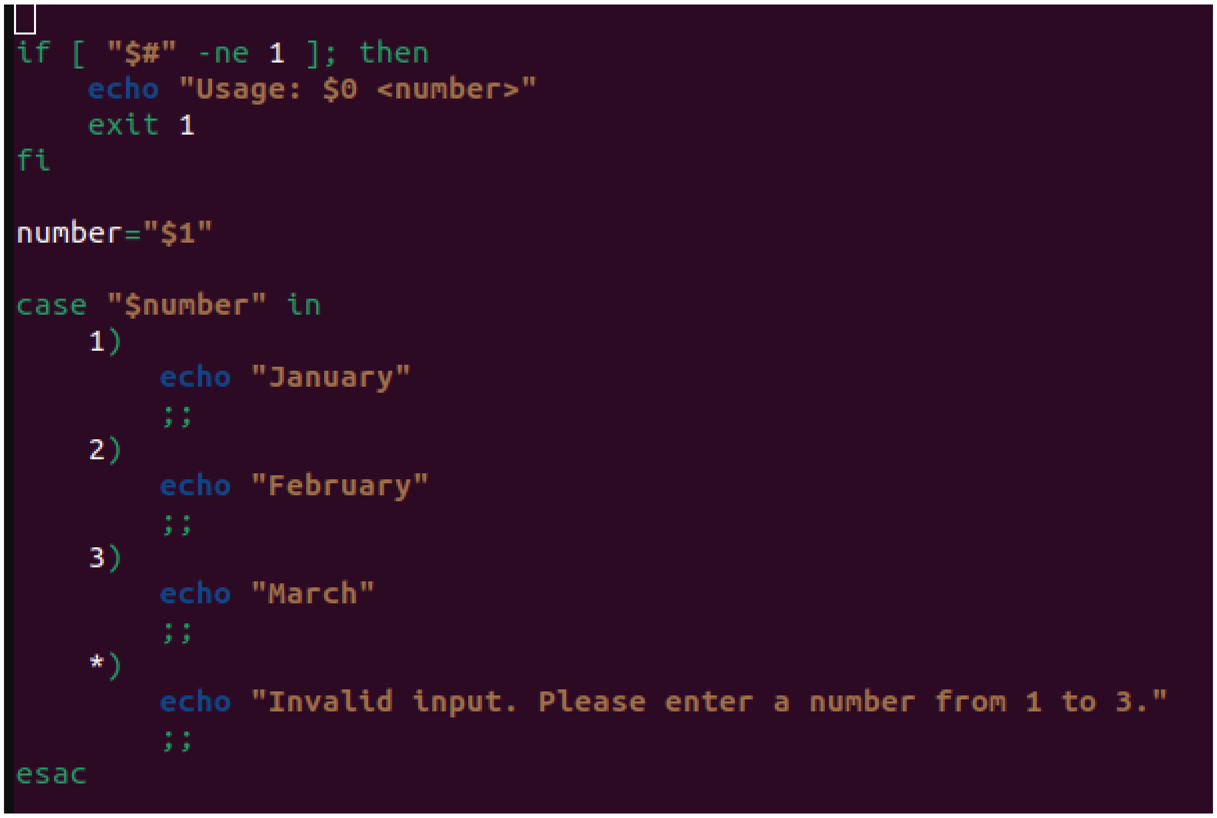
Output:



```
masfiyan@masfiyan-HP-EliteBook-840-G2:~$ sh asfi.sh
Enter the first string:
asfiyan
Enter the second string:
asfiyan
The two strings are equal.
```

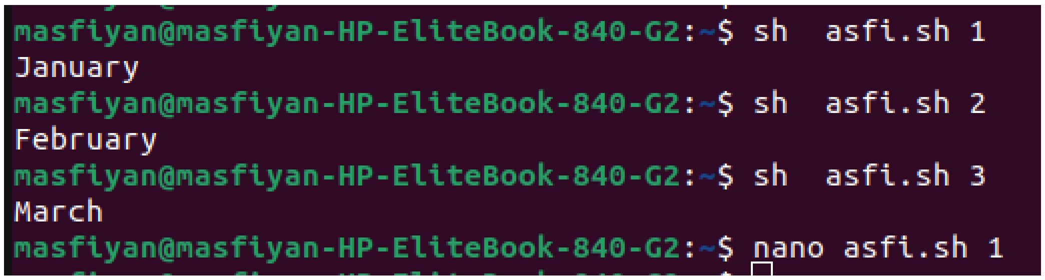
Ques: 3

Shell:



```
if [ "$#" -ne 1 ]; then
    echo "Usage: $0 <number>"
    exit 1
fi

number="$1"

case "$number" in
    1)
        echo "January"
        ;;
    2)
        echo "February"
        ;;
    3)
        echo "March"
        ;;
    *)
        echo "Invalid input. Please enter a number from 1 to 3."
        ;;
esac
```

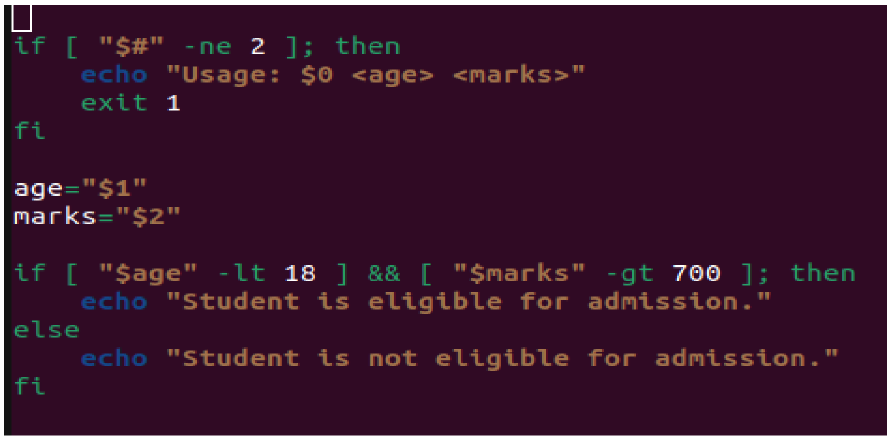
Output:



```
masfiyan@masfiyan-HP-EliteBook-840-G2:~$ sh  asfi.sh 1
January
masfiyan@masfiyan-HP-EliteBook-840-G2:~$ sh  asfi.sh 2
February
masfiyan@masfiyan-HP-EliteBook-840-G2:~$ sh  asfi.sh 3
March
masfiyan@masfiyan-HP-EliteBook-840-G2:~$ nano asfi.sh 1
```

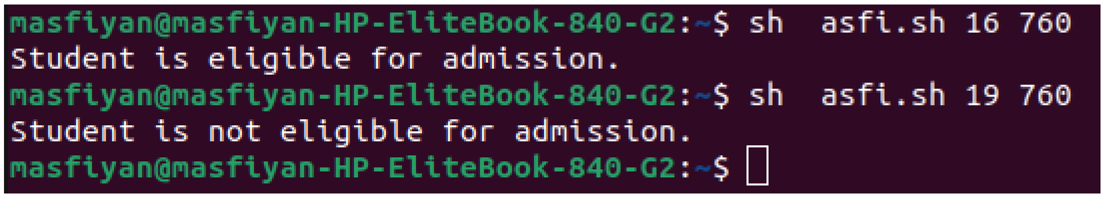
Ques: 4

Shell:



```
if [ "$#" -ne 2 ]; then
    echo "Usage: $0 <age> <marks>"
    exit 1
fi

age="$1"
marks="$2"

if [ "$age" -lt 18 ] && [ "$marks" -gt 700 ]; then
    echo "Student is eligible for admission."
else
    echo "Student is not eligible for admission."
fi
```

Output:



```
masfiyan@masfiyan-HP-EliteBook-840-G2:~$ sh  asfi.sh 16 760
Student is eligible for admission.
masfiyan@masfiyan-HP-EliteBook-840-G2:~$ sh  asfi.sh 19 760
Student is not eligible for admission.
masfiyan@masfiyan-HP-EliteBook-840-G2:~$ 
```

USMAN INSTITUTE OF TECHNOLOGY

Affiliated with NED University of Engineering & Technology, Karachi

Department of Computer Science Fall-2023

CS312 – Operating System Lab #5

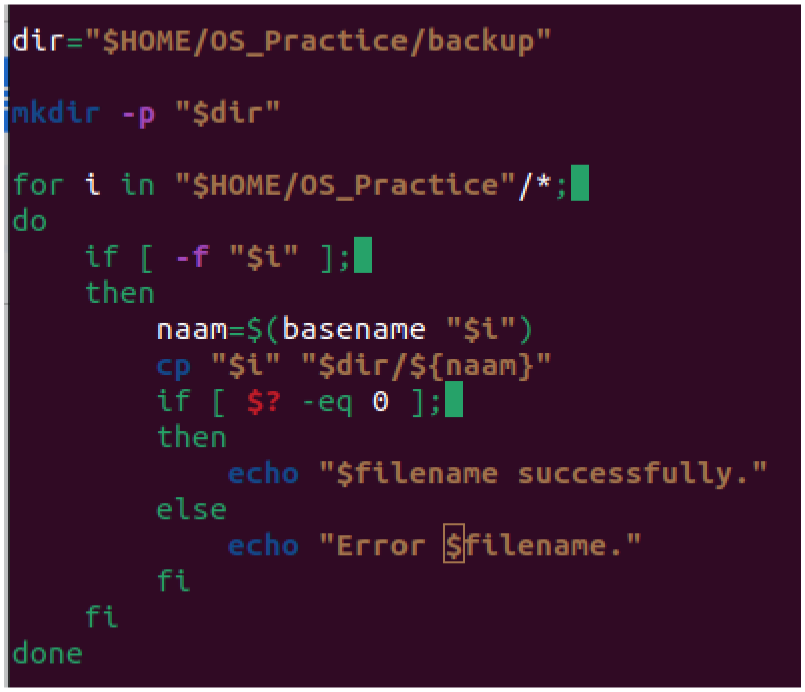
Objective:

Executing some of the most frequently used Linux commands

To understand and use basic utilities/Commands of UNIX/LINUX.

Name of Student	**Muhammad Asfiyan**
Student ID	**21B-111-SE**
Marks Obtained	
Remarks	
Signature	

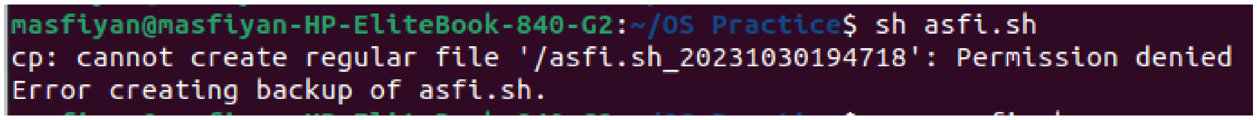
Question 1



```
dir="$HOME/OS_Practice/backup"

mkdir -p "$dir"

for i in "$HOME/OS_Practice"/*;
do
    if [ -f "$i" ];
    then
        naam=$(basename "$i")
        cp "$i" "$dir/${naam}"
        if [ $? -eq 0 ];
        then
            echo "$filename successfully."
        else
            echo "Error $filename."
        fi
    fi
done
```

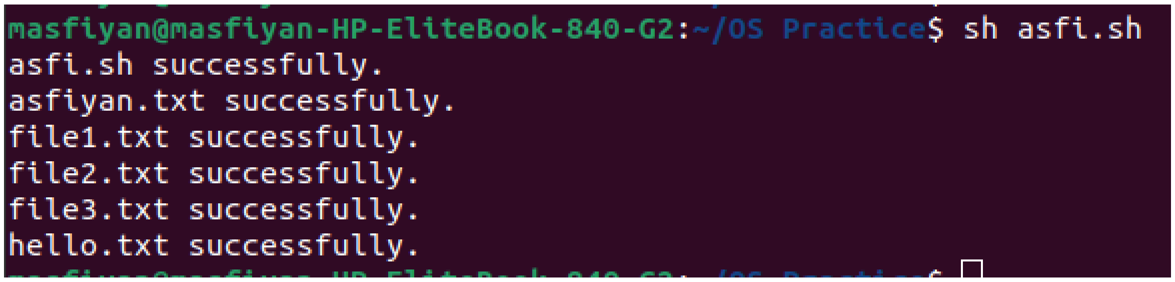
Output:

If there's no file exit



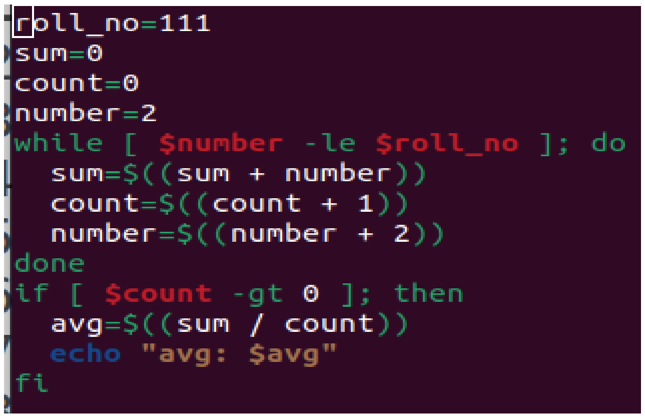
```
masfiyan@masfiyan-HP-EliteBook-840-G2:~/OS_Practice$ sh asfi.sh
cp: cannot create regular file '/asfi.sh_20231030194718': Permission denied
Error creating backup of asfi.sh.
```

If file exist



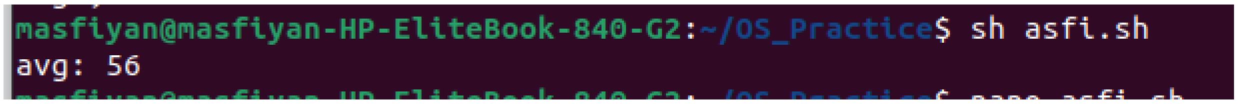
```
masfiyan@masfiyan-HP-EliteBook-840-G2:~/OS_Practice$ sh asfi.sh
asfi.sh successfully.
asfiyan.txt successfully.
file1.txt successfully.
file2.txt successfully.
file3.txt successfully.
hello.txt successfully.
```

Question 2:



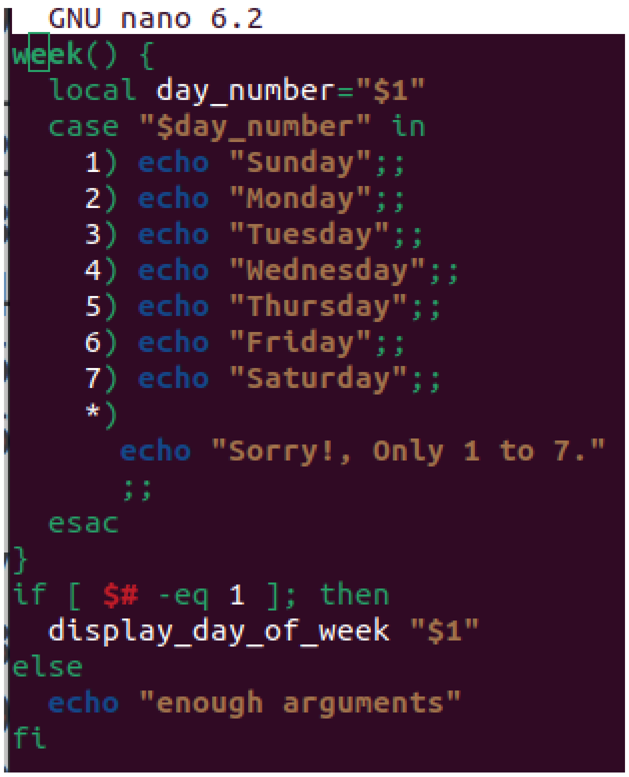
```
roll_no=111
sum=0
count=0
number=2
while [ $number -le $roll_no ]; do
  sum=$((sum + number))
  count=$((count + 1))
  number=$((number + 2))
done
if [ $count -gt 0 ]; then
  avg=$((sum / count))
  echo "avg: $avg"
fi
```

Output:



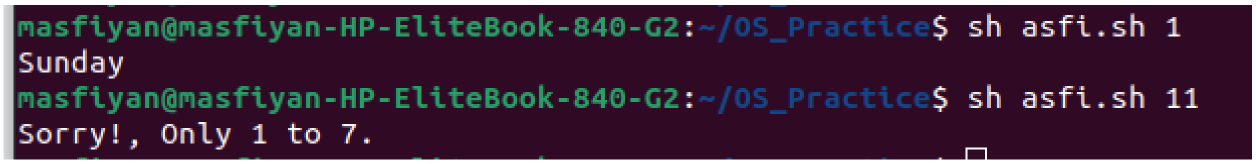
```
masfiyan@masfiyan-HP-EliteBook-840-G2:~/OS_Practice$ sh asfi.sh
avg: 56
```

Question 3:



```
  GNU nano 6.2
week() {
  local day_number="$1"
  case "$day_number" in
    1) echo "Sunday";;
    2) echo "Monday";;
    3) echo "Tuesday";;
    4) echo "Wednesday";;
    5) echo "Thursday";;
    6) echo "Friday";;
    7) echo "Saturday";;
    *)
      echo "Sorry!, Only 1 to 7."
      ;;
  esac
}
if [ $# -eq 1 ]; then
  display_day_of_week "$1"
else
  echo "enough arguments"
fi
```

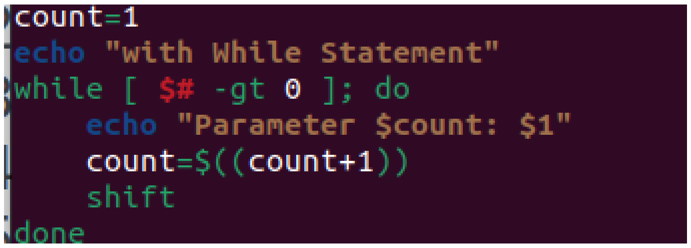
Output:



```
masfiyan@masfiyan-HP-EliteBook-840-G2:~/OS_Practice$ sh asfi.sh 1
Sunday
masfiyan@masfiyan-HP-EliteBook-840-G2:~/OS_Practice$ sh asfi.sh 11
Sorry!, Only 1 to 7.
```

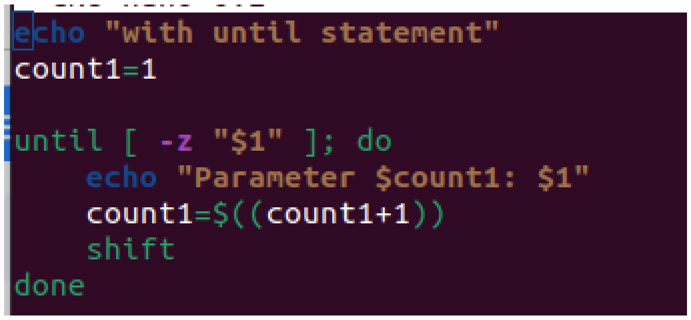
Question 4:

While Statement



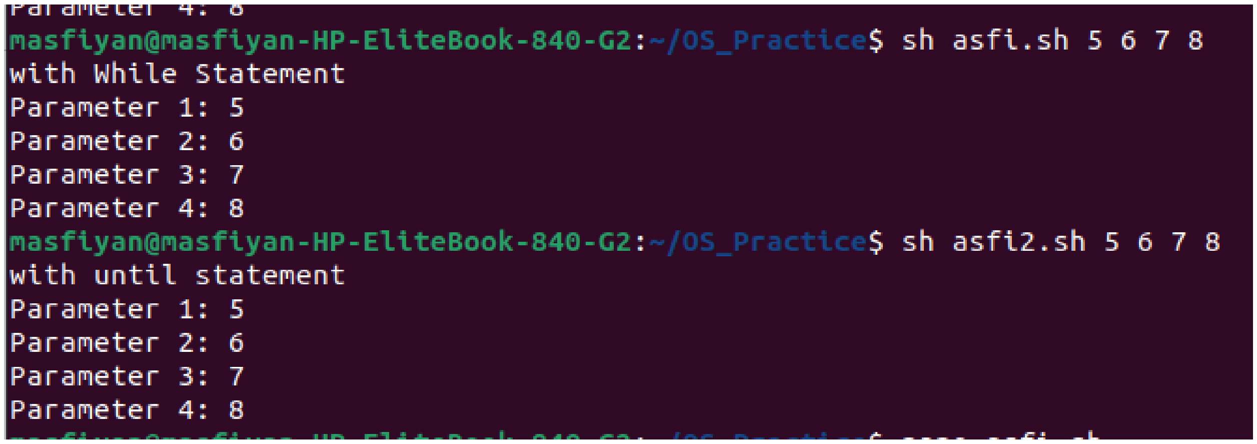
```
count=1
echo "with While Statement"
while [ $# -gt 0 ]; do
    echo "Parameter $count: $1"
    count=$((count+1))
    shift
done
```

Until Statement:



```
echo "with until statement"
count1=1

until [ -z "$1" ]; do
    echo "Parameter $count1: $1"
    count1=$((count1+1))
    shift
done
```

Output:



```
Parameter 4: 8
masfiyan@masfiyan-HP-EliteBook-840-G2:~/OS_Practice$ sh asfi.sh 5 6 7 8
with While Statement
Parameter 1: 5
Parameter 2: 6
Parameter 3: 7
Parameter 4: 8
masfiyan@masfiyan-HP-EliteBook-840-G2:~/OS_Practice$ sh asfi2.sh 5 6 7 8
with until statement
Parameter 1: 5
Parameter 2: 6
Parameter 3: 7
Parameter 4: 8
```

USMAN INSTITUTE OF TECHNOLOGY

Affiliated with NED University of Engineering & Technology, Karachi

Department of Computer Science Fall-2023

CS312 – Operating System Lab #6

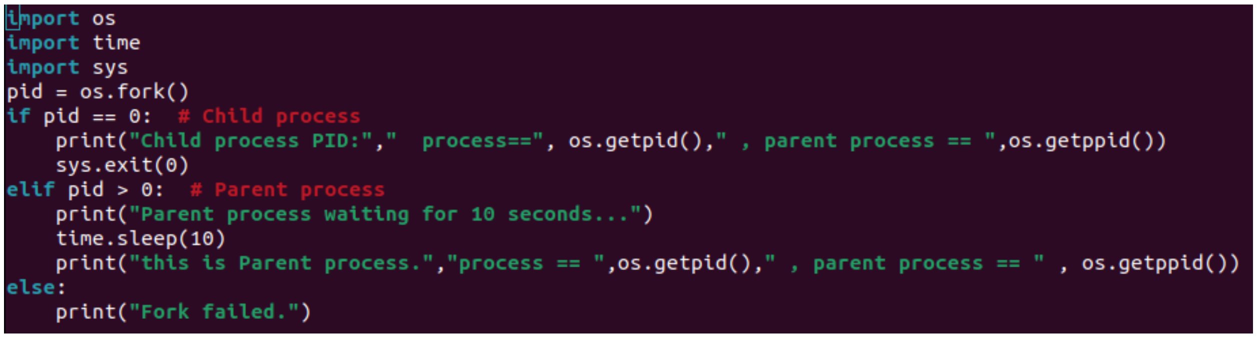
Objective:

Executing some of the most frequently used Linux commands

To understand and use basic utilities/Commands of UNIX/LINUX.

