**Lab File**

**OPERATING SYSTEMS**

**(CS 202)**

**DEPARTMENT OF COMPUTER SCIENCE AND**

**ENGINEERING**

Logo

Description automatically generated

Submitted to: Submitted by:

Ms. Seema Sharma Priyansh Raj Mahendra

Asstt. Professor A2305219267

CSE Department, ASET B.Tech. CSE

4CSE – 4Y

AMITY SCHOOL OF ENGINEERING AND TECHNOLOGY AMITY UNIVERSITY UTTAR PRADESH

NOIDA – 201301

**Index**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **S.NO** | **Assignment Category** | **Code** | **Experiment No.** | **Name of Experiment** | **Date of Allotment of Experiment** | **Date of Evaluation** | **Max. Marks** | **Marks Obtained** | **Signature of Faculty** |
| **1.** | **Mandatory**  **Experiment** | **LR (10)** | **1** | To explore basic Linux commands | 17/12/20 |  | 1 |  |  |
| **2.** | **2** | To explore basic Linux commands | 24/12/20 |  | 1 |  |  |
| **3.** | **3** | To explore Linux file commands | 07/01/21 |  | 1 |  |  |
| **4.** | **4** | To explore Linux file commands | 14//01/21 |  | 1 |  |  |
| **5.** | **5** | Programs based on Shell Scripting | 21/01/21 |  | 1 |  |  |
| **6.** | **6** | Programs based on Shell Scripting | 04/02/21 |  | 1 |  |  |
| **7.** | **7** | Programs based on Shell Scripting | 11/02/21 |  | 1 |  |  |
| **8.** | **8** | Program to simulate First Come First Serve CPU scheduling algorithm | 18/02/21 |  | 1 |  |  |
| **9.** | **9** | Program to simulate Shortest Job First CPU scheduling algorithm | 04/03/21 |  | 1 |  |  |
| **10.** | **10** | Program to simulate round robin scheduling algorithm | 18/03/21 |  | 1 |  |  |
| **11.** | **11** | Program to simulate banker’s algorithm | 25/03/21 |  | 1 |  |  |
| **12.** | **12** | Program to simulate SSTF disk scheduling | 01/04/21 |  | 1 |  |  |
|  | Viva |  | **Viva** |  |  |  |  |  |  |

**Experiment-01**

Date: 17/12/2020

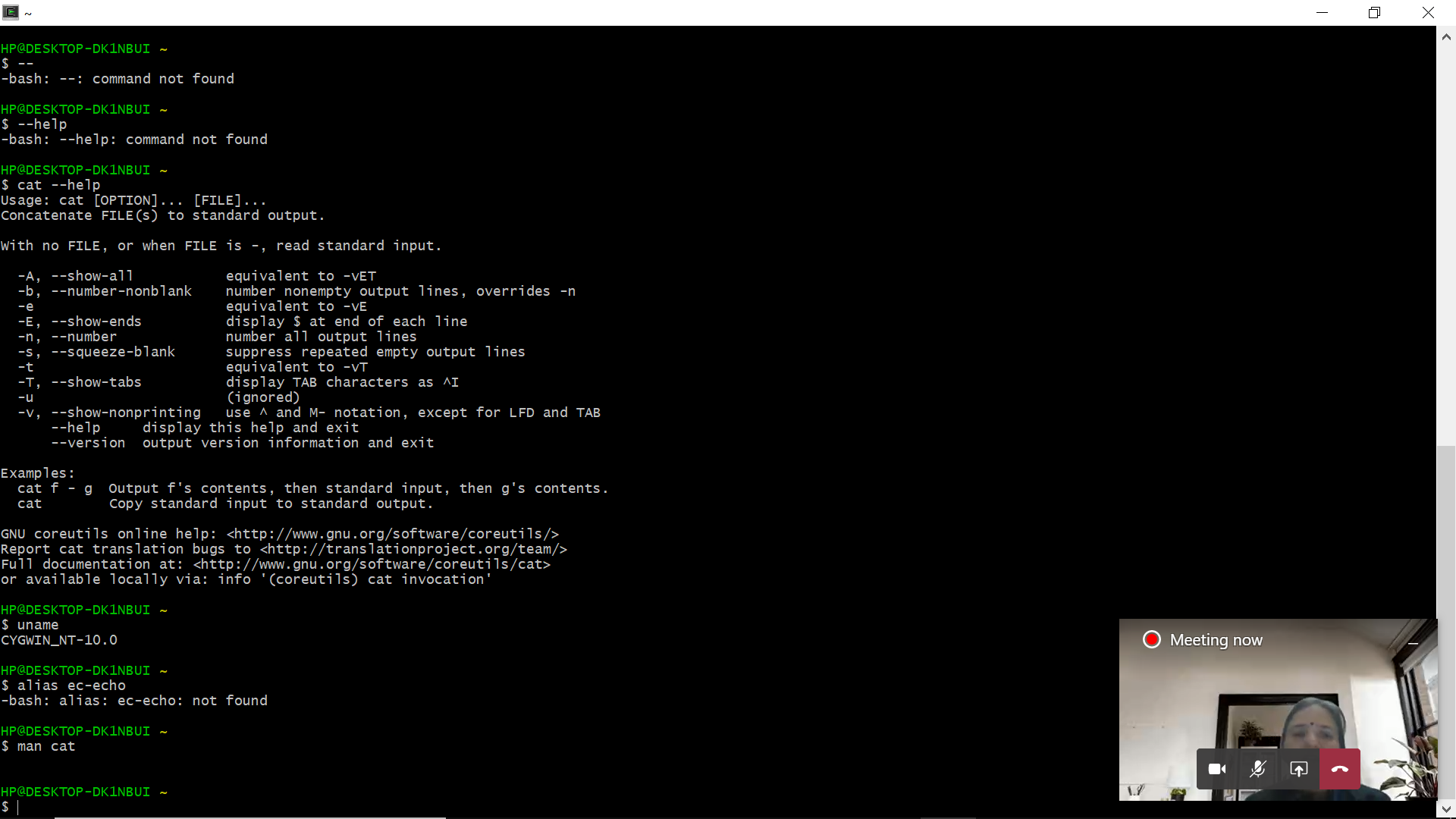
Objective: To execute basic Linux commands on terminal.