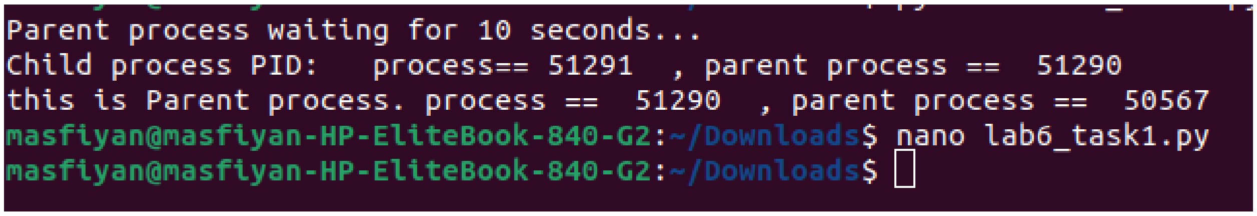
Name of Student	**Muhammad Asfiyan**
Student ID	**21B-111-SE**
Marks Obtained	
Remarks	
Signature	

Question 1:



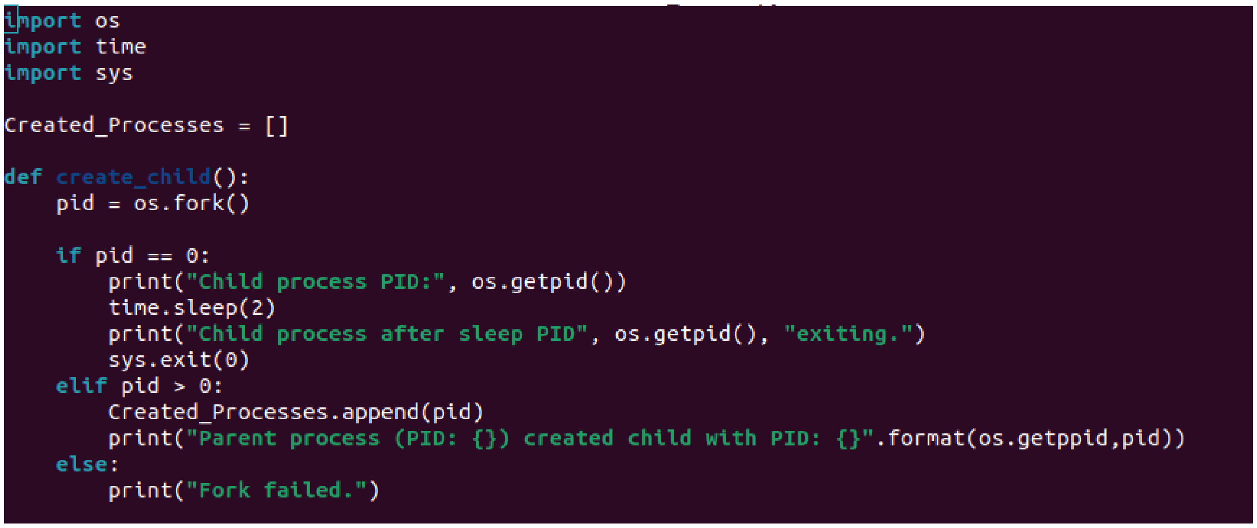
```
import os
import time
import sys
pid = os.fork()
if pid == 0:  # Child process
    print("Child process PID:","  process==", os.getpid()," , parent process == ",os.getppid())
    sys.exit(0)
elif pid > 0:  # Parent process
    print("Parent process waiting for 10 seconds...")
    time.sleep(10)
    print("this is Parent process.","process == ",os.getpid()," , parent process == " , os.getppid())
else:
    print("Fork failed.")
```

OUTPUT:



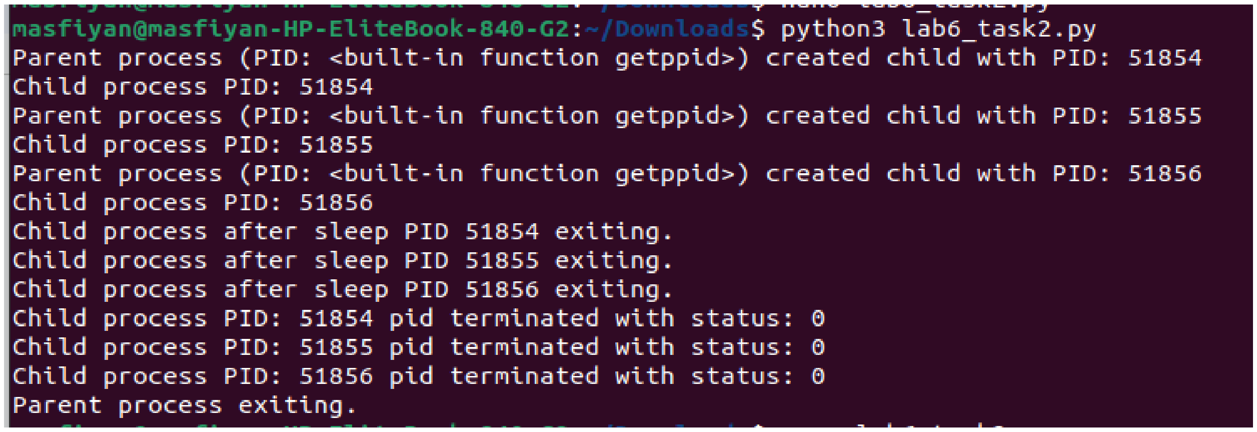
```
Parent process waiting for 10 seconds...
Child process PID:   process== 51291  , parent process ==  51290
this is Parent process. process ==  51290  , parent process ==  50567
masfiyan@masfiyan-HP-EliteBook-840-G2:~/Downloads$ nano lab6_task1.py
masfiyan@masfiyan-HP-EliteBook-840-G2:~/Downloads$ 
```

QUESTION 2:



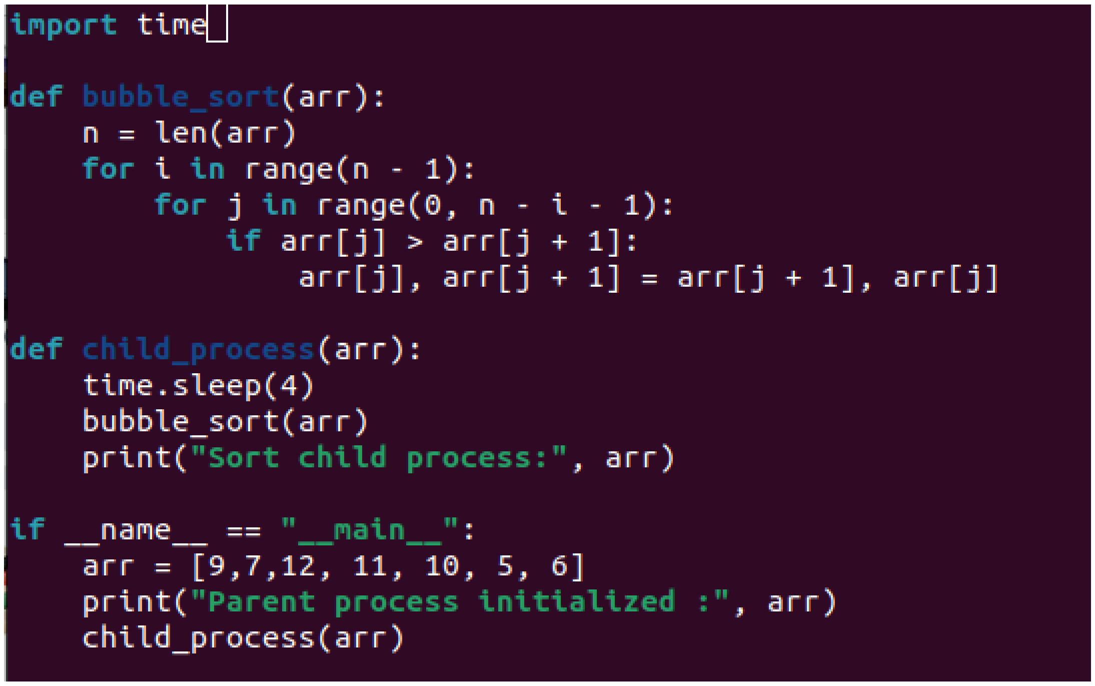
```
import os
import time
import sys

Created_Processes = []

def create_child():
    pid = os.fork()

    if pid == 0:
        print("Child process PID:", os.getpid())
        time.sleep(2)
        print("Child process after sleep PID", os.getpid(), "exiting.")
        sys.exit(0)
    elif pid > 0:
        Created_Processes.append(pid)
        print("Parent process (PID: {}) created child with PID: {}".format(os.getppid,pid))
    else:
        print("Fork failed.")
```

OUTPUT:



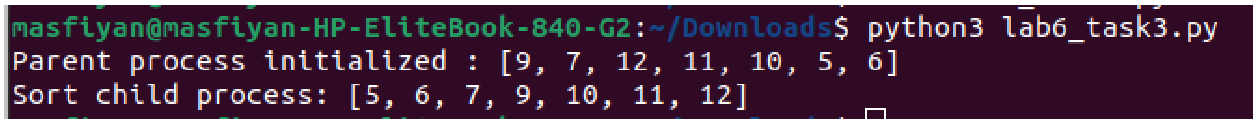
```
masfiyan@masfiyan-HP-EliteBook-840-G2:~/Downloads$ python3 lab6_task2.py
Parent process (PID: <built-in function getppid>) created child with PID: 51854
Child process PID: 51854
Parent process (PID: <built-in function getppid>) created child with PID: 51855
Child process PID: 51855
Parent process (PID: <built-in function getppid>) created child with PID: 51856
Child process PID: 51856
Child process after sleep PID 51854 exiting.
Child process after sleep PID 51855 exiting.
Child process after sleep PID 51856 exiting.
Child process PID: 51854 pid terminated with status: 0
Child process PID: 51855 pid terminated with status: 0
Child process PID: 51856 pid terminated with status: 0
Parent process exiting.
```

QUESTION 3:



```python
import time

def bubble_sort(arr):
    n = len(arr)
    for i in range(n - 1):
        for j in range(0, n - i - 1):
            if arr[j] > arr[j + 1]:
                arr[j], arr[j + 1] = arr[j + 1], arr[j]

def child_process(arr):
    time.sleep(4)
    bubble_sort(arr)
    print("Sort child process:", arr)

if __name__ == "__main__":
    arr = [9,7,12, 11, 10, 5, 6]
    print("Parent process initialized :", arr)
    child_process(arr)
```

OUTPUT:



```
masfiyan@masfiyan-HP-EliteBook-840-G2:~/Downloads$ python3 lab6_task3.py
Parent process initialized : [9, 7, 12, 11, 10, 5, 6]
Sort child process: [5, 6, 7, 9, 10, 11, 12]
```

USMAN INSTITUTE OF TECHNOLOGY

Affiliated with NED University of Engineering & Technology, Karachi

Department of Computer Science Fall-2023

CS312 – Operating System Lab #7

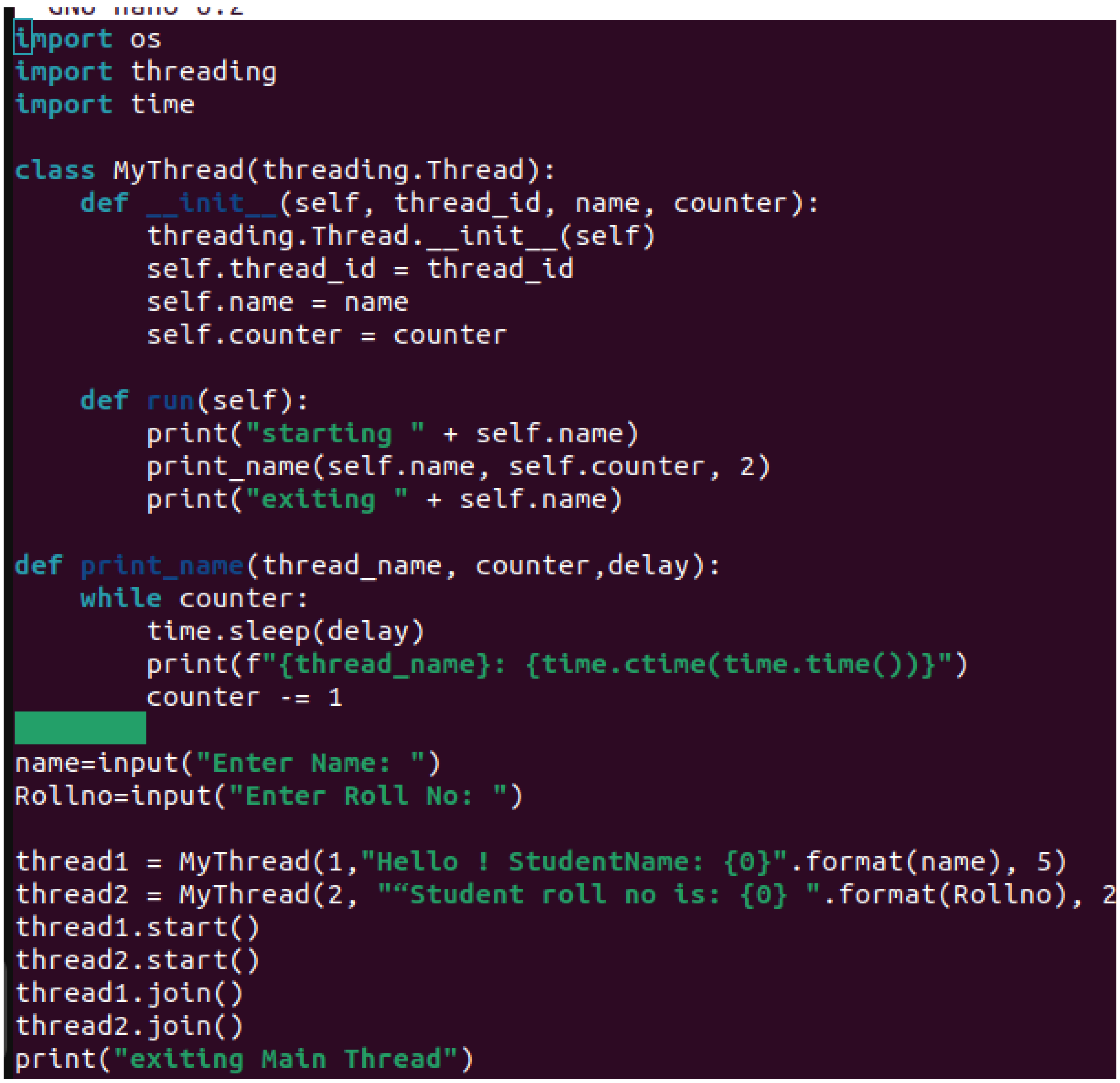
Objective:

Executing some of the most frequently used Linux commands

To understand and use basic utilities/Commands of UNIX/LINUX.

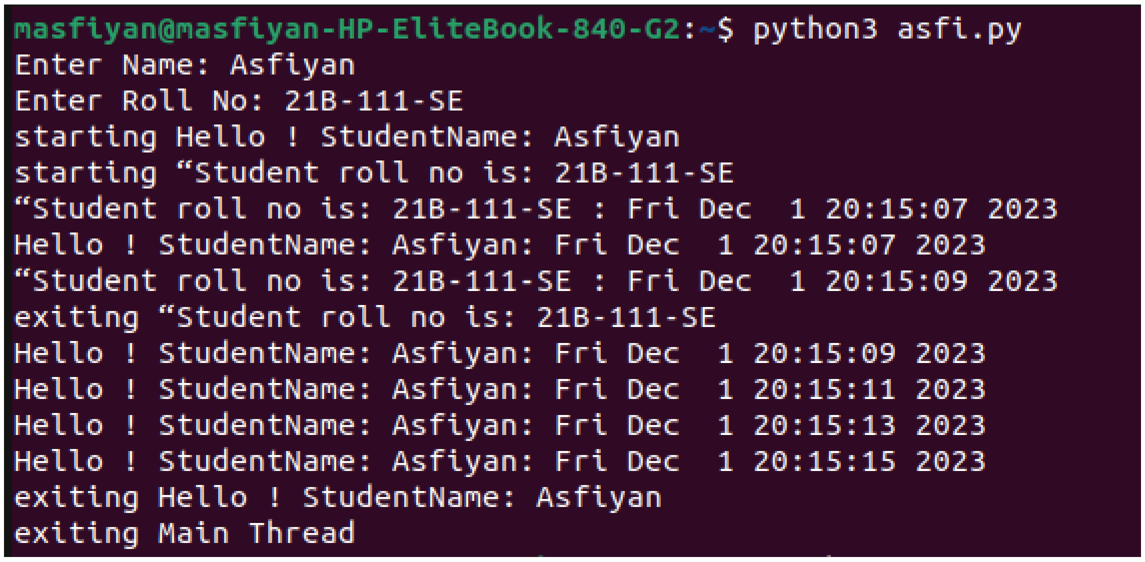
Name of Student	**Muhammad Asfiyan**
Student ID	**21B-111-SE**
Marks Obtained	
Remarks	
Signature	

Question 1:



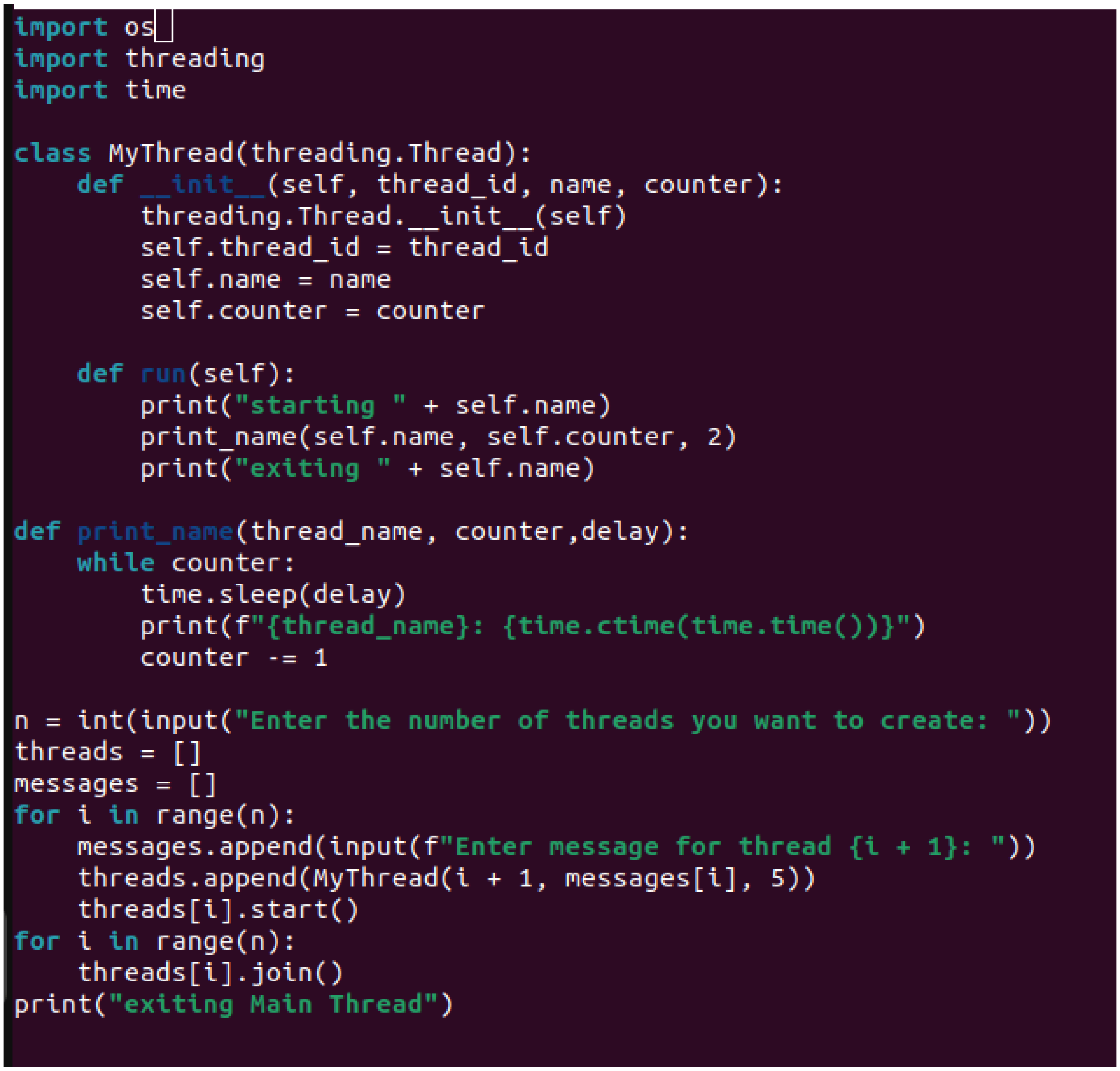
```
import os
import threading
import time

class MyThread(threading.Thread):
    def __init__(self, thread_id, name, counter):
        threading.Thread.__init__(self)
        self.thread_id = thread_id
        self.name = name
        self.counter = counter

    def run(self):
        print("starting " + self.name)
        print_name(self.name, self.counter, 2)
        print("exiting " + self.name)

def print_name(thread_name, counter,delay):
    while counter:
        time.sleep(delay)
        print(f"{thread_name}: {time.ctime(time.time())}")
        counter -= 1

name=input("Enter Name: ")
Rollno=input("Enter Roll No: ")

thread1 = MyThread(1,"Hello ! StudentName: {0}".format(name), 5)
thread2 = MyThread(2, "“Student roll no is: {0} ".format(Rollno), 2
thread1.start()
thread2.start()
thread1.join()
thread2.join()
print("exiting Main Thread")
```

Output:



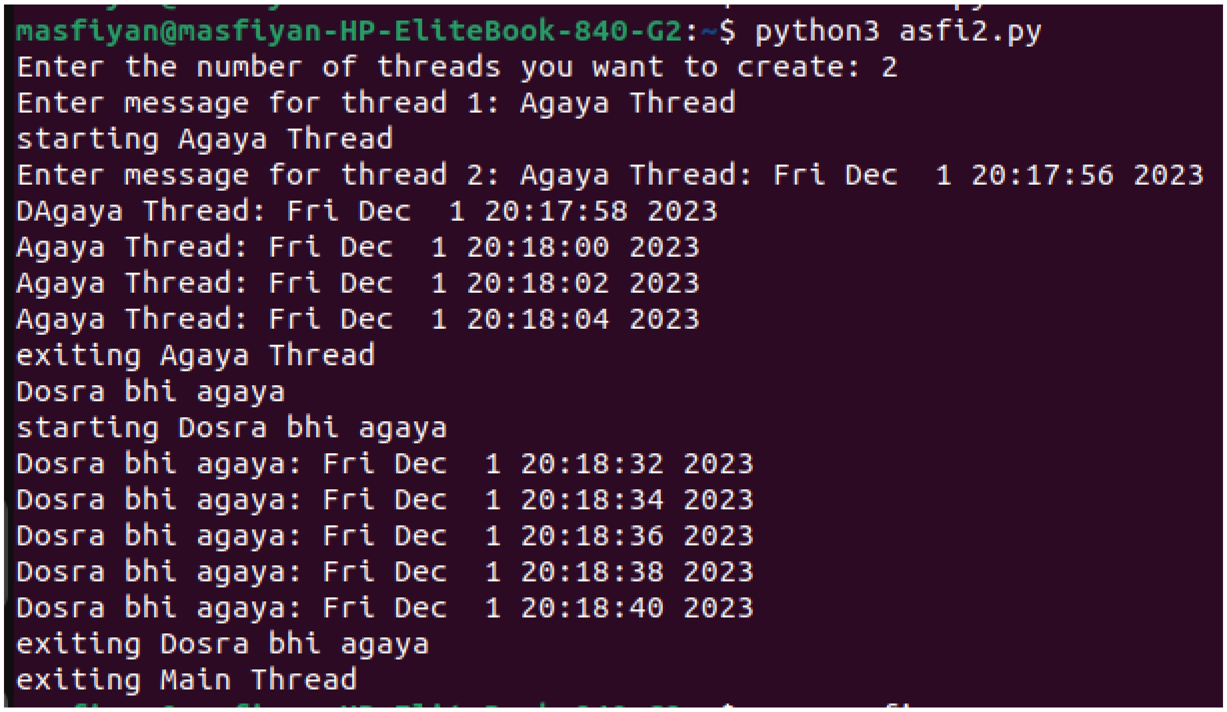
```
masfiyan@masfiyan-HP-EliteBook-840-G2:~$ python3 asfi.py
Enter Name: Asfiyan
Enter Roll No: 21B-111-SE
starting Hello ! StudentName: Asfiyan
starting “Student roll no is: 21B-111-SE
“Student roll no is: 21B-111-SE : Fri Dec  1 20:15:07 2023
Hello ! StudentName: Asfiyan: Fri Dec  1 20:15:07 2023
“Student roll no is: 21B-111-SE : Fri Dec  1 20:15:09 2023
exiting “Student roll no is: 21B-111-SE
Hello ! StudentName: Asfiyan: Fri Dec  1 20:15:09 2023
Hello ! StudentName: Asfiyan: Fri Dec  1 20:15:11 2023
Hello ! StudentName: Asfiyan: Fri Dec  1 20:15:13 2023
Hello ! StudentName: Asfiyan: Fri Dec  1 20:15:15 2023
exiting Hello ! StudentName: Asfiyan
exiting Main Thread
```

Question 2:



```
import os
import threading
import time

class MyThread(threading.Thread):
    def __init__(self, thread_id, name, counter):
        threading.Thread.__init__(self)
        self.thread_id = thread_id
        self.name = name
        self.counter = counter

    def run(self):
        print("starting " + self.name)
        print_name(self.name, self.counter, 2)
        print("exiting " + self.name)

def print_name(thread_name, counter,delay):
    while counter:
        time.sleep(delay)
        print(f"{thread_name}: {time.ctime(time.time())}")
        counter -= 1

n = int(input("Enter the number of threads you want to create: "))
threads = []
messages = []
for i in range(n):
    messages.append(input(f"Enter message for thread {i + 1}: "))
    threads.append(MyThread(i + 1, messages[i], 5))
    threads[i].start()
for i in range(n):
    threads[i].join()
print("exiting Main Thread")
```

Output:


```
masfiyan@masfiyan-HP-EliteBook-840-G2:~$ python3 asfi2.py
Enter the number of threads you want to create: 2
Enter message for thread 1: Agaya Thread
starting Agaya Thread
Enter message for thread 2: Agaya Thread: Fri Dec  1 20:17:56 2023
DAgaya Thread: Fri Dec  1 20:17:58 2023
Agaya Thread: Fri Dec  1 20:18:00 2023
Agaya Thread: Fri Dec  1 20:18:02 2023
Agaya Thread: Fri Dec  1 20:18:04 2023
exiting Agaya Thread
Dosra bhi agaya
starting Dosra bhi agaya
Dosra bhi agaya: Fri Dec  1 20:18:32 2023
Dosra bhi agaya: Fri Dec  1 20:18:34 2023
Dosra bhi agaya: Fri Dec  1 20:18:36 2023
Dosra bhi agaya: Fri Dec  1 20:18:38 2023
Dosra bhi agaya: Fri Dec  1 20:18:40 2023
exiting Dosra bhi agaya
exiting Main Thread
```

USMAN INSTITUTE OF TECHNOLOGY

Affiliated with NED University of Engineering & Technology, Karachi

Department of Computer Science Fall-2023

CS312 – Operating System Lab #8

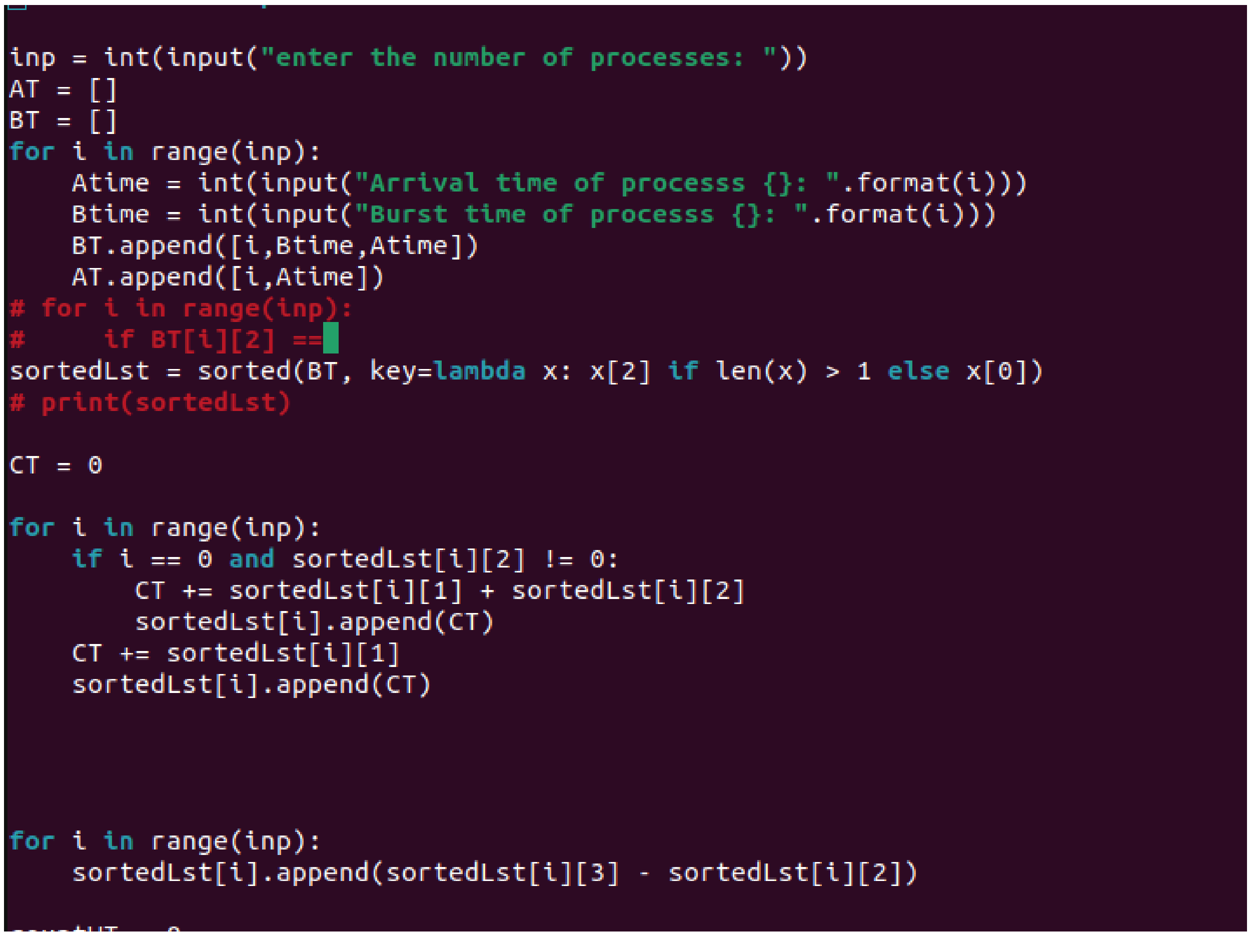
Objective:

Executing some of the most frequently used Linux commands

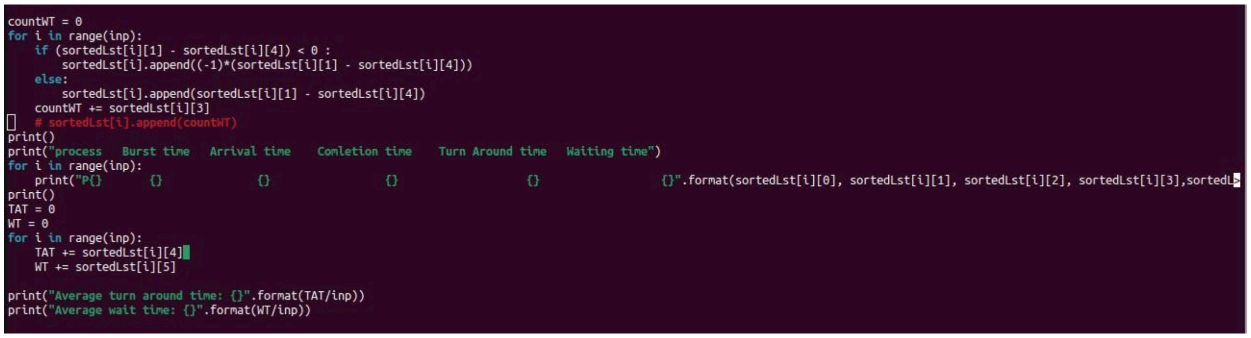
To understand and use basic utilities/Commands of UNIX/LINUX.

Name of Student	**Muhammad Asfiyan**
Student ID	**21B-111-SE**
Marks Obtained	
Remarks	
Signature	

FCFS 👍

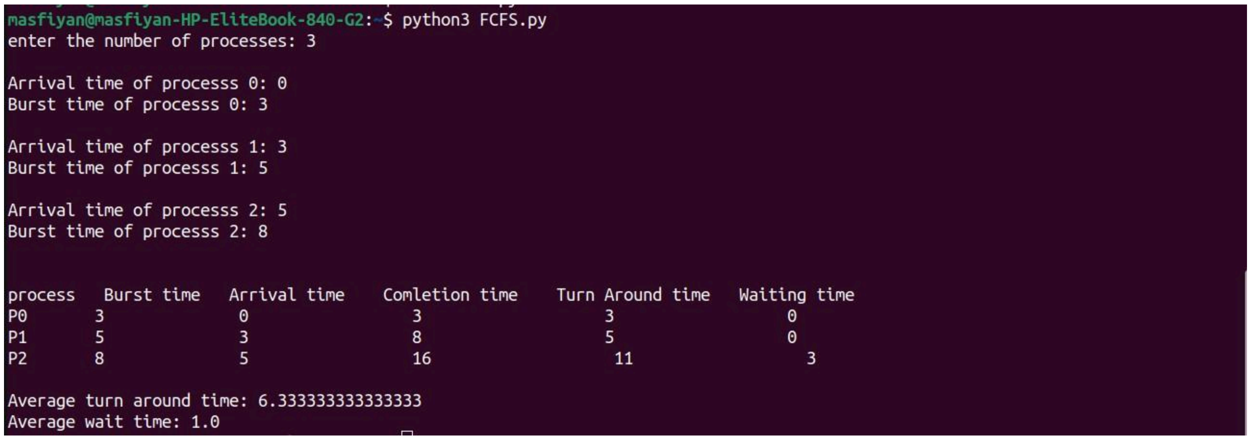


```
inp = int(input("enter the number of processes: "))
AT = []
BT = []
for i in range(inp):
    Atime = int(input("Arrival time of processs {}: ".format(i)))
    Btime = int(input("Burst time of processs {}: ".format(i)))
    BT.append([i,Btime,Atime])
    AT.append([i,Atime])
# for i in range(inp):
#     if BT[i][2] ==
sortedLst = sorted(BT, key=lambda x: x[2] if len(x) > 1 else x[0])
# print(sortedLst)

CT = 0

for i in range(inp):
    if i == 0 and sortedLst[i][2] != 0:
        CT += sortedLst[i][1] + sortedLst[i][2]
        sortedLst[i].append(CT)
    CT += sortedLst[i][1]
    sortedLst[i].append(CT)



for i in range(inp):
    sortedLst[i].append(sortedLst[i][3] - sortedLst[i][2])
```



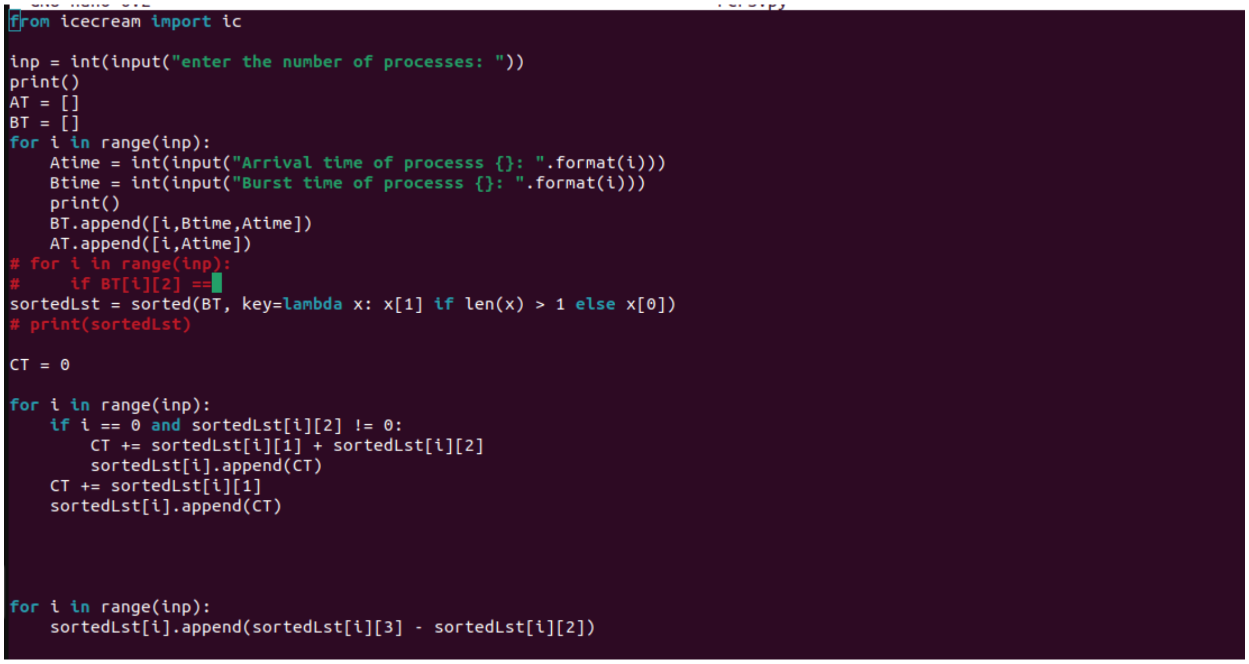
```
countWT = 0
for i in range(inp):
    if (sortedLst[i][1] - sortedLst[i][4]) < 0 :
        sortedLst[i].append((-1)*(sortedLst[i][1] - sortedLst[i][4]))
    else:
        sortedLst[i].append(sortedLst[i][1] - sortedLst[i][4])
    countWT += sortedLst[i][3]
    # sortedLst[i].append(countWT)
print()
print("process   Burst time   Arrival time    Comletion time    Turn Around time   Waiting time")
for i in range(inp):
    print("P{}       {}              {}                {}                 {}                  {}".format(sortedLst[i][0], sortedLst[i][1], sortedLst[i][2], sortedLst[i][3],sortedL
print()
TAT = 0
WT = 0
for i in range(inp):
    TAT += sortedLst[i][4]
    WT += sortedLst[i][5]

print("Average turn around time: {}".format(TAT/inp))
print("Average wait time: {}".format(WT/inp))
```

Output:

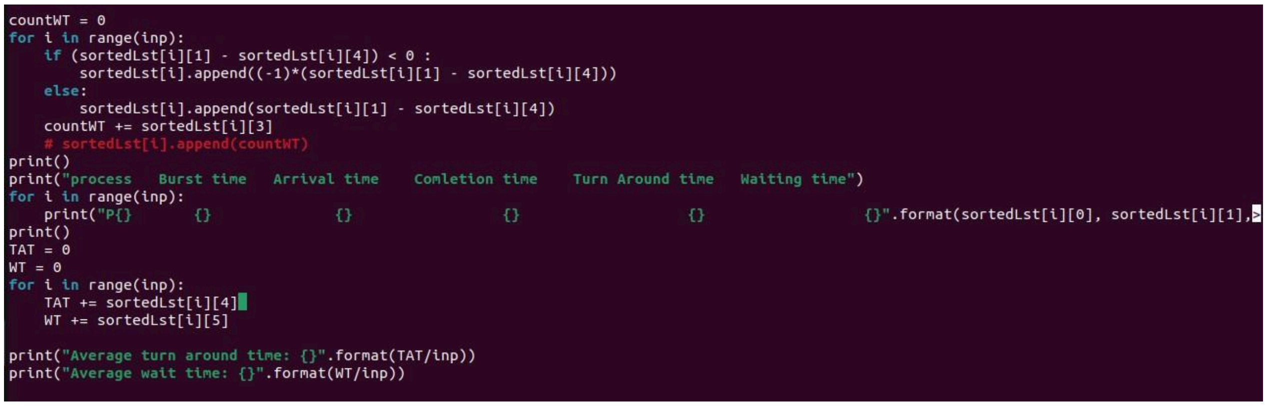


```
masfiyan@masfiyan-HP-EliteBook-840-G2:~$ python3 FCFS.py
enter the number of processes: 3

Arrival time of processs 0: 0
Burst time of processs 0: 3

Arrival time of processs 1: 3
Burst time of processs 1: 5

Arrival time of processs 2: 5
Burst time of processs 2: 8


process   Burst time   Arrival time    Comletion time    Turn Around time   Waiting time
P0       3              0                3                  3                  0
P1       5              3                8                  5                  0
P2       8              5                16                  11                  3

Average turn around time: 6.333333333333333
Average wait time: 1.0
```

SJF

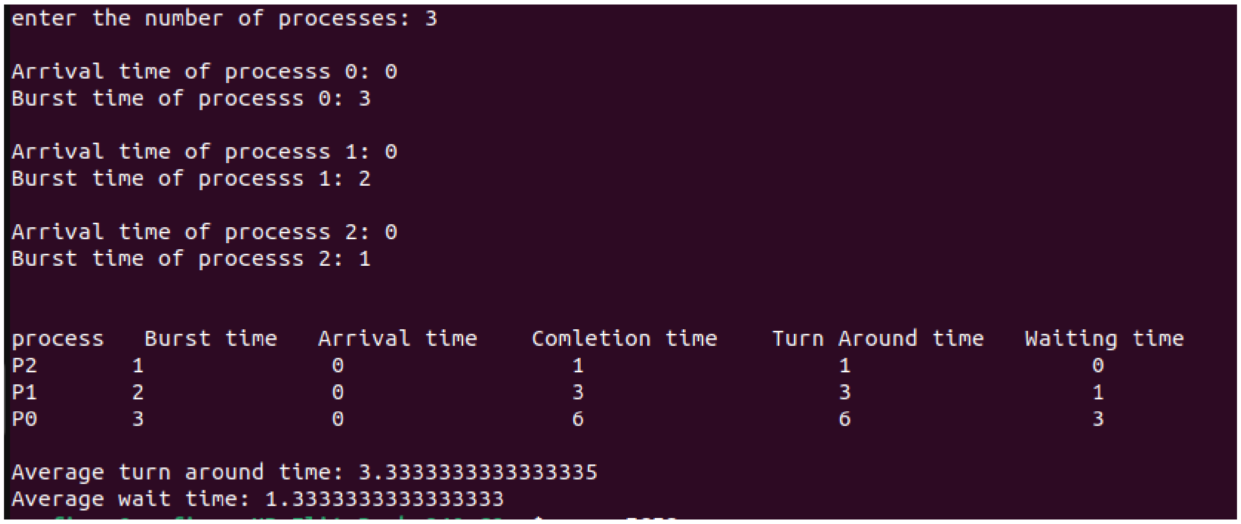


```
from icecream import ic

inp = int(input("enter the number of processes: "))
print()
AT = []
BT = []
for i in range(inp):
    Atime = int(input("Arrival time of processs {}: ".format(i)))
    Btime = int(input("Burst time of processs {}: ".format(i)))
    print()
    BT.append([i,Btime,Atime])
    AT.append([i,Atime])
# for i in range(inp):
#     if BT[i][2] ==
sortedLst = sorted(BT, key=lambda x: x[1] if len(x) > 1 else x[0])
# print(sortedLst)

CT = 0

for i in range(inp):
    if i == 0 and sortedLst[i][2] != 0:
        CT += sortedLst[i][1] + sortedLst[i][2]
        sortedLst[i].append(CT)
    CT += sortedLst[i][1]
    sortedLst[i].append(CT)



for i in range(inp):
    sortedLst[i].append(sortedLst[i][3] - sortedLst[i][2])
```



```
countWT = 0
for i in range(inp):
    if (sortedLst[i][1] - sortedLst[i][4]) < 0 :
        sortedLst[i].append((-1)*(sortedLst[i][1] - sortedLst[i][4]))
    else:
        sortedLst[i].append(sortedLst[i][1] - sortedLst[i][4])
    countWT += sortedLst[i][3]
    # sortedLst[i].append(countWT)
print()
print("process   Burst time   Arrival time    Comletion time    Turn Around time   Waiting time")
for i in range(inp):
    print("P{}       {}              {}                 {}                  {}                  {}".format(sortedLst[i][0], sortedLst[i][1],>
print()
TAT = 0
WT = 0
for i in range(inp):
    TAT += sortedLst[i][4]
    WT += sortedLst[i][5]

print("Average turn around time: {}".format(TAT/inp))
print("Average wait time: {}".format(WT/inp))
```

OUTPUT:



```
enter the number of processes: 3

Arrival time of processs 0: 0
Burst time of processs 0: 3

Arrival time of processs 1: 0
Burst time of processs 1: 2

Arrival time of processs 2: 0
Burst time of processs 2: 1


process   Burst time   Arrival time    Comletion time    Turn Around time   Waiting time
P2       1              0                1                   1                   0
P1       2              0                3                   3                   1
P0       3              0                6                   6                   3

Average turn around time: 3.3333333333333335
Average wait time: 1.3333333333333333
```

USMAN INSTITUTE OF TECHNOLOGY

Affiliated with NED University of Engineering & Technology, Karachi

Department of Computer Science Fall-2023

CS312 – Operating System Lab #9

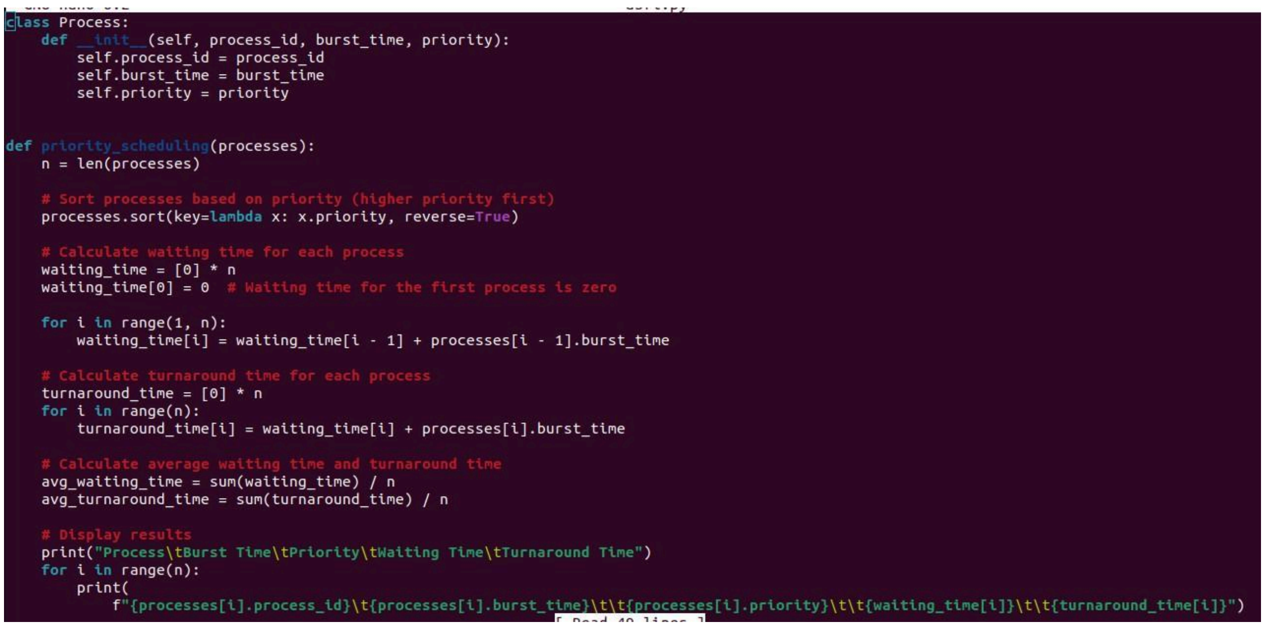
Objective:

Executing some of the most frequently used Linux commands

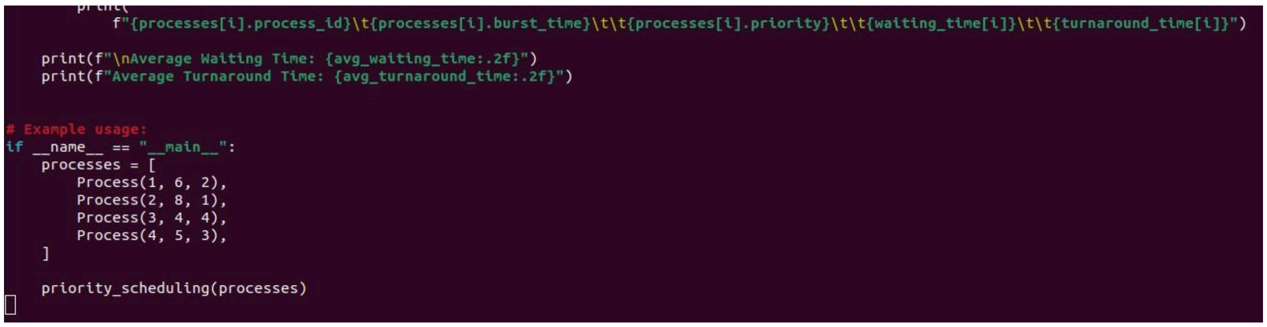
To understand and use basic utilities/Commands of UNIX/LINUX.

Name of Student	**Muhammad Asfiyan**
Student ID	**21B-111-SE**
Marks Obtained	
Remarks	
Signature	

QUES 1:

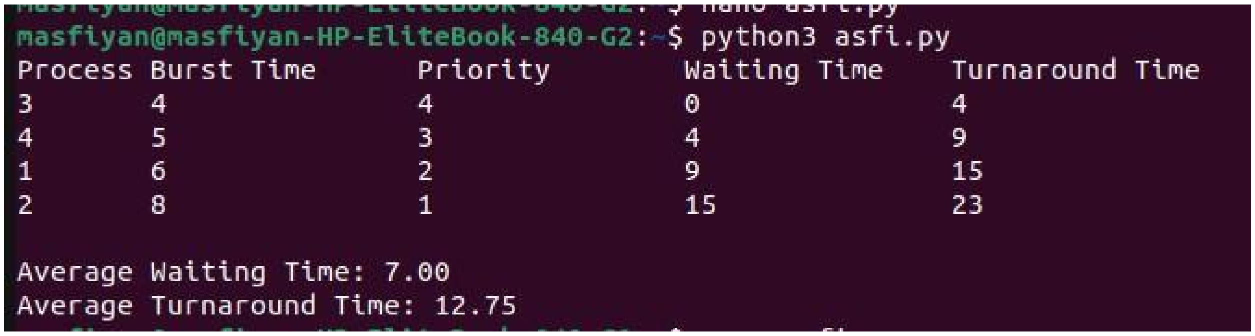


```python
class Process:
    def __init__(self, process_id, burst_time, priority):
        self.process_id = process_id
        self.burst_time = burst_time
        self.priority = priority


def priority_scheduling(processes):
    n = len(processes)

    # Sort processes based on priority (higher priority first)
    processes.sort(key=lambda x: x.priority, reverse=True)

    # Calculate waiting time for each process
    waiting_time = [0] * n
    waiting_time[0] = 0  # Waiting time for the first process is zero

    for i in range(1, n):
        waiting_time[i] = waiting_time[i - 1] + processes[i - 1].burst_time

    # Calculate turnaround time for each process
    turnaround_time = [0] * n
    for i in range(n):
        turnaround_time[i] = waiting_time[i] + processes[i].burst_time

    # Calculate average waiting time and turnaround time
    avg_waiting_time = sum(waiting_time) / n
    avg_turnaround_time = sum(turnaround_time) / n

    # Display results
    print("Process\tBurst Time\tPriority\tWaiting Time\tTurnaround Time")
    for i in range(n):
        print(
            f"{processes[i].process_id}\t{processes[i].burst_time}\t\t{processes[i].priority}\t\t{waiting_time[i]}\t\t{turnaround_time[i]}")
```



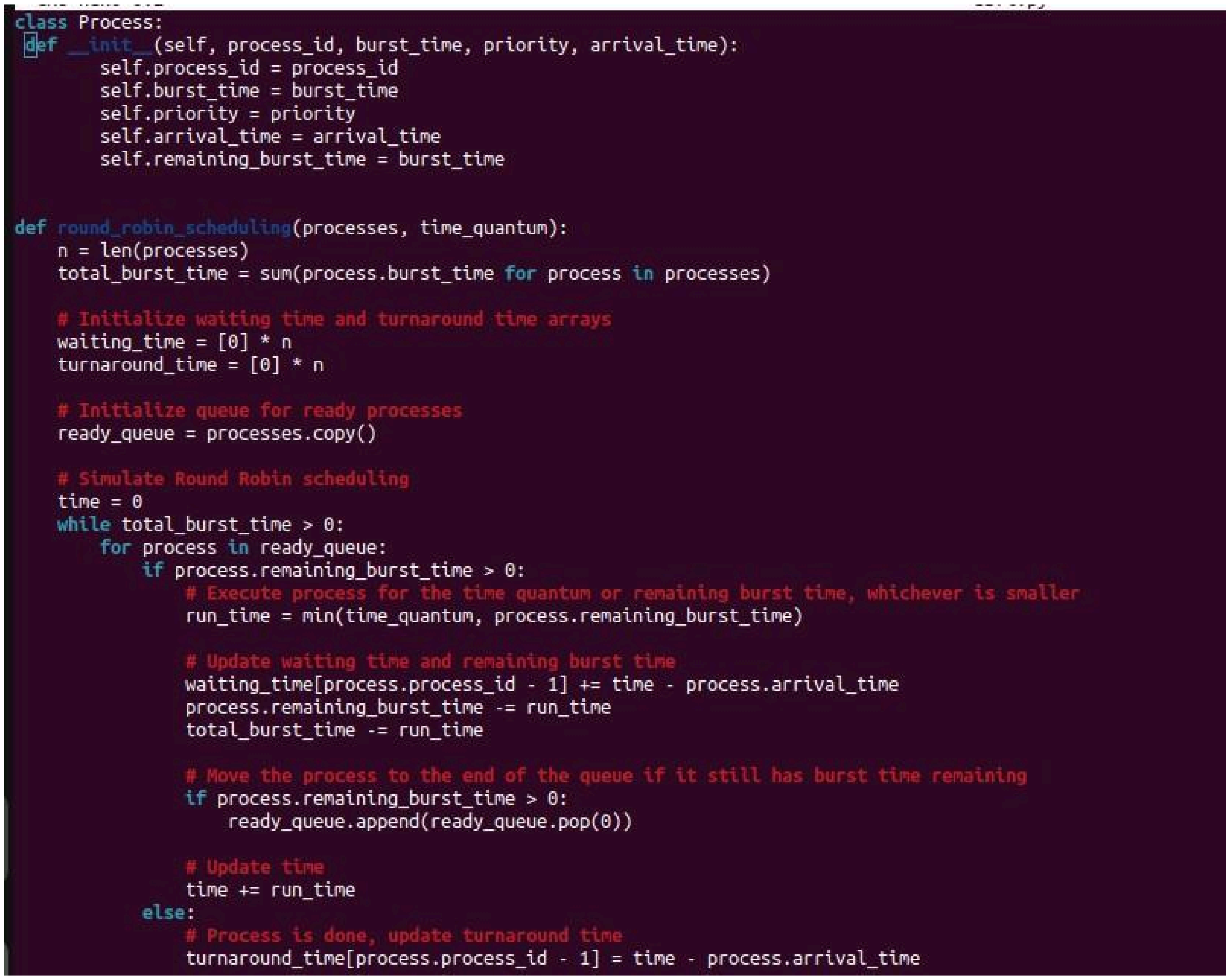
```python
            f"{processes[i].process_id}\t{processes[i].burst_time}\t\t{processes[i].priority}\t\t{waiting_time[i]}\t\t{turnaround_time[i]}")

    print(f"\nAverage Waiting Time: {avg_waiting_time:.2f}")
    print(f"Average Turnaround Time: {avg_turnaround_time:.2f}")


# Example usage:
if __name__ == "__main__":
    processes = [
        Process(1, 6, 2),
        Process(2, 8, 1),
        Process(3, 4, 4),
        Process(4, 5, 3),
    ]

    priority_scheduling(processes)
```

OUTPUT:

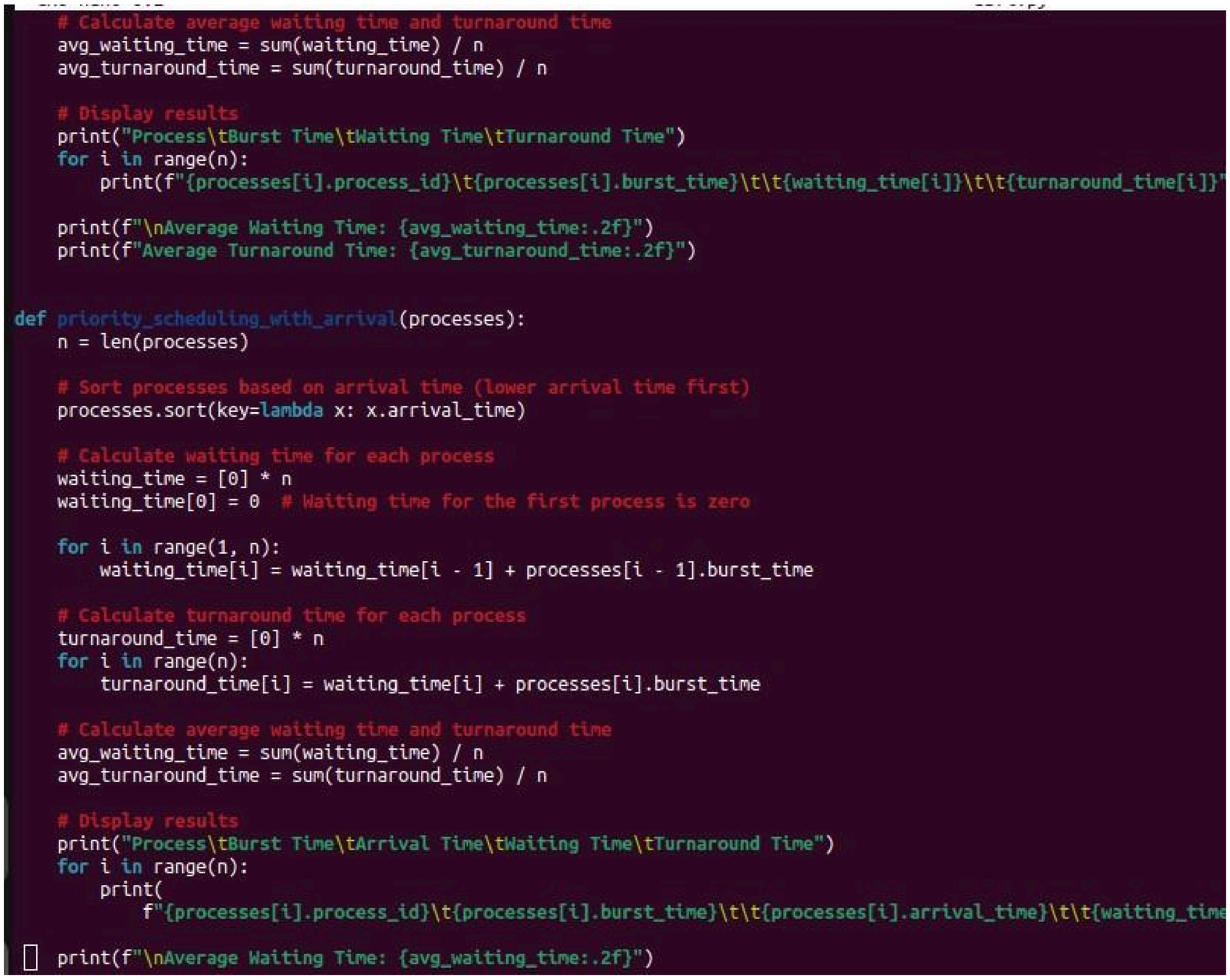


```
masfiyan@masfiyan-HP-EliteBook-840-G2:~$ python3 asfi.py
Process Burst Time      Priority        Waiting Time    Turnaround Time
3       4               4               0               4
4       5               3               4               9
1       6               2               9               15
2       8               1               15              23

Average Waiting Time: 7.00
Average Turnaround Time: 12.75
```

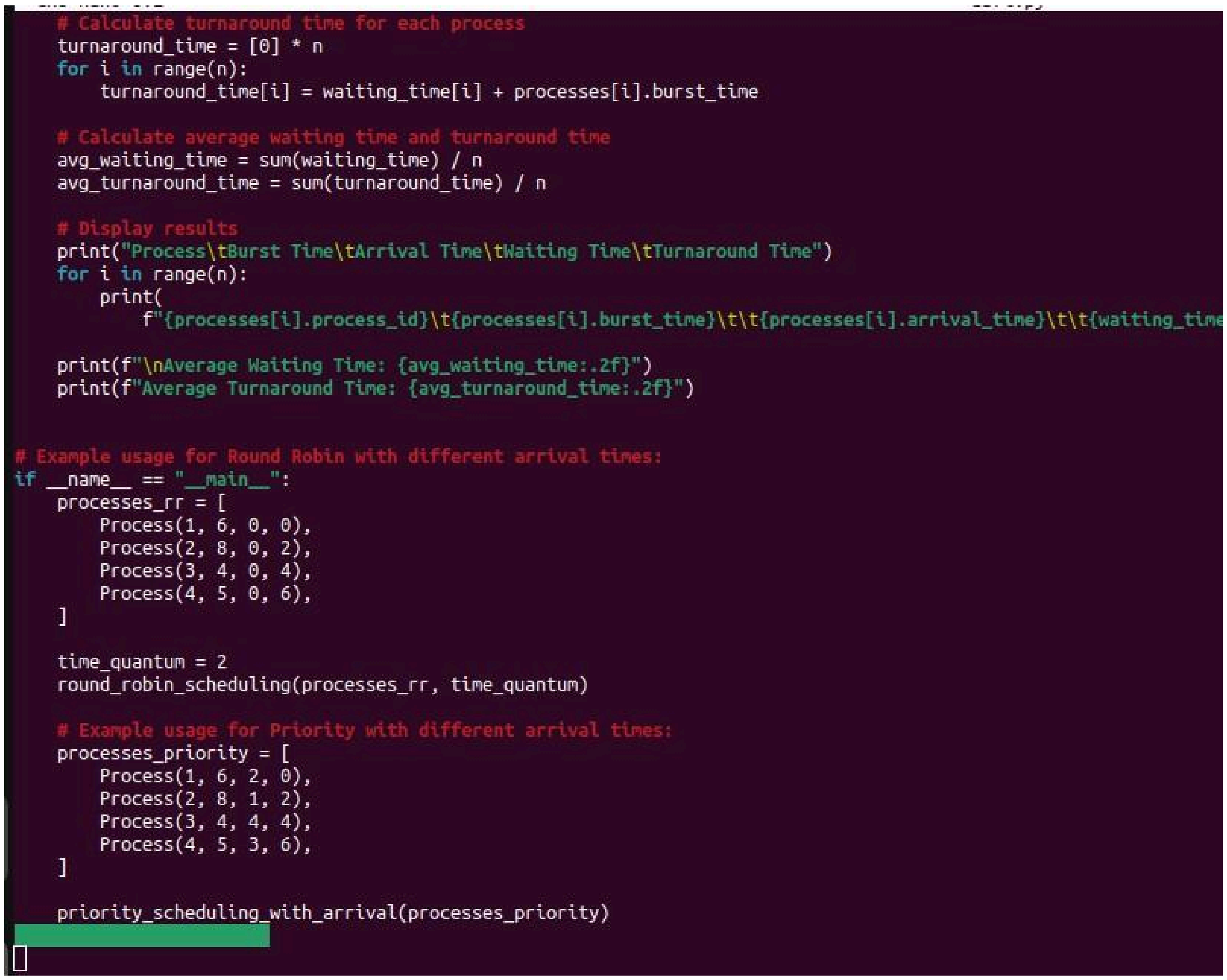
QUES 2:



```
class Process:
 def __init__(self, process_id, burst_time, priority, arrival_time):
        self.process_id = process_id
        self.burst_time = burst_time
        self.priority = priority
        self.arrival_time = arrival_time
        self.remaining_burst_time = burst_time


def round_robin_scheduling(processes, time_quantum):
    n = len(processes)
    total_burst_time = sum(process.burst_time for process in processes)

    # Initialize waiting time and turnaround time arrays
    waiting_time = [0] * n
    turnaround_time = [0] * n

    # Initialize queue for ready processes
    ready_queue = processes.copy()

    # Simulate Round Robin scheduling
    time = 0
    while total_burst_time > 0:
        for process in ready_queue:
            if process.remaining_burst_time > 0:
                # Execute process for the time quantum or remaining burst time, whichever is smaller
                run_time = min(time_quantum, process.remaining_burst_time)

                # Update waiting time and remaining burst time
                waiting_time[process.process_id - 1] += time - process.arrival_time
                process.remaining_burst_time -= run_time
                total_burst_time -= run_time

                # Move the process to the end of the queue if it still has burst time remaining
                if process.remaining_burst_time > 0:
                    ready_queue.append(ready_queue.pop(0))

                # Update time
                time += run_time
            else:
                # Process is done, update turnaround time
                turnaround_time[process.process_id - 1] = time - process.arrival_time
```

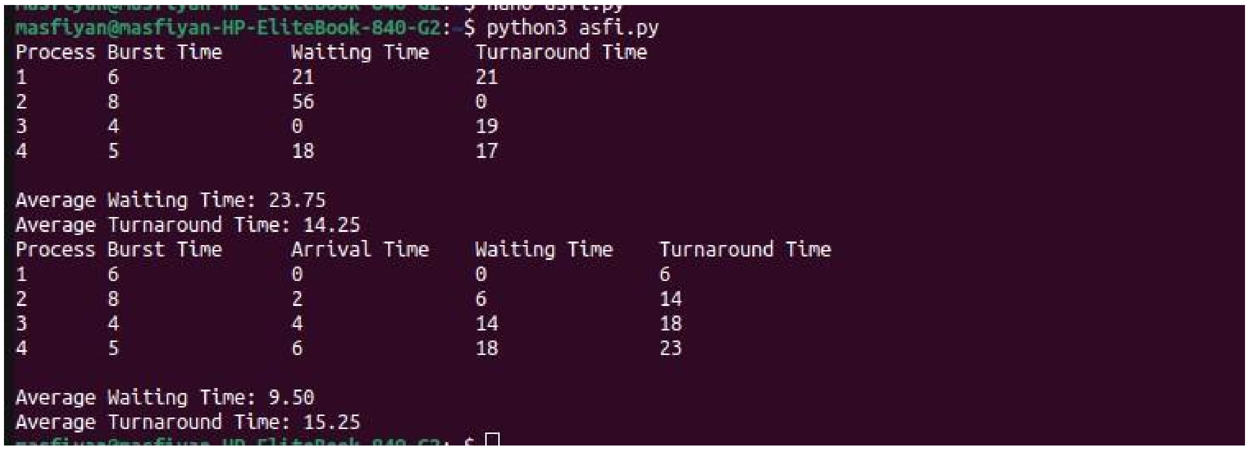


```
    # Calculate average waiting time and turnaround time
    avg_waiting_time = sum(waiting_time) / n
    avg_turnaround_time = sum(turnaround_time) / n

    # Display results
    print("Process\tBurst Time\tWaiting Time\tTurnaround Time")
    for i in range(n):
        print(f"{processes[i].process_id}\t{processes[i].burst_time}\t\t{waiting_time[i]}\t\t{turnaround_time[i]}"

    print(f"\nAverage Waiting Time: {avg_waiting_time:.2f}")
    print(f"Average Turnaround Time: {avg_turnaround_time:.2f}")


def priority_scheduling_with_arrival(processes):
    n = len(processes)

    # Sort processes based on arrival time (lower arrival time first)
    processes.sort(key=lambda x: x.arrival_time)

    # Calculate waiting time for each process
    waiting_time = [0] * n
    waiting_time[0] = 0  # Waiting time for the first process is zero

    for i in range(1, n):
        waiting_time[i] = waiting_time[i - 1] + processes[i - 1].burst_time

    # Calculate turnaround time for each process
    turnaround_time = [0] * n
    for i in range(n):
        turnaround_time[i] = waiting_time[i] + processes[i].burst_time

    # Calculate average waiting time and turnaround time
    avg_waiting_time = sum(waiting_time) / n
    avg_turnaround_time = sum(turnaround_time) / n

    # Display results
    print("Process\tBurst Time\tArrival Time\tWaiting Time\tTurnaround Time")
    for i in range(n):
        print(
            f"{processes[i].process_id}\t{processes[i].burst_time}\t\t{processes[i].arrival_time}\t\t{waiting_time

    print(f"\nAverage Waiting Time: {avg_waiting_time:.2f}")
```



```
    # Calculate turnaround time for each process
    turnaround_time = [0] * n
    for i in range(n):
        turnaround_time[i] = waiting_time[i] + processes[i].burst_time

    # Calculate average waiting time and turnaround time
    avg_waiting_time = sum(waiting_time) / n
    avg_turnaround_time = sum(turnaround_time) / n

    # Display results
    print("Process\tBurst Time\tArrival Time\tWaiting Time\tTurnaround Time")
    for i in range(n):
        print(
            f"{processes[i].process_id}\t{processes[i].burst_time}\t\t{processes[i].arrival_time}\t\t{waiting_time

    print(f"\nAverage Waiting Time: {avg_waiting_time:.2f}")
    print(f"Average Turnaround Time: {avg_turnaround_time:.2f}")


# Example usage for Round Robin with different arrival times:
if __name__ == "__main__":
    processes_rr = [
        Process(1, 6, 0, 0),
        Process(2, 8, 0, 2),
        Process(3, 4, 0, 4),
        Process(4, 5, 0, 6),
    ]

    time_quantum = 2
    round_robin_scheduling(processes_rr, time_quantum)

    # Example usage for Priority with different arrival times:
    processes_priority = [
        Process(1, 6, 2, 0),
        Process(2, 8, 1, 2),
        Process(3, 4, 4, 4),
        Process(4, 5, 3, 6),
    ]

    priority_scheduling_with_arrival(processes_priority)
```

OUTPUT:



```
masfiyan@masfiyan-HP-EliteBook-840-G2:~$ python3 asfi.py
Process Burst Time      Waiting Time    Turnaround Time
1       6               21              21
2       8               56              0
3       4               0               19
4       5               18              17

Average Waiting Time: 23.75
Average Turnaround Time: 14.25
Process Burst Time      Arrival Time    Waiting Time    Turnaround Time
1       6               0               0               6
2       8               2               6               14
3       4               4               14              18
4       5               6               18              23

Average Waiting Time: 9.50
Average Turnaround Time: 15.25
```

USMAN INSTITUTE OF TECHNOLOGY

Affiliated with NED University of Engineering & Technology, Karachi

Department of Computer Science Fall-2023

CS312 – Operating System Lab #10

Objective:

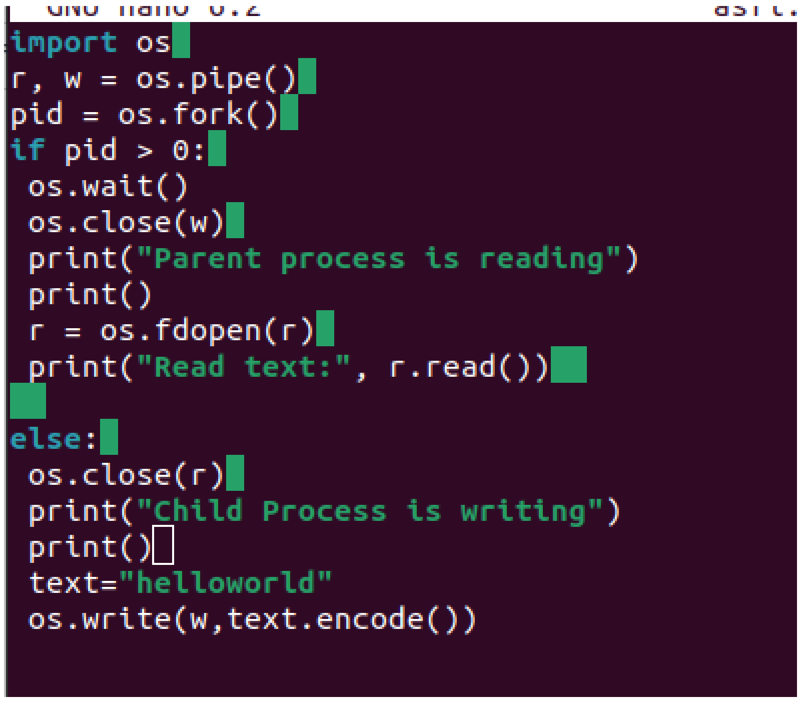
Executing some of the most frequently used Linux commands

To understand and use basic utilities/Commands of UNIX/LINUX.

Name of Student	**Muhammad Asfiyan**
Student ID	**21B-111-SE**
Marks Obtained	
Remarks	

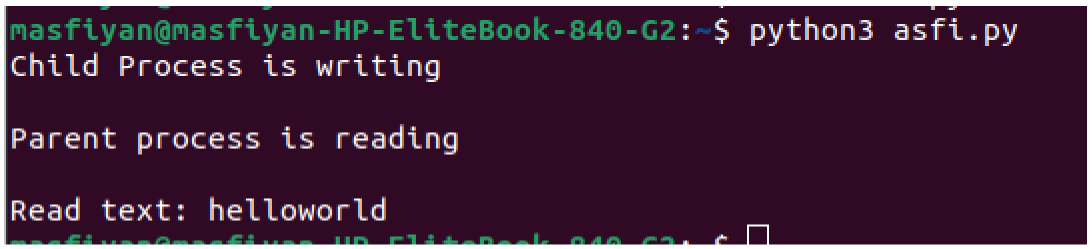
Signature	

Class TasK:



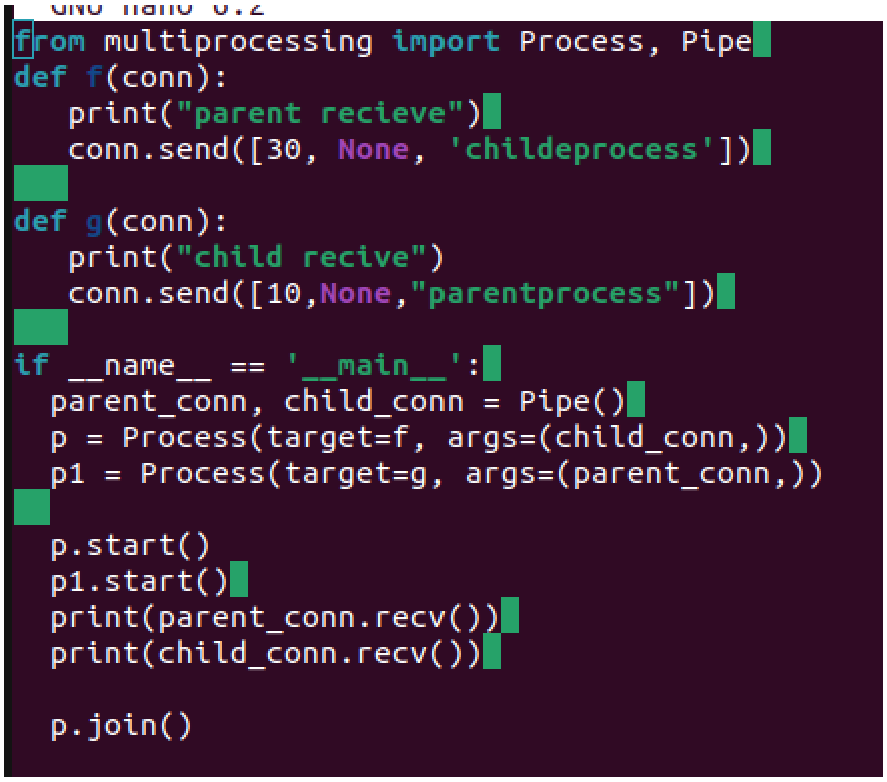
```
import os
r, w = os.pipe()
pid = os.fork()
if pid > 0:
 os.wait()
 os.close(w)
 print("Parent process is reading")
 print()
 r = os.fdopen(r)
 print("Read text:", r.read())

else:
 os.close(r)
 print("Child Process is writing")
 print()
 text="helloworld"
 os.write(w,text.encode())
```

Output:



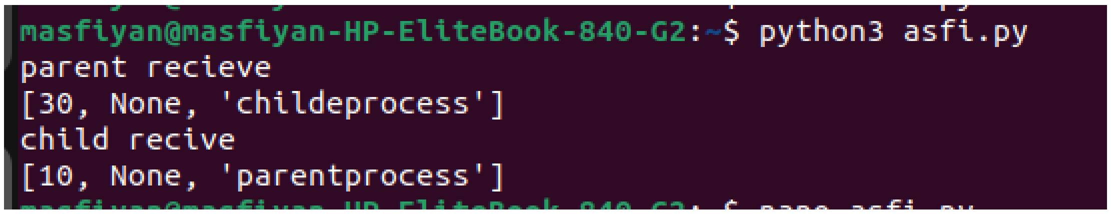
```
masfiyan@masfiyan-HP-EliteBook-840-G2:~$ python3 asfi.py
Child Process is writing

Parent process is reading

Read text: helloworld
```

TASK1:



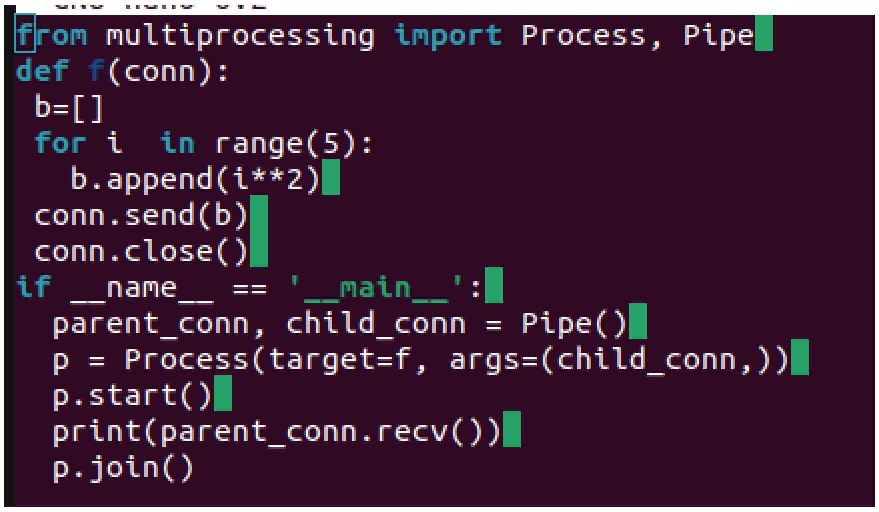
```
from multiprocessing import Process, Pipe
def f(conn):
    print("parent recieve")
    conn.send([30, None, 'childeprocess'])

def g(conn):
    print("child recive")
    conn.send([10,None,"parentprocess"])

if __name__ == '__main__':
  parent_conn, child_conn = Pipe()
  p = Process(target=f, args=(child_conn,))
  p1 = Process(target=g, args=(parent_conn,))

  p.start()
  p1.start()
  print(parent_conn.recv())
  print(child_conn.recv())

  p.join()
```

OUTPUT:



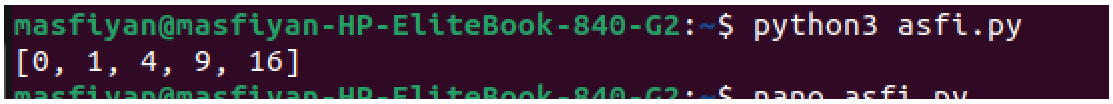
```
masfiyan@masfiyan-HP-EliteBook-840-G2:~$ python3 asfi.py
parent recieve
[30, None, 'childeprocess']
child recive
[10, None, 'parentprocess']
```

TASK2:



```
from multiprocessing import Process, Pipe
def f(conn):
 b=[]
 for i  in range(5):
   b.append(i**2)
 conn.send(b)
 conn.close()
if __name__ == '__main__':
  parent_conn, child_conn = Pipe()
  p = Process(target=f, args=(child_conn,))
  p.start()
  print(parent_conn.recv())
  p.join()
```

OUTPUT:



```
masfiyan@masfiyan-HP-EliteBook-840-G2:~$ python3 asfi.py
[0, 1, 4, 9, 16]
masfiyan@masfiyan-HP-EliteBook-840-G2:~$ nano asfi.py
```

USMAN INSTITUTE OF TECHNOLOGY

Affiliated with NED University of Engineering & Technology, Karachi

Department of Computer Science Fall-2023

CS312 – Operating System Lab #11

Objective:

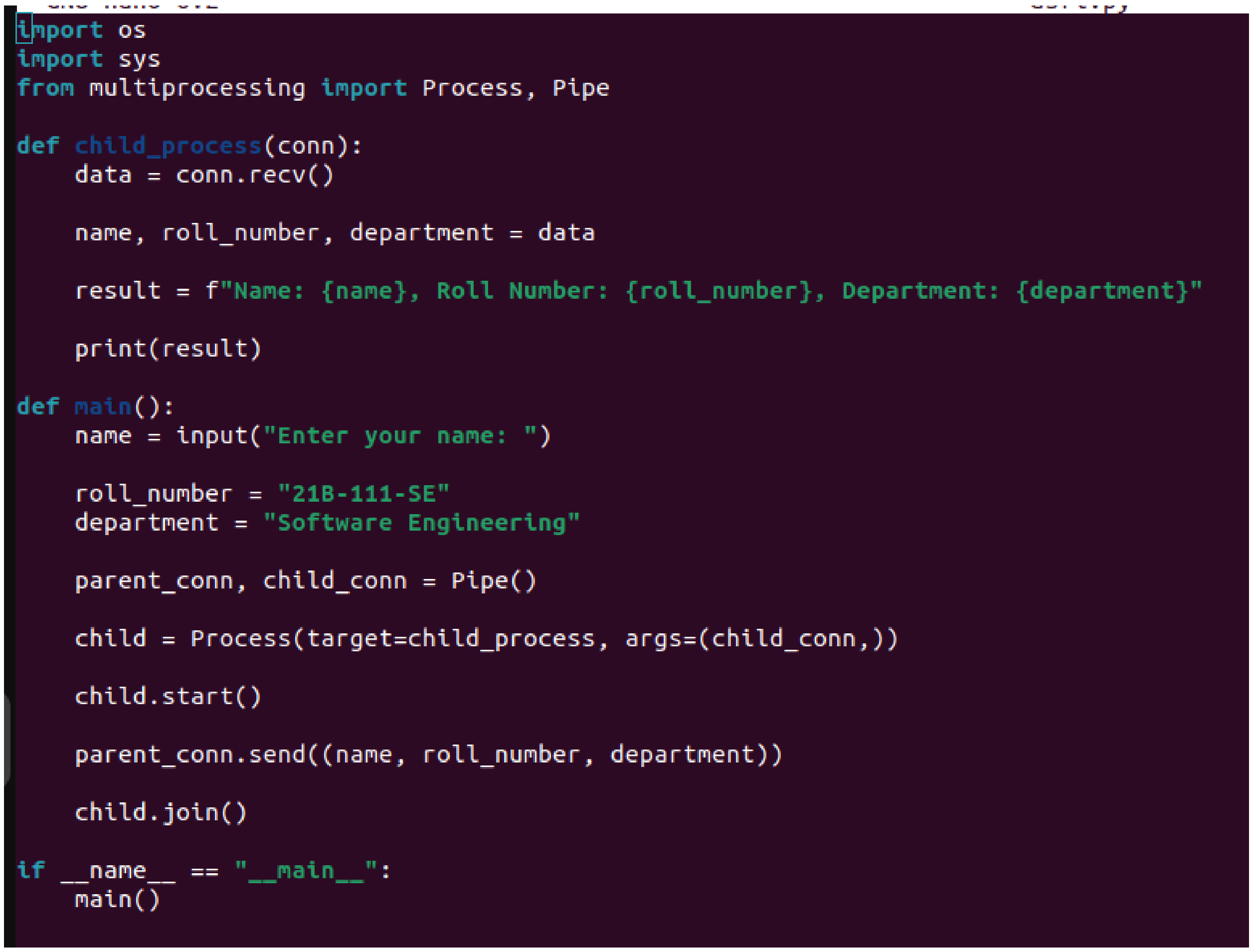
Executing some of the most frequently used Linux commands

To understand and use basic utilities/Commands of UNIX/LINUX.

Name of Student	**Muhammad Asfiyan**
Student ID	**21B-111-SE**
Marks Obtained	
Remarks	

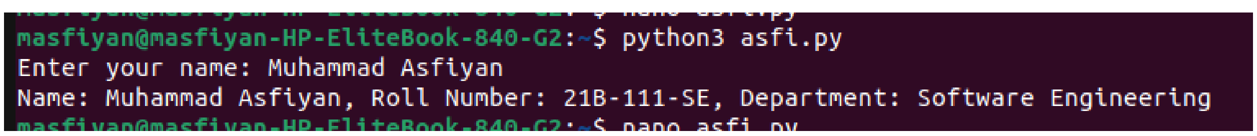
Signature	

QUESTION 1:



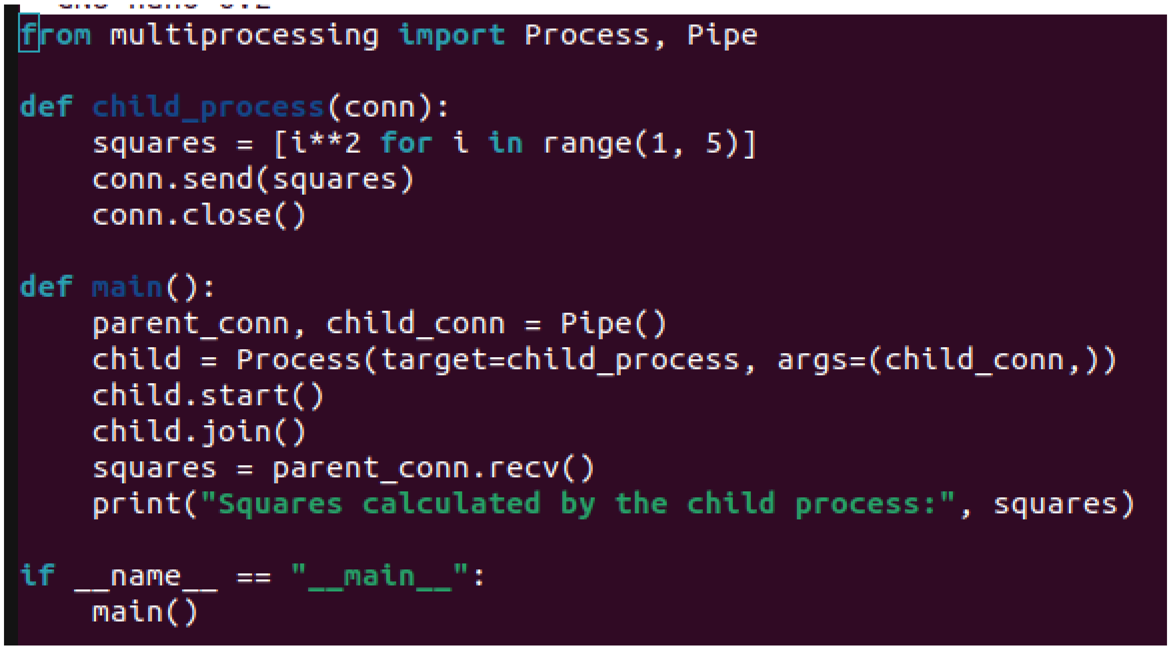
```python
import os
import sys
from multiprocessing import Process, Pipe

def child_process(conn):
    data = conn.recv()

    name, roll_number, department = data

    result = f"Name: {name}, Roll Number: {roll_number}, Department: {department}"

    print(result)

def main():
    name = input("Enter your name: ")

    roll_number = "21B-111-SE"
    department = "Software Engineering"

    parent_conn, child_conn = Pipe()

    child = Process(target=child_process, args=(child_conn,))

    child.start()

    parent_conn.send((name, roll_number, department))

    child.join()

if __name__ == "__main__":
    main()
```

OUTPU:



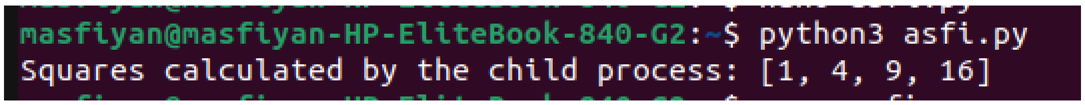
```
masfiyan@masfiyan-HP-EliteBook-840-G2:~$ python3 asfi.py
Enter your name: Muhammad Asfiyan
Name: Muhammad Asfiyan, Roll Number: 21B-111-SE, Department: Software Engineering
masfiyan@masfiyan-HP-EliteBook-840-G2:~$ nano asfi.py
```

QUESTION 2:



```python
from multiprocessing import Process, Pipe

def child_process(conn):
    squares = [i**2 for i in range(1, 5)]
    conn.send(squares)
    conn.close()

def main():
    parent_conn, child_conn = Pipe()
    child = Process(target=child_process, args=(child_conn,))
    child.start()
    child.join()
    squares = parent_conn.recv()
    print("Squares calculated by the child process:", squares)

if __name__ == "__main__":
    main()
```

OUTPUT:



```
masfiyan@masfiyan-HP-EliteBook-840-G2:~$ python3 asfi.py
Squares calculated by the child process: [1, 4, 9, 16]
```

USMAN INSTITUTE OF TECHNOLOGY

Affiliated with NED University of Engineering & Technology, Karachi

Department of Computer Science Fall-2023

CS312 – Operating System
Lab #12

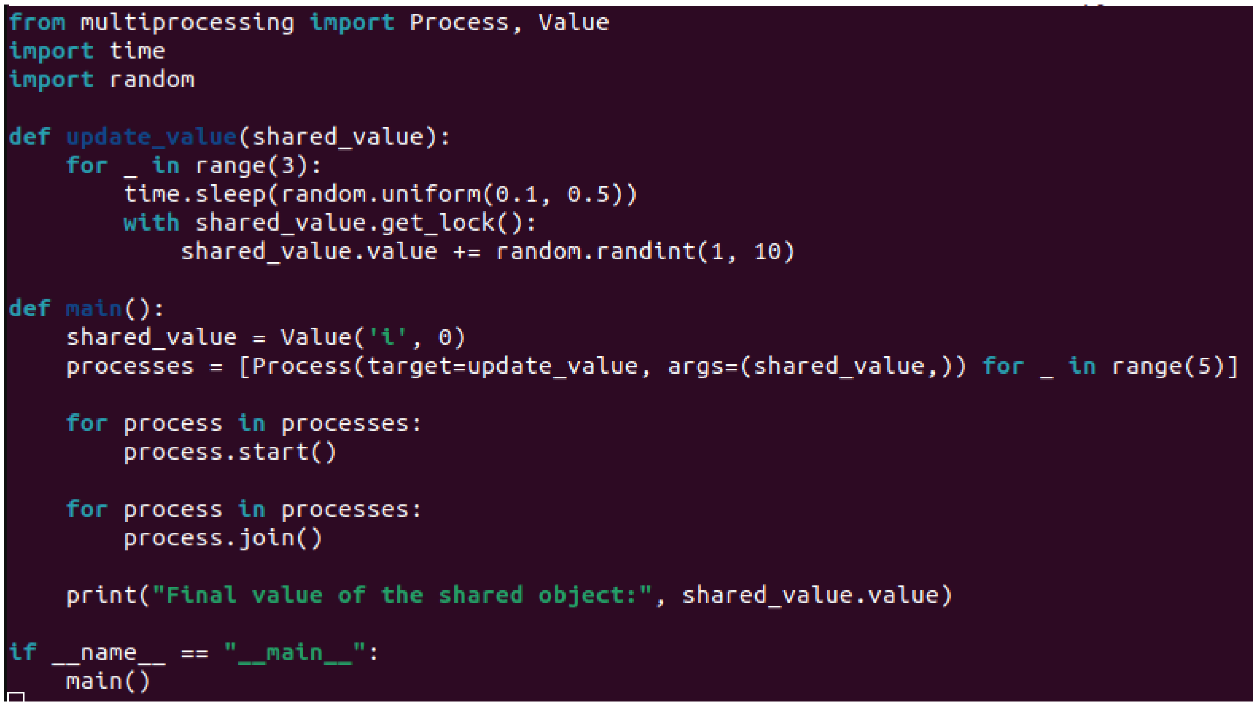
Objective:

Executing some of the most frequently used Linux commands

To understand and use basic utilities/Commands of UNIX/LINUX.

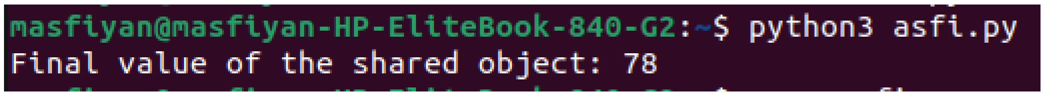
Name of Student	**Muhammad Asfiyan**
Student ID	**21B-111-SE**
Marks Obtained	
Remarks	
Signature	

QUESTION 1:



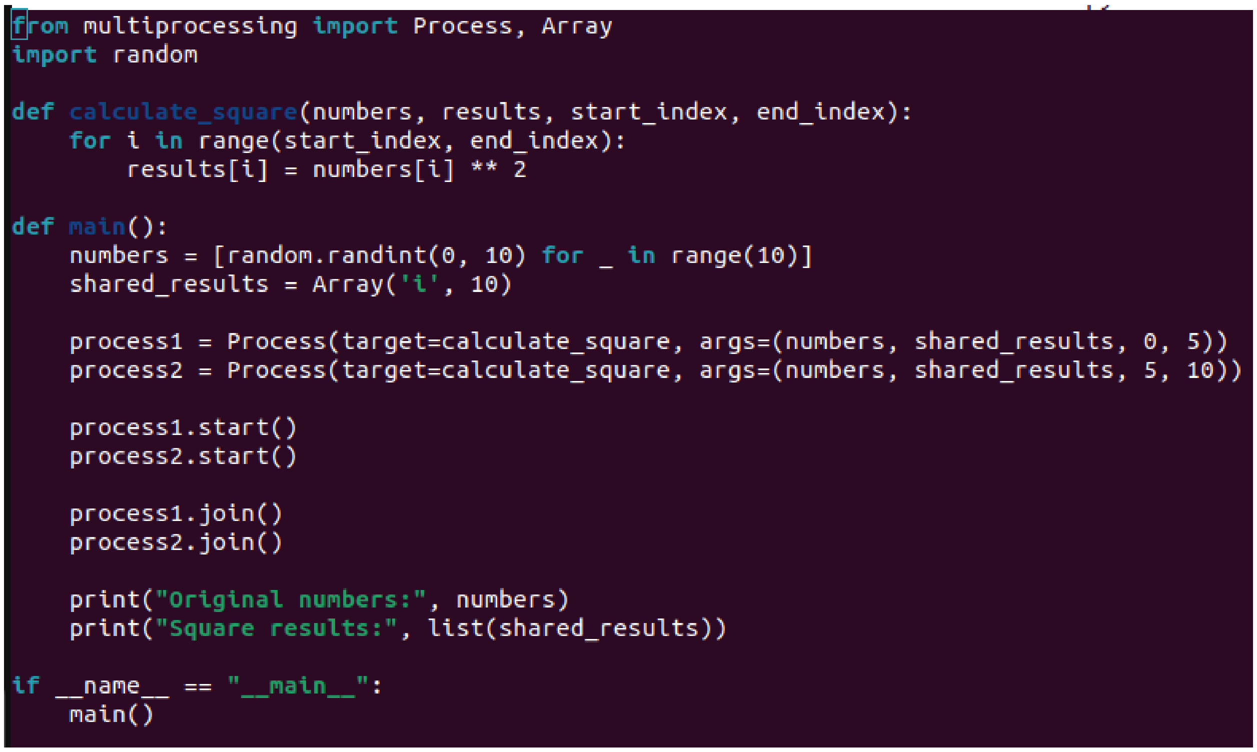
```
from multiprocessing import Process, Value
import time
import random

def update_value(shared_value):
    for _ in range(3):
        time.sleep(random.uniform(0.1, 0.5))
        with shared_value.get_lock():
            shared_value.value += random.randint(1, 10)

def main():
    shared_value = Value('i', 0)
    processes = [Process(target=update_value, args=(shared_value,)) for _ in range(5)]

    for process in processes:
        process.start()

    for process in processes:
        process.join()

    print("Final value of the shared object:", shared_value.value)

if __name__ == "__main__":
    main()
```

OUTPUT:



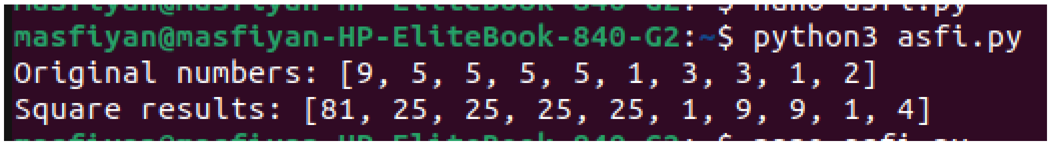
```
masfiyan@masfiyan-HP-EliteBook-840-G2:~$ python3 asfi.py
Final value of the shared object: 78
```

QUESTION 2:



```
from multiprocessing import Process, Array
import random

def calculate_square(numbers, results, start_index, end_index):
    for i in range(start_index, end_index):
        results[i] = numbers[i] ** 2

def main():
    numbers = [random.randint(0, 10) for _ in range(10)]
    shared_results = Array('i', 10)

    process1 = Process(target=calculate_square, args=(numbers, shared_results, 0, 5))
    process2 = Process(target=calculate_square, args=(numbers, shared_results, 5, 10))

    process1.start()
    process2.start()

    process1.join()
    process2.join()

    print("Original numbers:", numbers)
    print("Square results:", list(shared_results))

if __name__ == "__main__":
    main()
```

OUTPUT:



```
masfiyan@masfiyan-HP-EliteBook-840-G2:~$ nano asfi.py
masfiyan@masfiyan-HP-EliteBook-840-G2:~$ python3 asfi.py
Original numbers: [9, 5, 5, 5, 5, 1, 3, 3, 1, 2]
Square results: [81, 25, 25, 25, 25, 1, 9, 9, 1, 4]
masfiyan@masfiyan-HP-EliteBook-840-G2:~$ nano asfi.py
```

USMAN INSTITUTE OF TECHNOLOGY

Affiliated with NED University of Engineering & Technology, Karachi

Department of Computer Science Fall-2023

CS312 – Operating System Lab #13

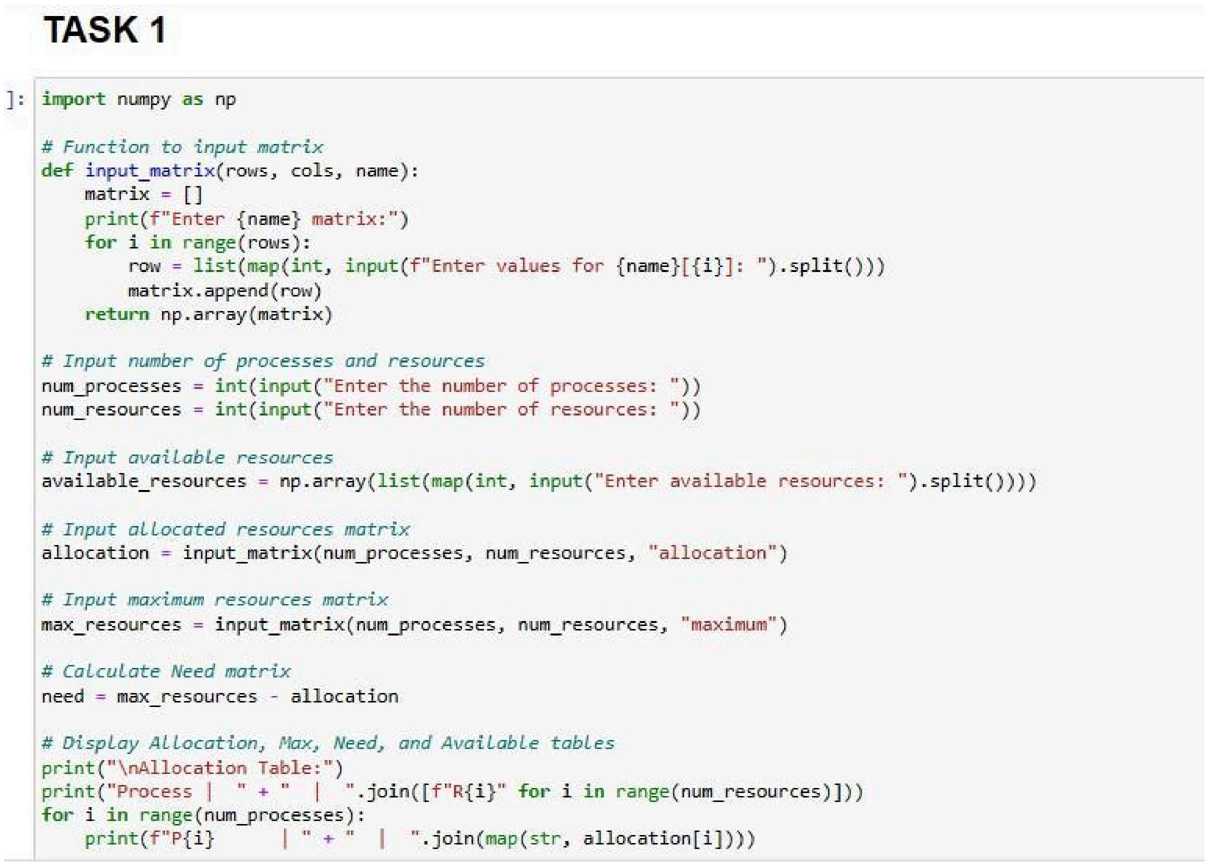
Objective:

Executing some of the most frequently used Linux commands

To understand and use basic utilities/Commands of UNIX/LINUX.

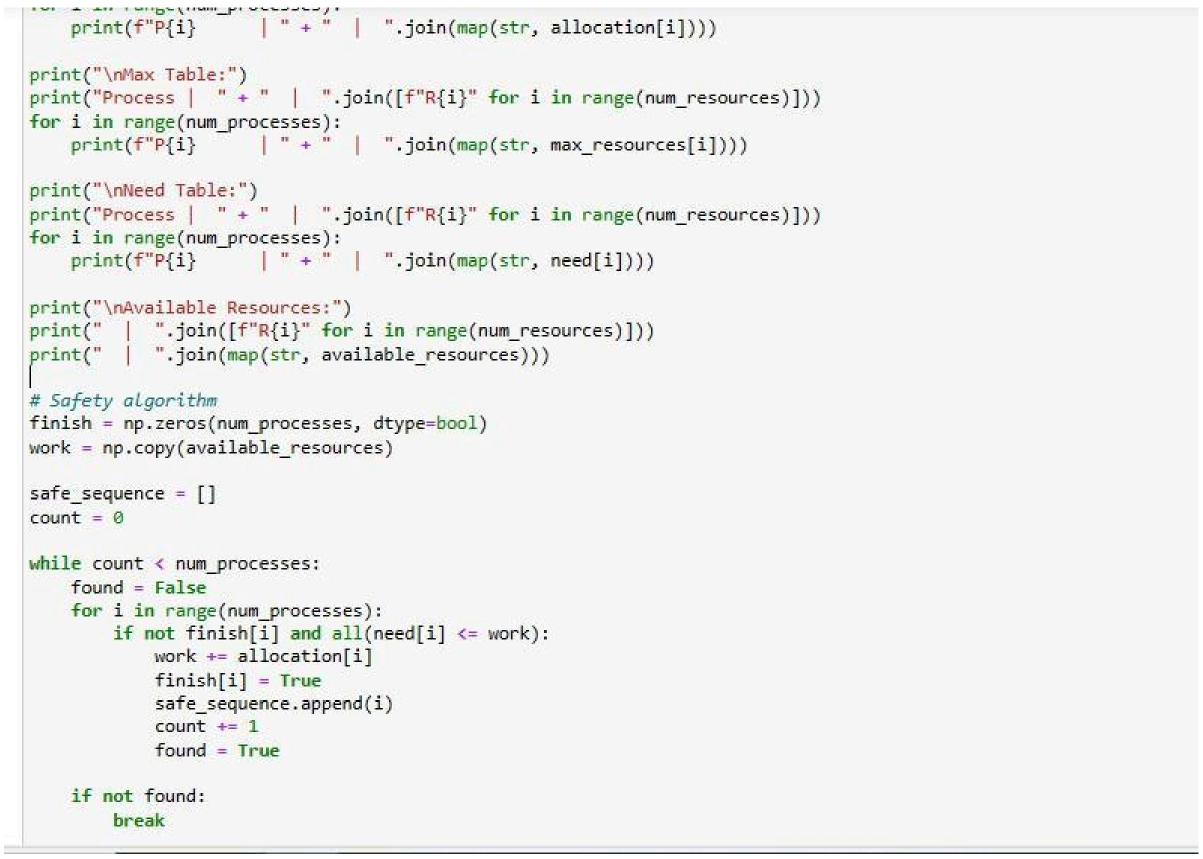
Name of Student	**Muhammad Asfiyan**
Student ID	**21B-111-SE**
Marks Obtained	
Remarks	
Signature	

TASK 1

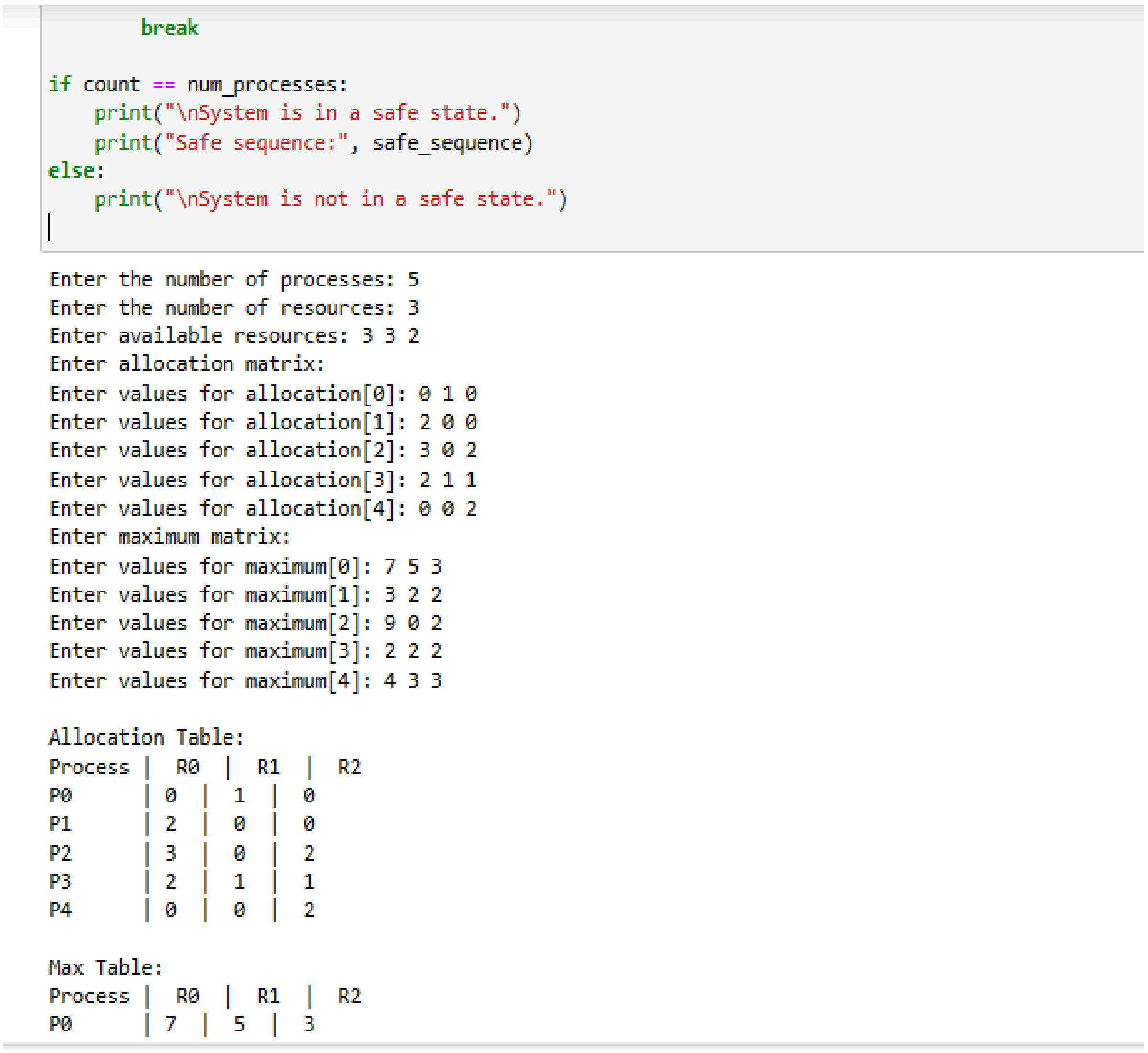


TASK 1

```
]: import numpy as np

   # Function to input matrix
   def input_matrix(rows, cols, name):
       matrix = []
       print(f"Enter {name} matrix:")
       for i in range(rows):
           row = list(map(int, input(f"Enter values for {name}[{i}]: ").split()))
           matrix.append(row)
       return np.array(matrix)

   # Input number of processes and resources
   num_processes = int(input("Enter the number of processes: "))
   num_resources = int(input("Enter the number of resources: "))

   # Input available resources
   available_resources = np.array(list(map(int, input("Enter available resources: ").split())))

   # Input allocated resources matrix
   allocation = input_matrix(num_processes, num_resources, "allocation")

   # Input maximum resources matrix
   max_resources = input_matrix(num_processes, num_resources, "maximum")

   # Calculate Need matrix
   need = max_resources - allocation

   # Display Allocation, Max, Need, and Available tables
   print("\nAllocation Table:")
   print("Process |  " + "  |  ".join([f"R{i}" for i in range(num_resources)]))
   for i in range(num_processes):
       print(f"P{i}      | " + "  |  ".join(map(str, allocation[i])))
```

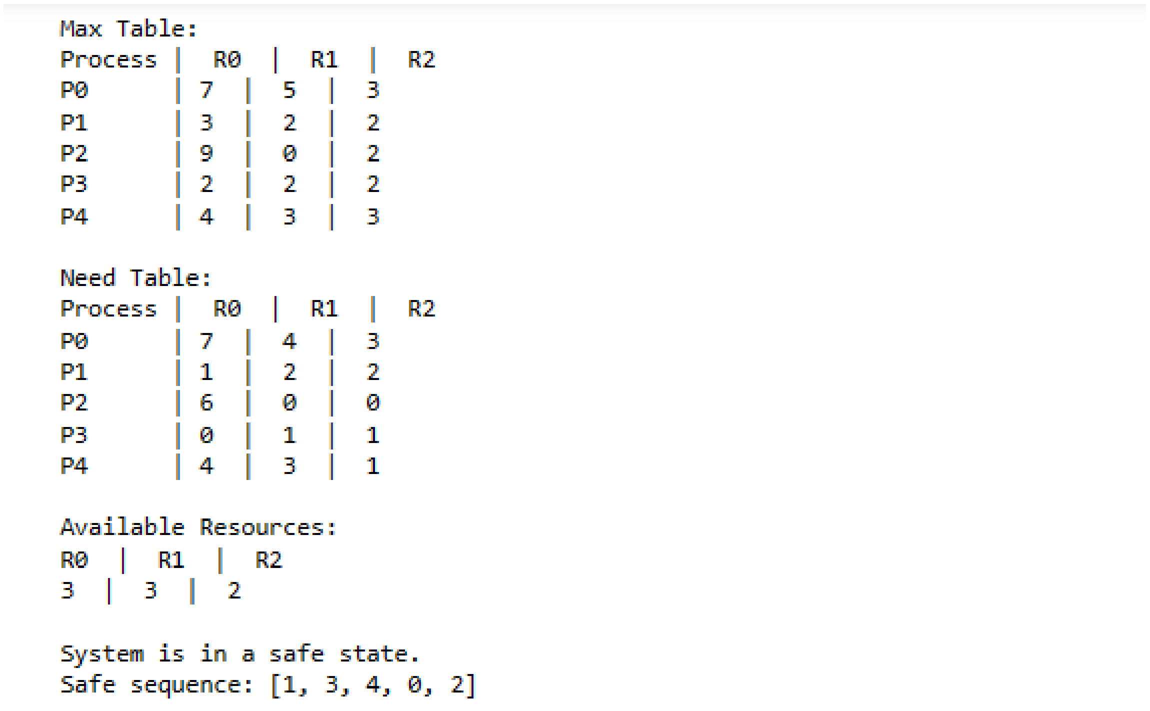


```
for i in range(num_processes):
    print(f"P{i}      | " + "  |  ".join(map(str, allocation[i])))

print("\nMax Table:")
print("Process |  " + "  |  ".join([f"R{i}" for i in range(num_resources)]))
for i in range(num_processes):
    print(f"P{i}      | " + "  |  ".join(map(str, max_resources[i])))

print("\nNeed Table:")
print("Process |  " + "  |  ".join([f"R{i}" for i in range(num_resources)]))
for i in range(num_processes):
    print(f"P{i}      | " + "  |  ".join(map(str, need[i])))

print("\nAvailable Resources:")
print("  |  ".join([f"R{i}" for i in range(num_resources)]))
print("  |  ".join(map(str, available_resources)))

# Safety algorithm
finish = np.zeros(num_processes, dtype=bool)
work = np.copy(available_resources)

safe_sequence = []
count = 0

while count < num_processes:
    found = False
    for i in range(num_processes):
        if not finish[i] and all(need[i] <= work):
            work += allocation[i]
            finish[i] = True
            safe_sequence.append(i)
            count += 1
            found = True

    if not found:
        break
```



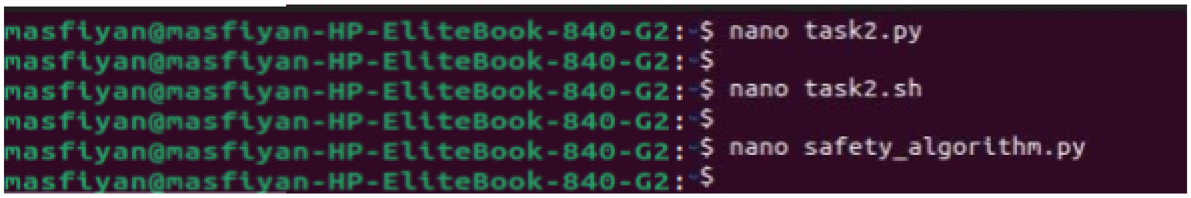
```python
        break

if count == num_processes:
    print("\nSystem is in a safe state.")
    print("Safe sequence:", safe_sequence)
else:
    print("\nSystem is not in a safe state.")
```

```
Enter the number of processes: 5
Enter the number of resources: 3
Enter available resources: 3 3 2
Enter allocation matrix:
Enter values for allocation[0]: 0 1 0
Enter values for allocation[1]: 2 0 0
Enter values for allocation[2]: 3 0 2
Enter values for allocation[3]: 2 1 1
Enter values for allocation[4]: 0 0 2
Enter maximum matrix:
Enter values for maximum[0]: 7 5 3
Enter values for maximum[1]: 3 2 2
Enter values for maximum[2]: 9 0 2
Enter values for maximum[3]: 2 2 2
Enter values for maximum[4]: 4 3 3

Allocation Table:
Process |  R0  |  R1  |  R2
P0      | 0  |  1  |  0
P1      | 2  |  0  |  0
P2      | 3  |  0  |  2
P3      | 2  |  1  |  1
P4      | 0  |  0  |  2

Max Table:
Process |  R0  |  R1  |  R2
P0      | 7  |  5  |  3
```

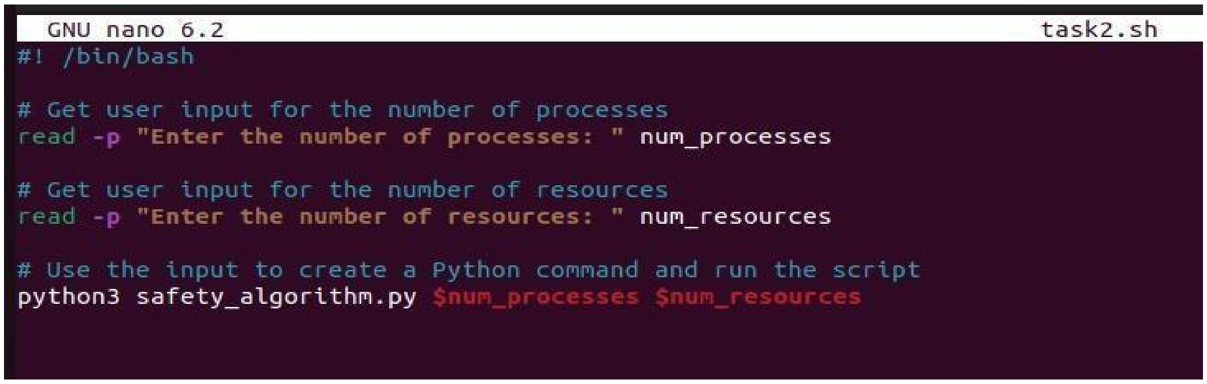


```
Max Table:
Process |  R0  |  R1  |  R2
P0      | 7  |  5  |  3
P1      | 3  |  2  |  2
P2      | 9  |  0  |  2
P3      | 2  |  2  |  2
P4      | 4  |  3  |  3

Need Table:
Process |  R0  |  R1  |  R2
P0      | 7  |  4  |  3
P1      | 1  |  2  |  2
P2      | 6  |  0  |  0
P3      | 0  |  1  |  1
P4      | 4  |  3  |  1

Available Resources:
R0  |  R1  |  R2
3  |  3  |  2

System is in a safe state.
Safe sequence: [1, 3, 4, 0, 2]
```

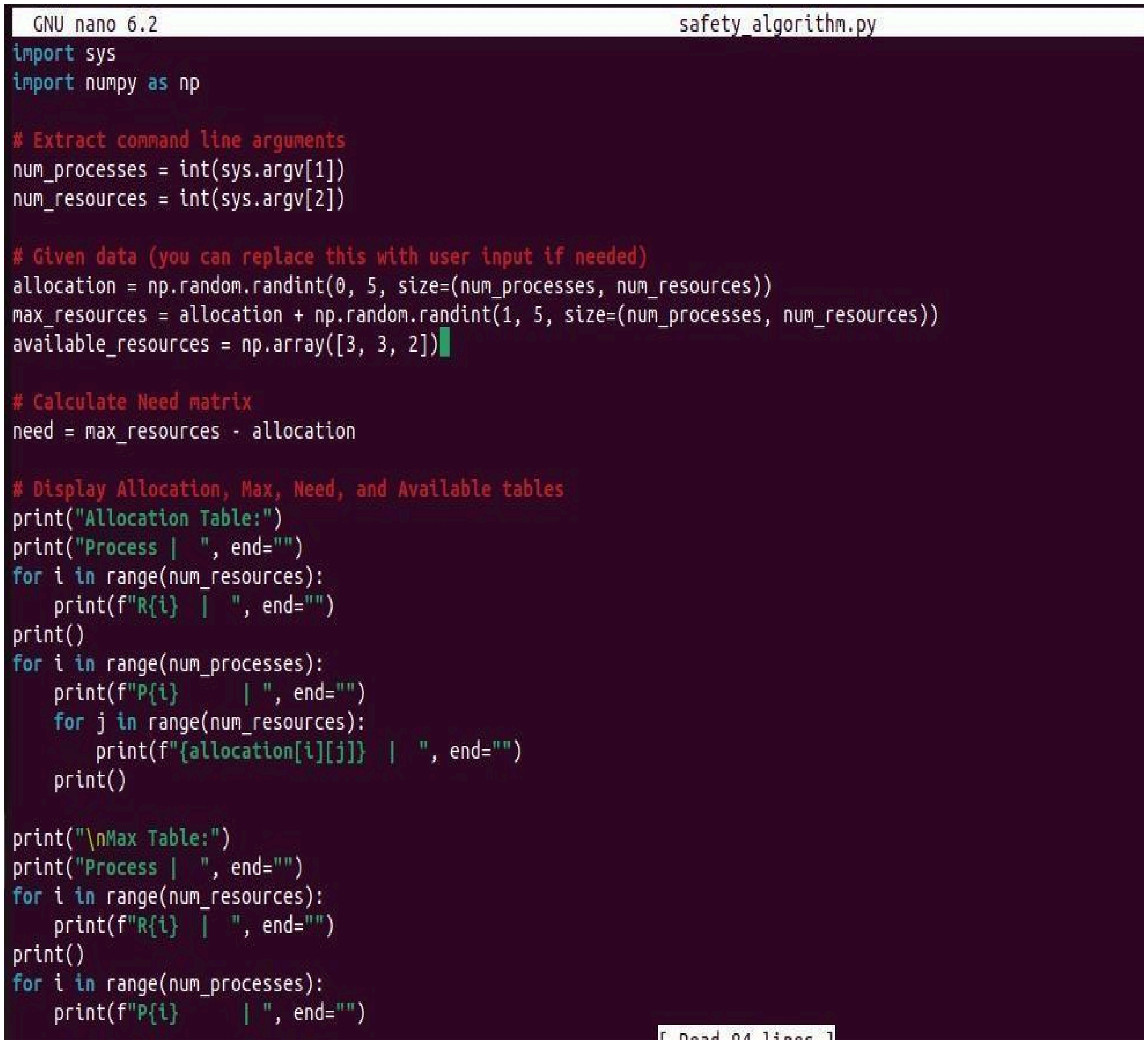
TASK 2



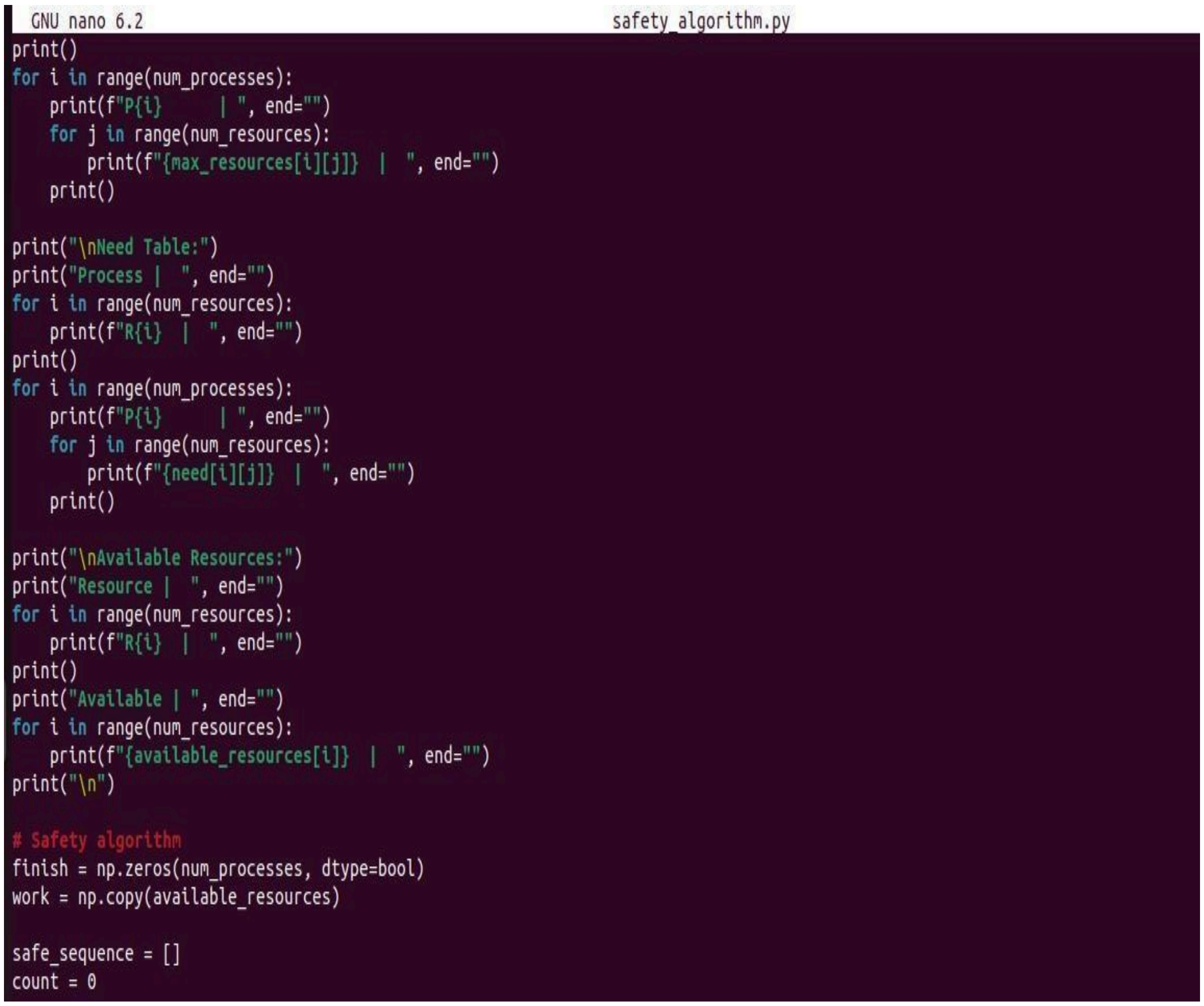
```
masfiyan@masfiyan-HP-EliteBook-840-G2:~$ nano task2.py
masfiyan@masfiyan-HP-EliteBook-840-G2:~$
masfiyan@masfiyan-HP-EliteBook-840-G2:~$ nano task2.sh
masfiyan@masfiyan-HP-EliteBook-840-G2:~$
masfiyan@masfiyan-HP-EliteBook-840-G2:~$ nano safety_algorithm.py
masfiyan@masfiyan-HP-EliteBook-840-G2:~$
```



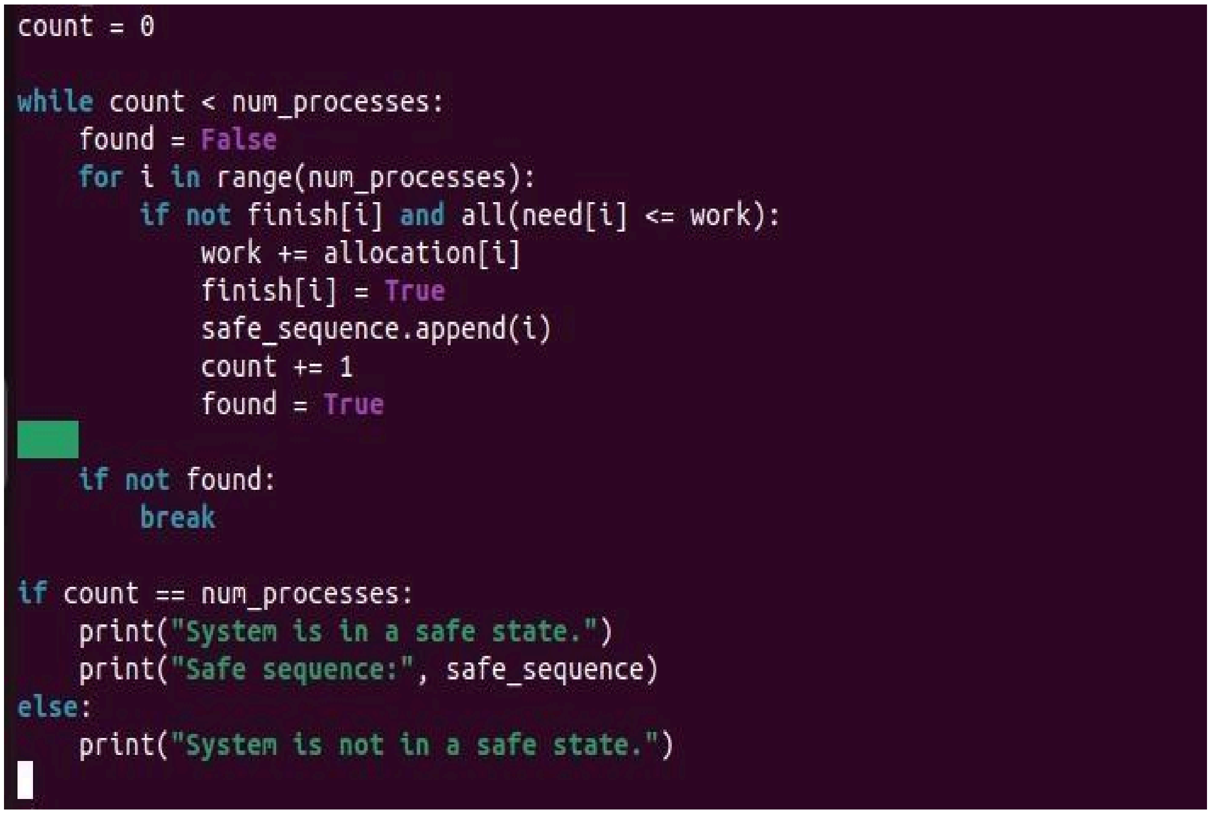
```
  GNU nano 6.2                                                      task2.sh
#! /bin/bash

# Get user input for the number of processes
read -p "Enter the number of processes: " num_processes

# Get user input for the number of resources
read -p "Enter the number of resources: " num_resources

# Use the input to create a Python command and run the script
python3 safety_algorithm.py $num_processes $num_resources
```



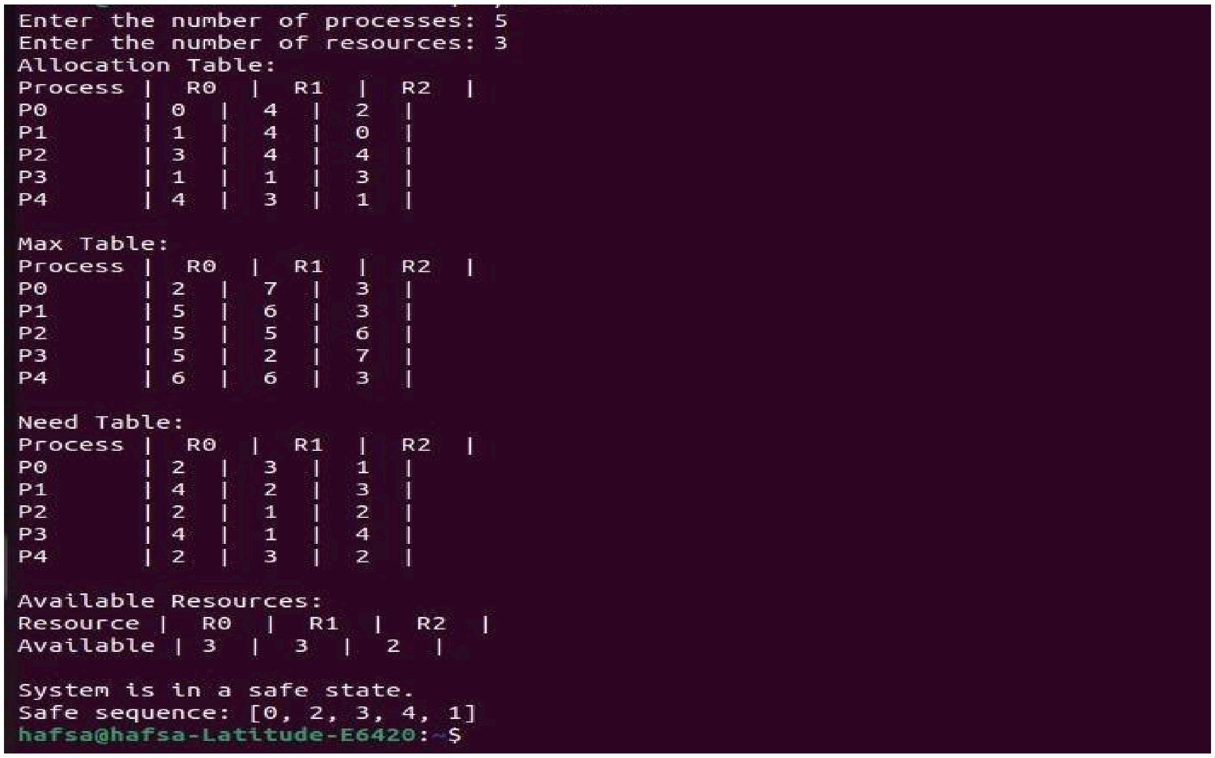
```
  GNU nano 6.2                                                  safety_algorithm.py
import sys
import numpy as np

# Extract command line arguments
num_processes = int(sys.argv[1])
num_resources = int(sys.argv[2])

# Given data (you can replace this with user input if needed)
allocation = np.random.randint(0, 5, size=(num_processes, num_resources))
max_resources = allocation + np.random.randint(1, 5, size=(num_processes, num_resources))
available_resources = np.array([3, 3, 2])

# Calculate Need matrix
need = max_resources - allocation

# Display Allocation, Max, Need, and Available tables
print("Allocation Table:")
print("Process |  ", end="")
for i in range(num_resources):
    print(f"R{i}  |  ", end="")
print()
for i in range(num_processes):
    print(f"P{i}      | ", end="")
    for j in range(num_resources):
        print(f"{allocation[i][j]}  |  ", end="")
    print()

print("\nMax Table:")
print("Process |  ", end="")
for i in range(num_resources):
    print(f"R{i}  |  ", end="")
print()
for i in range(num_processes):
    print(f"P{i}      | ", end="")
                                                         [ Read 94 lines ]
```



```
GNU nano 6.2                                   safety_algorithm.py
print()
for i in range(num_processes):
    print(f"P{i}      | ", end="")
    for j in range(num_resources):
        print(f"{max_resources[i][j]}  |  ", end="")
    print()

print("\nNeed Table:")
print("Process |  ", end="")
for i in range(num_resources):
    print(f"R{i}  |  ", end="")
print()
for i in range(num_processes):
    print(f"P{i}      | ", end="")
    for j in range(num_resources):
        print(f"{need[i][j]}  |  ", end="")
    print()

print("\nAvailable Resources:")
print("Resource |  ", end="")
for i in range(num_resources):
    print(f"R{i}  |  ", end="")
print()
print("Available | ", end="")
for i in range(num_resources):
    print(f"{available_resources[i]}  |  ", end="")
print("\n")

# Safety algorithm
finish = np.zeros(num_processes, dtype=bool)
work = np.copy(available_resources)

safe_sequence = []
count = 0
```



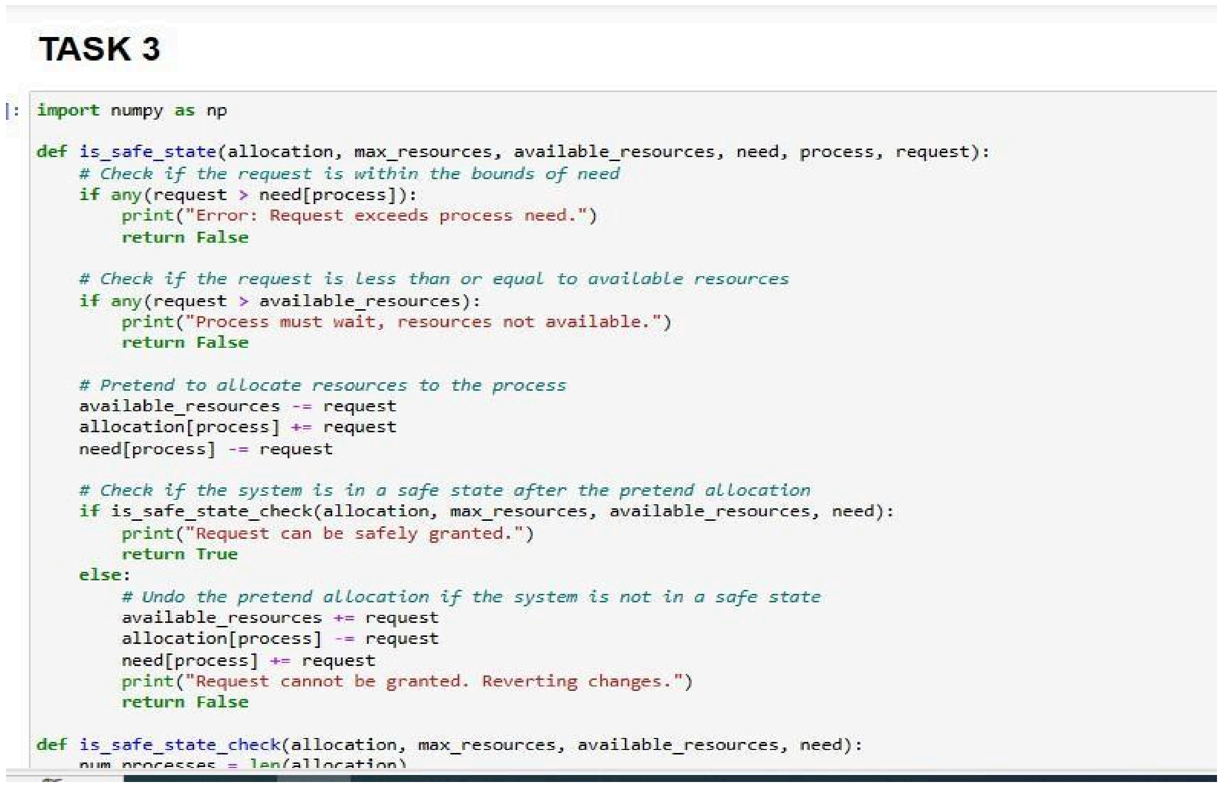
```
count = 0

while count < num_processes:
    found = False
    for i in range(num_processes):
        if not finish[i] and all(need[i] <= work):
            work += allocation[i]
            finish[i] = True
            safe_sequence.append(i)
            count += 1
            found = True

    if not found:
        break

if count == num_processes:
    print("System is in a safe state.")
    print("Safe sequence:", safe_sequence)
else:
    print("System is not in a safe state.")
```



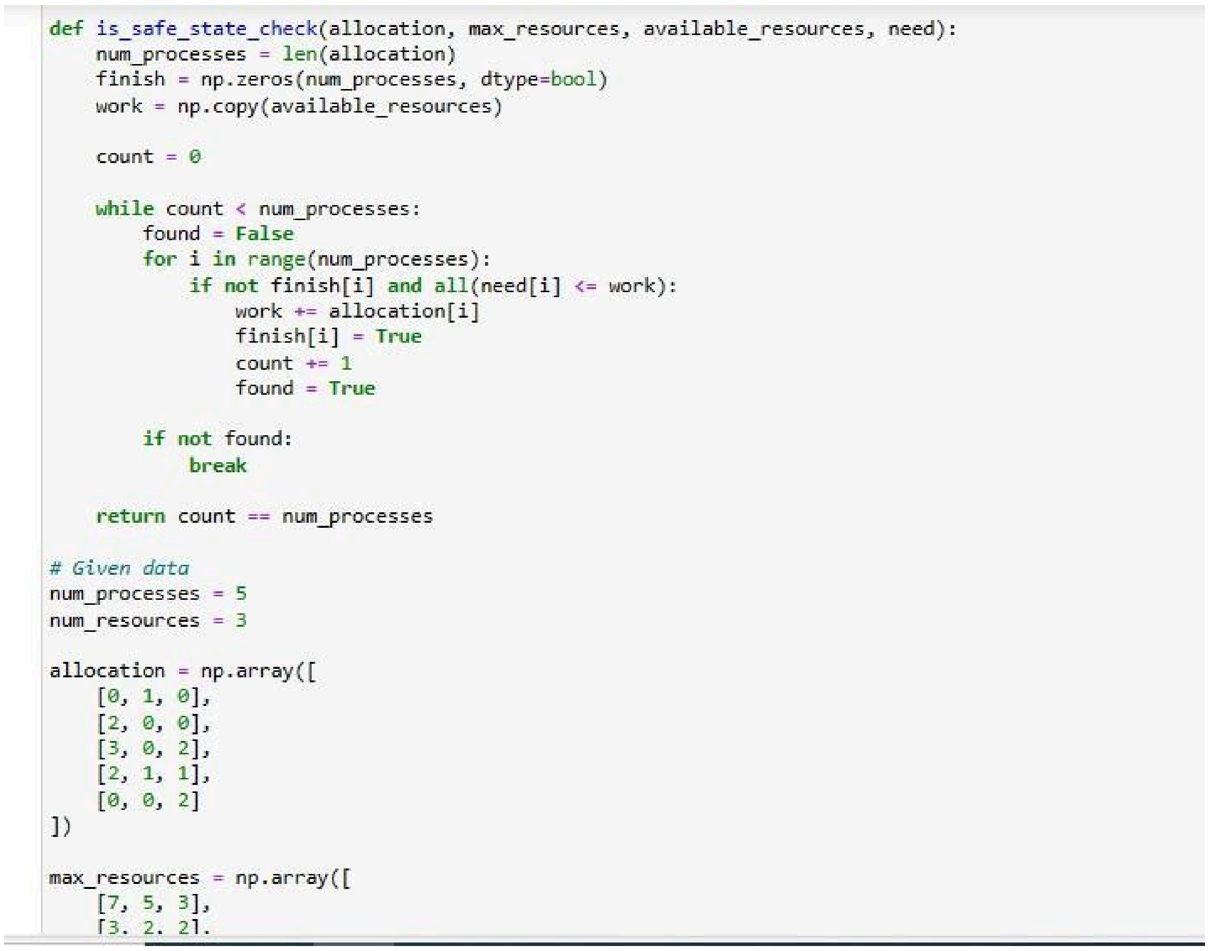
```
Enter the number of processes: 5
Enter the number of resources: 3
Allocation Table:
Process |  R0  |  R1  |  R2  |
P0      | 0  |  4  |  2  |
P1      | 1  |  4  |  0  |
P2      | 3  |  4  |  4  |
P3      | 1  |  1  |  3  |
P4      | 4  |  3  |  1  |

Max Table:
Process |  R0  |  R1  |  R2  |
P0      | 2  |  7  |  3  |
P1      | 5  |  6  |  3  |
P2      | 5  |  5  |  6  |
P3      | 5  |  2  |  7  |
P4      | 6  |  6  |  3  |

Need Table:
Process |  R0  |  R1  |  R2  |
P0      | 2  |  3  |  1  |
P1      | 4  |  2  |  3  |
P2      | 2  |  1  |  2  |
P3      | 4  |  1  |  4  |
P4      | 2  |  3  |  2  |

Available Resources:
Resource |  R0  |  R1  |  R2  |
Available | 3  |  3  |  2  |

System is in a safe state.
Safe sequence: [0, 2, 3, 4, 1]
hafsa@hafsa-Latitude-E6420:~$
```

TASK 3

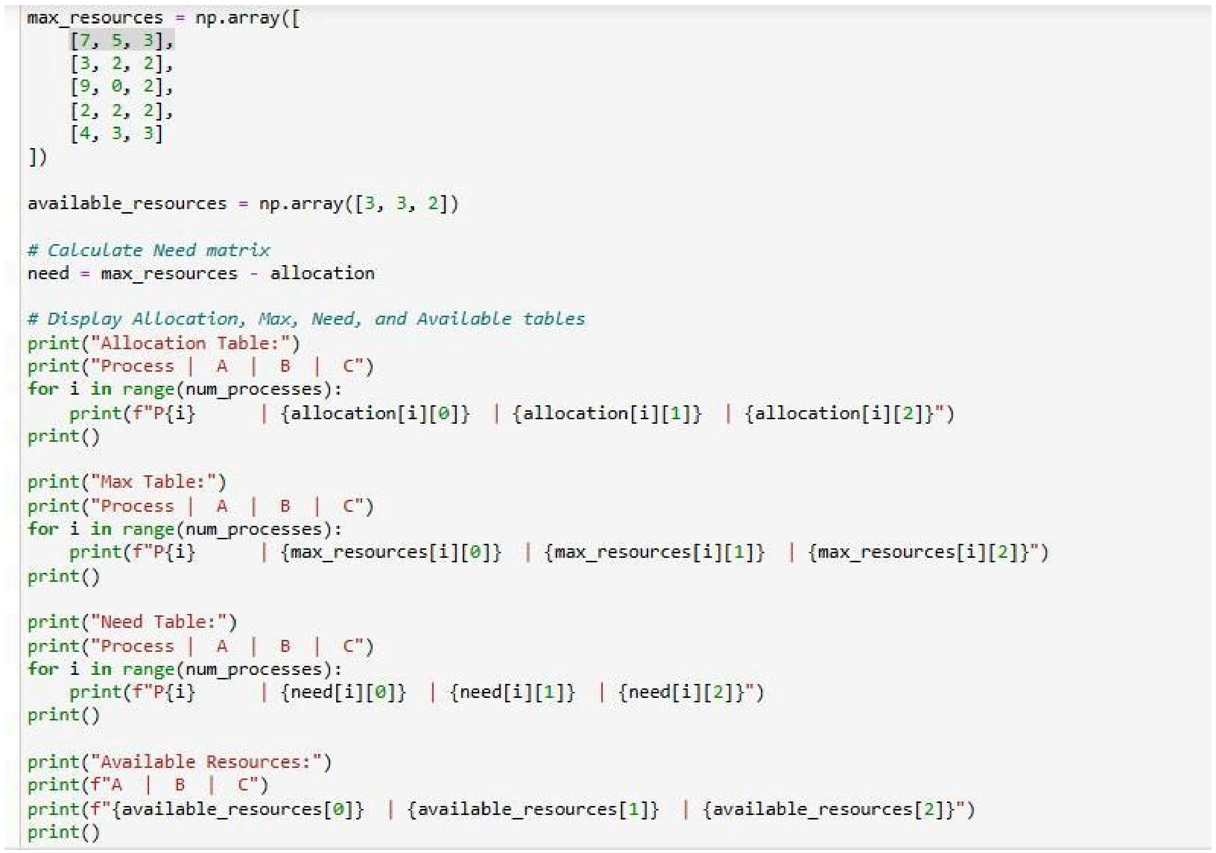
TASK 3



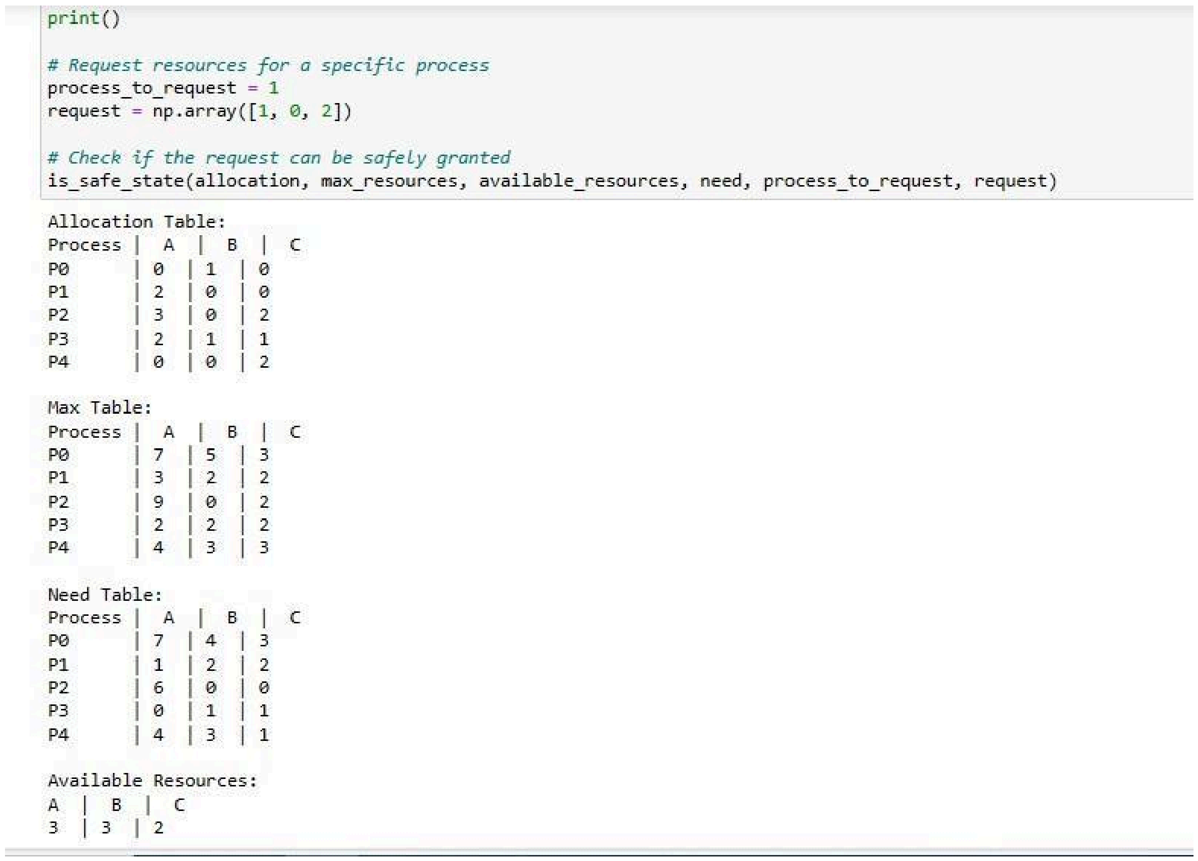
```python
import numpy as np

def is_safe_state(allocation, max_resources, available_resources, need, process, request):
    # Check if the request is within the bounds of need
    if any(request > need[process]):
        print("Error: Request exceeds process need.")
        return False

    # Check if the request is less than or equal to available resources
    if any(request > available_resources):
        print("Process must wait, resources not available.")
        return False

    # Pretend to allocate resources to the process
    available_resources -= request
    allocation[process] += request
    need[process] -= request

    # Check if the system is in a safe state after the pretend allocation
    if is_safe_state_check(allocation, max_resources, available_resources, need):
        print("Request can be safely granted.")
        return True
    else:
        # Undo the pretend allocation if the system is not in a safe state
        available_resources += request
        allocation[process] -= request
        need[process] += request
        print("Request cannot be granted. Reverting changes.")
        return False

def is_safe_state_check(allocation, max_resources, available_resources, need):
    num_processes = len(allocation)
```



```python
def is_safe_state_check(allocation, max_resources, available_resources, need):
    num_processes = len(allocation)
    finish = np.zeros(num_processes, dtype=bool)
    work = np.copy(available_resources)

    count = 0

    while count < num_processes:
        found = False
        for i in range(num_processes):
            if not finish[i] and all(need[i] <= work):
                work += allocation[i]
                finish[i] = True
                count += 1
                found = True

        if not found:
            break

    return count == num_processes

# Given data
num_processes = 5
num_resources = 3

allocation = np.array([
    [0, 1, 0],
    [2, 0, 0],
    [3, 0, 2],
    [2, 1, 1],
    [0, 0, 2]
])

max_resources = np.array([
    [7, 5, 3],
    [3, 2, 2],
```



```
max_resources = np.array([
    [7, 5, 3],
    [3, 2, 2],
    [9, 0, 2],
    [2, 2, 2],
    [4, 3, 3]
])

available_resources = np.array([3, 3, 2])

# Calculate Need matrix
need = max_resources - allocation

# Display Allocation, Max, Need, and Available tables
print("Allocation Table:")
print("Process |  A  |  B  |  C")
for i in range(num_processes):
    print(f"P{i}      | {allocation[i][0]}  | {allocation[i][1]}  | {allocation[i][2]}")
print()

print("Max Table:")
print("Process |  A  |  B  |  C")
for i in range(num_processes):
    print(f"P{i}      | {max_resources[i][0]}  | {max_resources[i][1]}  | {max_resources[i][2]}")
print()

print("Need Table:")
print("Process |  A  |  B  |  C")
for i in range(num_processes):
    print(f"P{i}      | {need[i][0]}  | {need[i][1]}  | {need[i][2]}")
print()

print("Available Resources:")
print(f"A  |  B  |  C")
print(f"{available_resources[0]}  | {available_resources[1]}  | {available_resources[2]}")
print()
```



```
print()

# Request resources for a specific process
process_to_request = 1
request = np.array([1, 0, 2])

# Check if the request can be safely granted
is_safe_state(allocation, max_resources, available_resources, need, process_to_request, request)
```

```
Allocation Table:
Process |  A  |  B  |  C
P0      | 0  | 1  | 0
P1      | 2  | 0  | 0
P2      | 3  | 0  | 2
P3      | 2  | 1  | 1
P4      | 0  | 0  | 2

Max Table:
Process |  A  |  B  |  C
P0      | 7  | 5  | 3
P1      | 3  | 2  | 2
P2      | 9  | 0  | 2
P3      | 2  | 2  | 2
P4      | 4  | 3  | 3

Need Table:
Process |  A  |  B  |  C
P0      | 7  | 4  | 3
P1      | 1  | 2  | 2
P2      | 6  | 0  | 0
P3      | 0  | 1  | 1
P4      | 4  | 3  | 1

Available Resources:
A  |  B  |  C
3  | 3  | 2
```

```
Available Resources:
A  |  B  |  C
3  | 3  | 2

Request can be safely granted.

Out[1]: True
```

USMAN INSTITUTE OF TECHNOLOGY

Affiliated with NED University of Engineering & Technology, Karachi

Department of Computer Science Fall-2023

CS312 – Operating System Lab #14

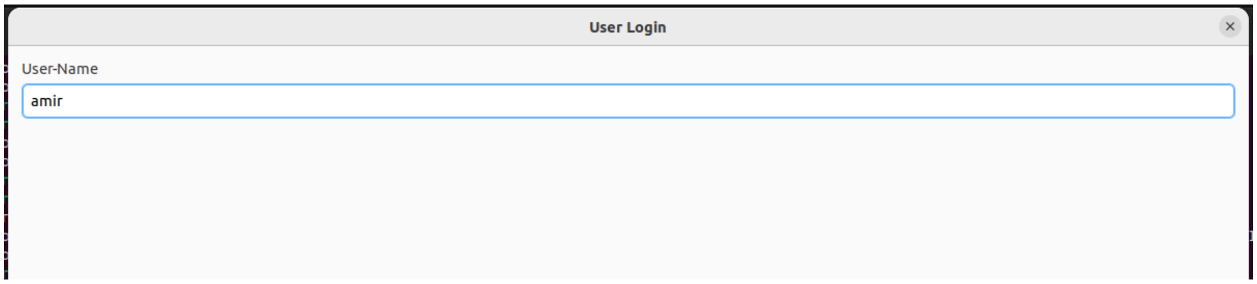
Objective:

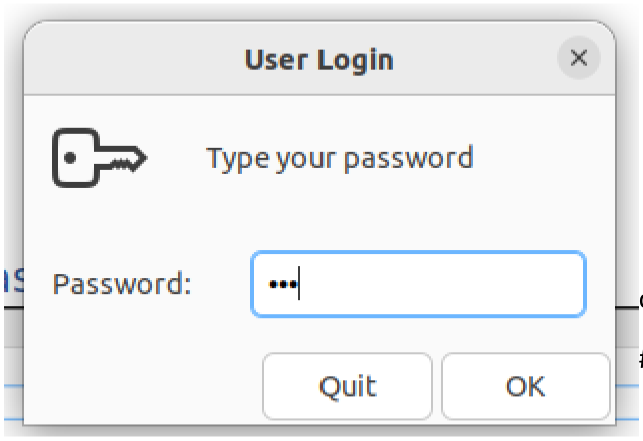
Executing some of the most frequently used Linux commands

To understand and use basic utilities/Commands of UNIX/LINUX.

Name of Student	**Muhammad Asfiyan**
Student ID	**21B-111-SE**
Marks Obtained	
Remarks	
Signature	

User Login and Password:





code:

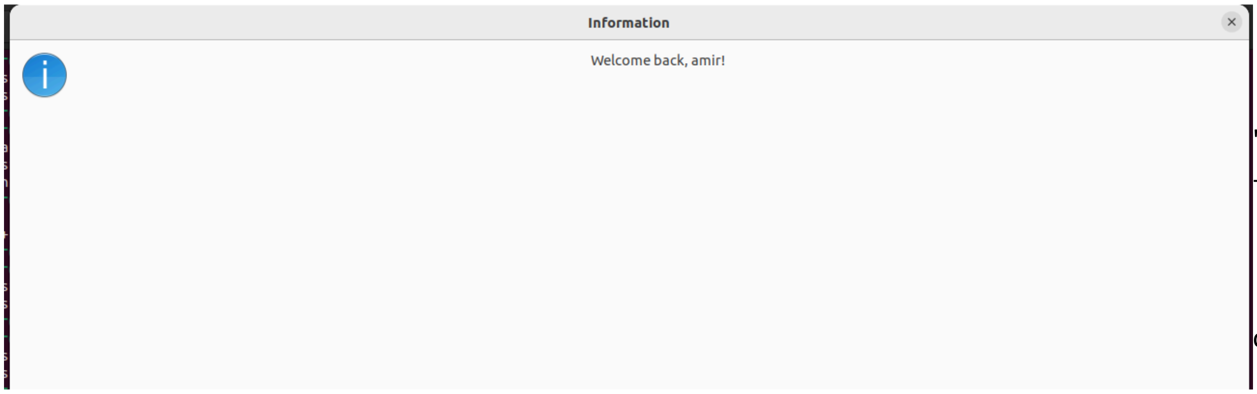
```
#user login

continue_loop=true


while [ "$continue_loop" = true ]; do
  USERNAME=$(zenity --entry --width=1200 --height=700 --title="User Login" --text="User-Name" --entry-text "" --extra-button "Add User" --ok-label "Login" --cancel-label "Quit" )
  BUTTON_CLICKED=$?


  # Check if the window is closed or the "Login" button is clicked
  if [ "$BUTTON_CLICKED" -eq 0 ]; then
    PASSWORD=$(zenity --password --width=1200 --height=700 --title="User Login" --text="Enter your password:" --cancel-label "Quit")


    # Check credentials
    if check_credentials "$USERNAME" "$PASSWORD"; then
      break
    else
      zenity --info --width=1200 --height=700 --text "Invalid credentials. Please try again." --ok-label "OK" --no-wrap
```

```
        fi
```

# Function to check if the user exists and validate the password

```
check_credentials() {
    local user=$1
    local password=$2

    # Validate the password
    if echo "$password" | su "$user" -c true 2>/dev/null; then
        zenity --info --width=1200 --height=700 --text "Welcome back, $user!" --ok-label "continue" --no-wrap
        return 0
    else
        zenity --info --width=1200 --height=700 --text "Incorrect password for $user. Please try again." --ok-label "OK" --no-wrap
        return 1
    fi
}
```

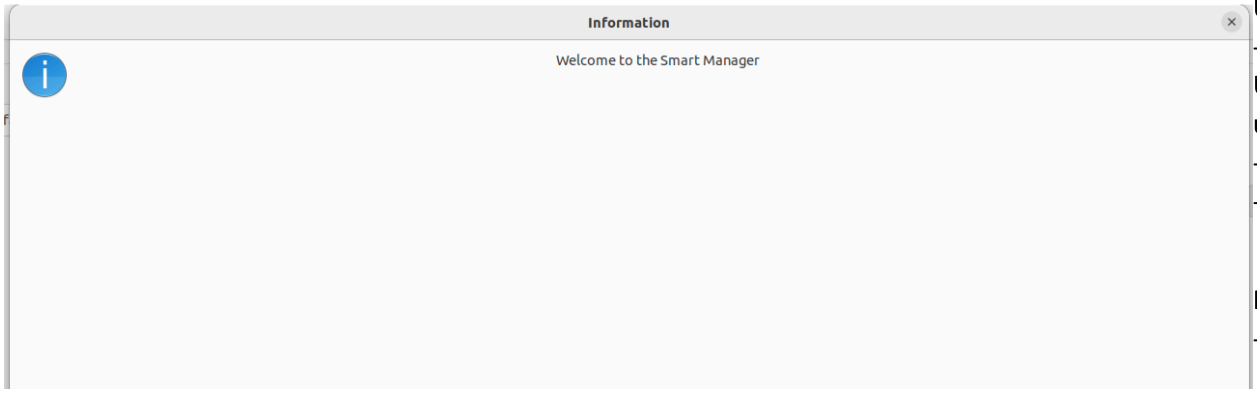
WELCOME



Information

Welcome back, amir!

```
 elif [
"$BUTTON_CLICKED"
-eq 1 ]; then

        # Add User button
clicked

USERNAME=$(zenity
--entry --title="Add
User" --text="Enter new
username:"
--width=1200
--height=700)

PASSWORD=$(zenity
--password --title="Add
```

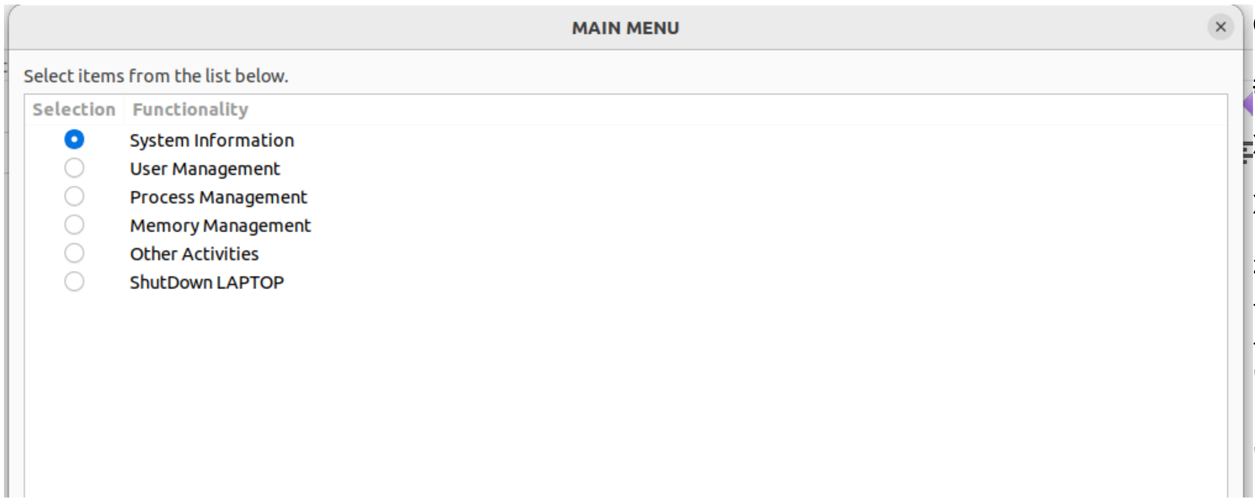


Information

Welcome to the Smart Manager

```
User" --text="Enter new password:" --width=1200 --height=700)

    # Add user
    add_user "$USERNAME" "$PASSWORD"
    zenity --info --width=1200 --height=700 --text "User $USERNAME added successfully!" --ok-label "Continue" --no-wrap
        else

            break
        fi


done

# Function to add user
add_user() {
  local user=$1
  local password=$2

  # Add user using the provided username and password
  sudo useradd -m -p $(openssl passwd -1 "$password") $user
}
```

MAIN-MENU:



MAIN MENU

Select items from the list below.

Selection	Functionality
●	System Information
○	User Management
○	Process Management
○	Memory Management
○	Other Activities
○	ShutDown LAPTOP

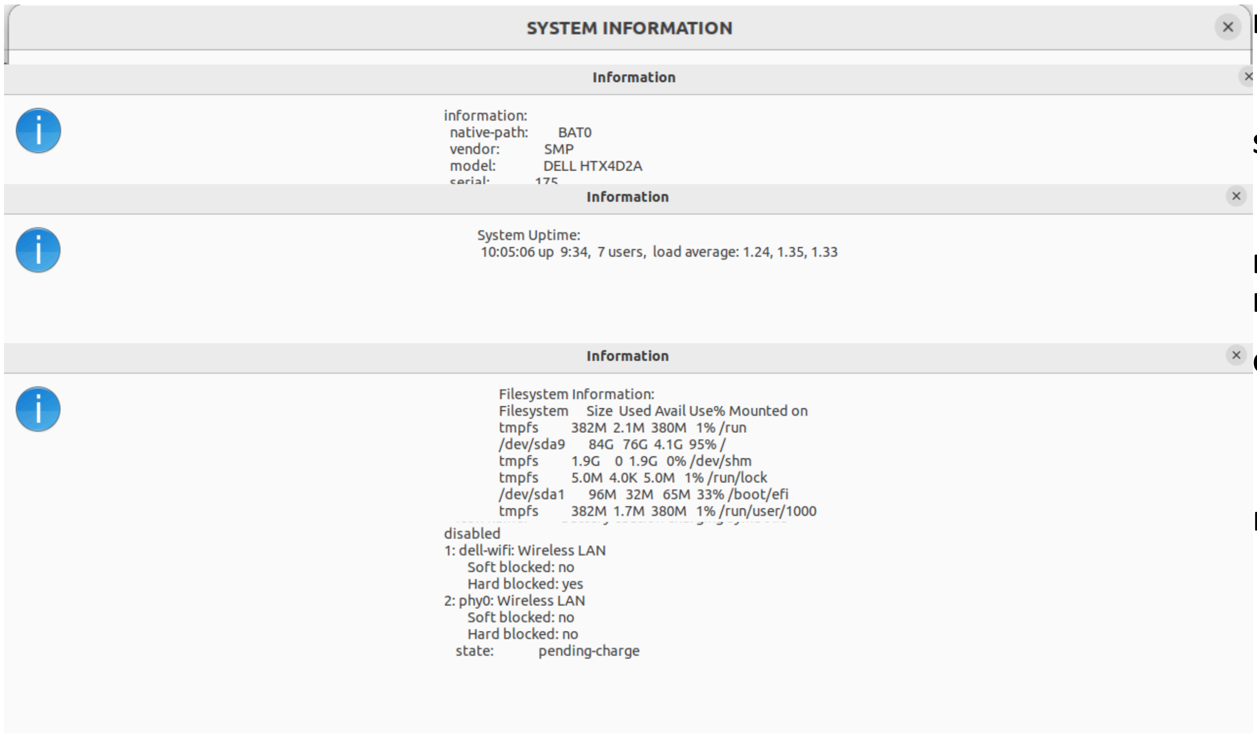
CODE:

# Main functionality list

```
X=0
X1=0
zenity --info --width=1200 --height=700 --text "Welcome to the Smart Manager" --ok-label "continue" --no-wrap

while [ "$X" -eq "$X1" ]; do
```

```
# Main category selection
CATEGORY="$(zenity --list --title "MAIN MENU" --column Selection --column Functionality \
   TRUE "System Information" \
   FALSE "User Management" \
   FALSE "Process Management" \
   FALSE "Memory Management" \
   FALSE "Other Activities" \
               FALSE "ShutDown LAPTOP" \
   --width=1000 --height=800 --radiolist --cancel-label "Quit")"
Stat=$?
```

SYSTEM INFORMATION:



```
SYSTEM INFORMATION

Information

information:
 native-path:      BAT0
 vendor:         SMP
 model:          DELL HTX4D2A
 serial:        175

Information

System Uptime:
 10:05:06 up  9:34,  7 users,  load average: 1.24, 1.35, 1.33

Information

Filesystem Information:
Filesystem     Size  Used Avail Use% Mounted on
tmpfs          382M  2.1M  380M   1% /run
/dev/sda9       84G   76G  4.1G  95% /
tmpfs          1.9G     0  1.9G   0% /dev/shm
tmpfs          5.0M  4.0K  5.0M   1% /run/lock
/dev/sda1       96M   32M   65M  33% /boot/efi
tmpfs          382M  1.7M  380M   1% /run/user/1000
disabled
1: dell-wifi: Wireless LAN
     Soft blocked: no
     Hard blocked: yes
2: phy0: Wireless LAN
     Soft blocked: no
     Hard blocked: no
   state:          pending-charge
```

INFORMATION:

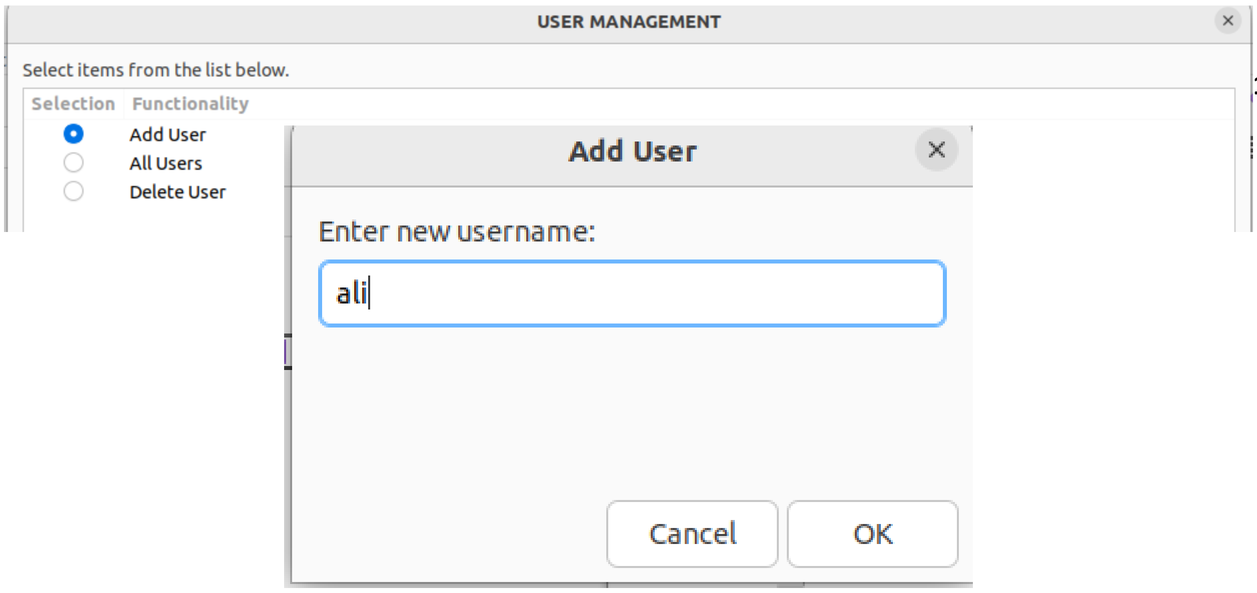
SYSTEM UPTIME:

FILE SYSTEM INFORMATION:

Code:

```
case "$CATEGORY" in
    "System Information")
        while true; do
SYSTEM_INFO_SELECTION="$(zenity --list --title "SYSTEM INFORMATION" --column Selection --column Functionality \
          TRUE "System Uptime" \
          FALSE "Filesystem Information" \
          --width=1000 --height=800 --radiolist --cancel-label "Back")"
        Stat=$?

        case "$SYSTEM_INFO_SELECTION" in
          "System Uptime")
          # Function to display system uptime
          SYSTEM_UPTIME=$(uptime)
          zenity --info --width=1200 --height=700 --text "System Uptime:\n$SYSTEM_UPTIME"
          ;;
        "Filesystem Information")
          # Function to display filesystem information
          FILESYSTEM_INFO=$(df -h)
          zenity --info --width=1200 --height=700 --text "Filesystem Information:\n$FILESYSTEM_INFO"
          ;;
        *)
          break  # Go back to the main category selection
          ;;
      esac
    done
    ;;
```

USER MANAGMENT:



USER MANAGEMENT

Select items from the list below.

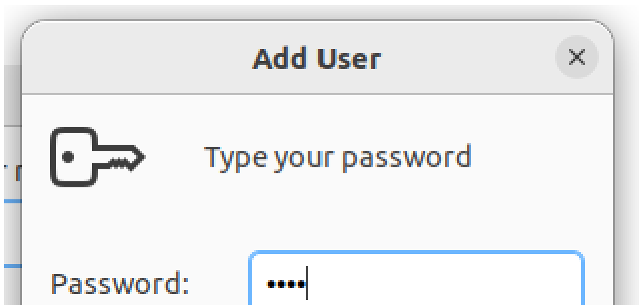
Selection	Functionality
●	Add User
○	All Users
○	Delete User

Add User

Enter new username:

ali

1.ADD USER:

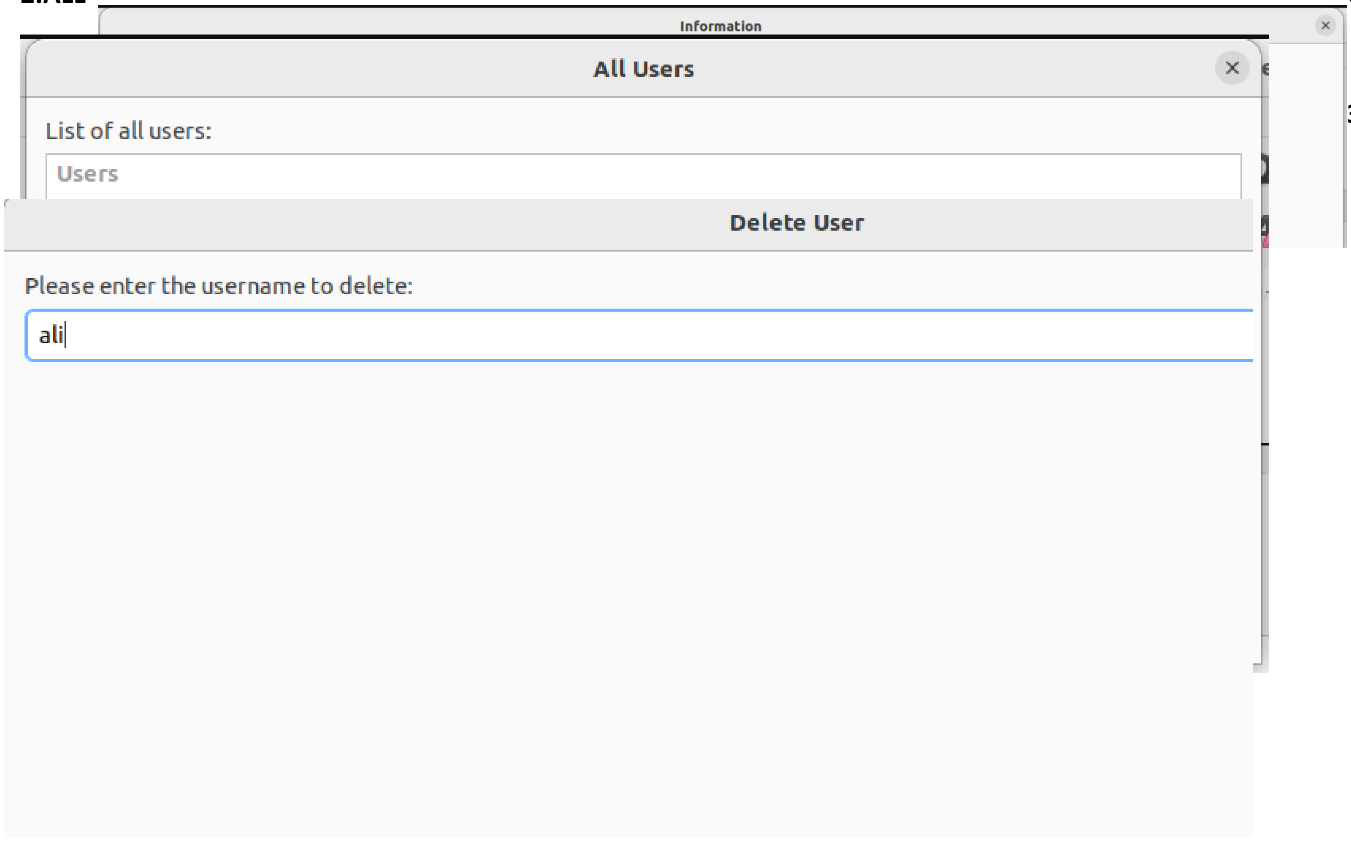


Add User

Type your password

Password: ••••

2.ALL USER



Information

All Users

List of all users:

Users

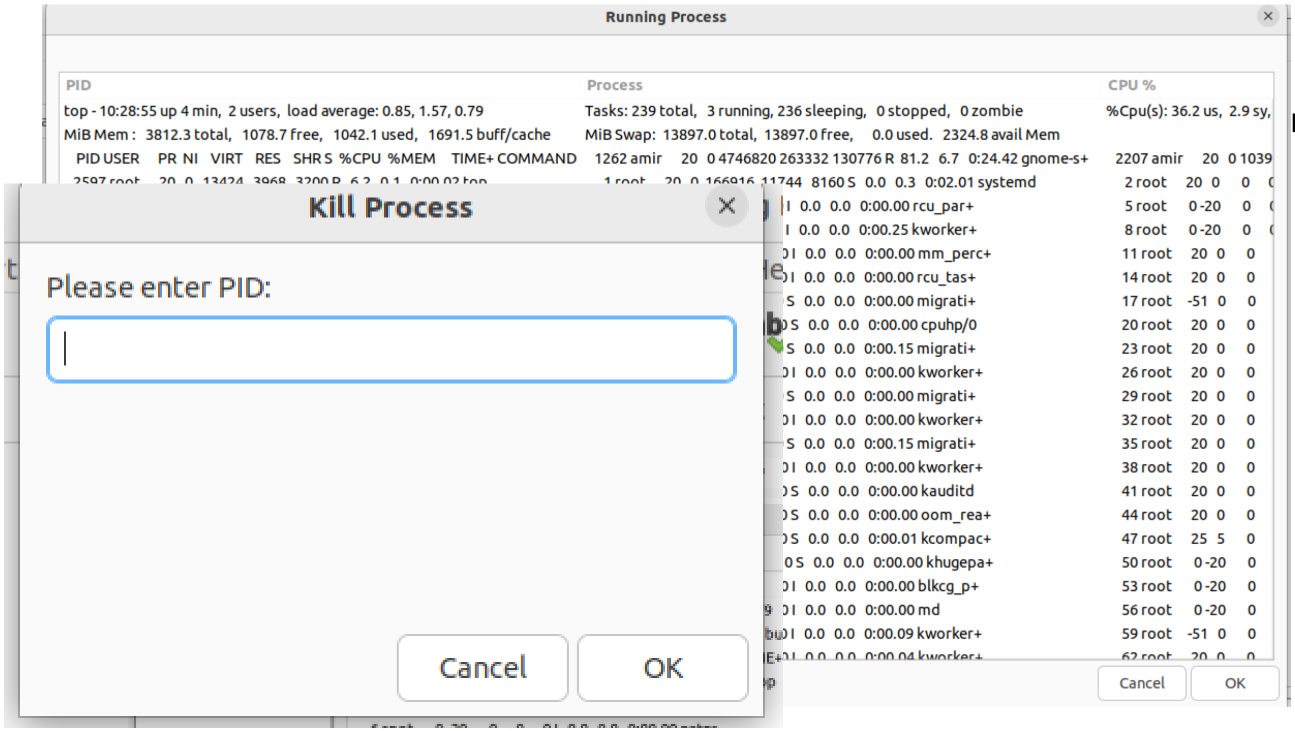
Delete User

Please enter the username to delete:

ali

3.Delete User

Process management



Running Process

```
PID                                                  Process                                                  CPU %
top - 10:28:55 up 4 min,  2 users,  load average: 0.85, 1.57, 0.79   Tasks: 239 total,   3 running, 236 sleeping,   0 stopped,   0 zombie   %Cpu(s): 36.2 us,  2.9 sy,
MiB Mem :  3812.3 total,  1078.7 free,  1042.1 used,  1691.5 buff/cache   MiB Swap:  13897.0 total,  13897.0 free,     0.0 used.  2324.8 avail Mem
  PID USER    PR  NI   VIRT   RES   SHR S  %CPU  %MEM    TIME+ COMMAND   1262 amir    20   0 4746820 263332 130776 R  81.2   6.7   0:24.42 gnome-s+   2207 amir    20   0 1039
 2597 root    20   0  13424  3968  3200 R   6.2   0.1   0:00.02 top        1 root    20   0 166916  11744   8160 S   0.0   0.3   0:02.01 systemd      2 root    20   0     0
                                                                                             I   0.0   0.0   0:00.00 rcu_par+      5 root     0 -20     0
                                                                                             I   0.0   0.0   0:00.25 kworker+      8 root     0 -20     0
                                                                                            0 I   0.0   0.0   0:00.00 mm_perc+     11 root    20   0     0
                                                                                            0 I   0.0   0.0   0:00.00 rcu_tas+     14 root    20   0     0
                                                                                            0 S   0.0   0.0   0:00.00 migrati+     17 root   -51   0     0
                                                                                            0 S   0.0   0.0   0:00.00 cpuhp/0      20 root    20   0     0
                                                                                            0 S   0.0   0.0   0:00.15 migrati+     23 root    20   0     0
                                                                                            0 I   0.0   0.0   0:00.00 kworker+     26 root    20   0     0
                                                                                            0 S   0.0   0.0   0:00.00 migrati+     29 root    20   0     0
                                                                                            0 I   0.0   0.0   0:00.00 kworker+     32 root    20   0     0
                                                                                            0 S   0.0   0.0   0:00.15 migrati+     35 root    20   0     0
                                                                                            0 I   0.0   0.0   0:00.00 kworker+     38 root    20   0     0
                                                                                            0 S   0.0   0.0   0:00.00 kauditd      41 root    20   0     0
                                                                                            0 S   0.0   0.0   0:00.00 oom_rea+     44 root    20   0     0
                                                                                            0 S   0.0   0.0   0:00.01 kcompac+     47 root    25   5     0
                                                                                            0 S   0.0   0.0   0:00.00 khugepa+     50 root     0 -20     0
                                                                                            0 I   0.0   0.0   0:00.00 blkcg_p+     53 root     0 -20     0
                                                                                            0 I   0.0   0.0   0:00.00 md           56 root     0 -20     0
                                                                                            0 I   0.0   0.0   0:00.09 kworker+     59 root   -51   0     0
                                                                                            0 I   0.0   0.0   0:00.04 kworker+     62 root    20   0     0
```

Cancel OK

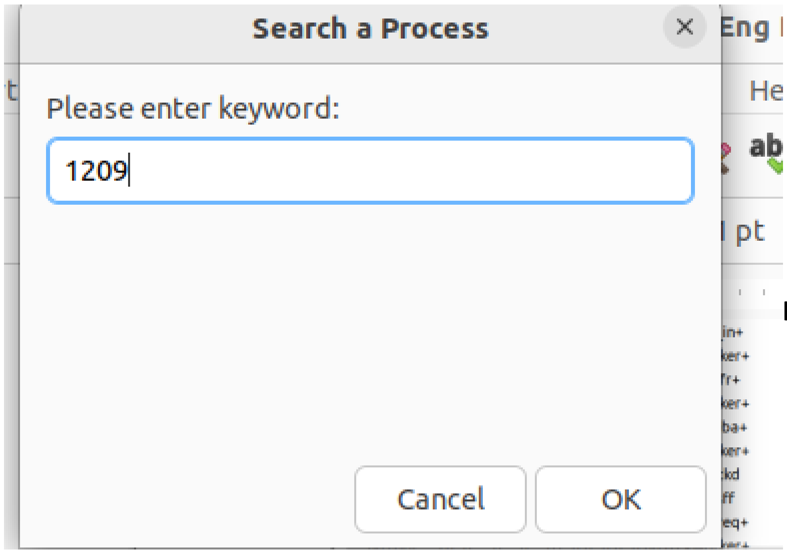
Kill Process

Please enter PID:

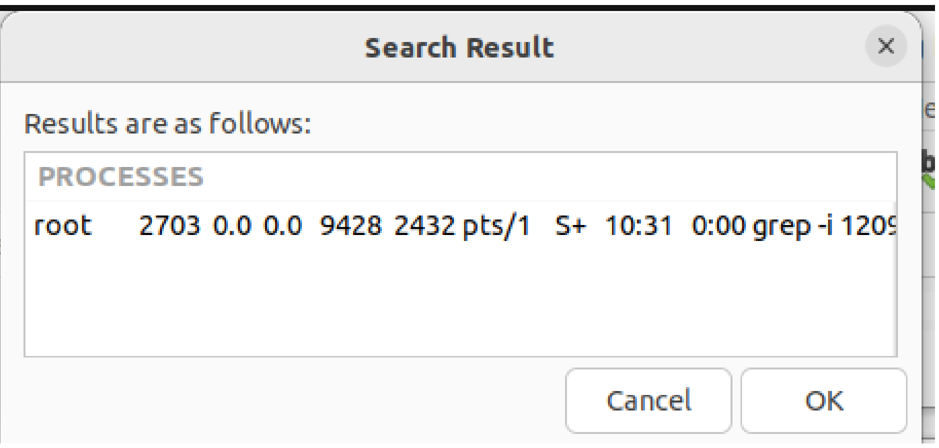
Cancel OK

Kill process

Search process:



process using cpu



```
root     2703  0.0  0.0   9428  2432 pts/1    S+   10:31   0:00 grep -i 1209
```

process using ram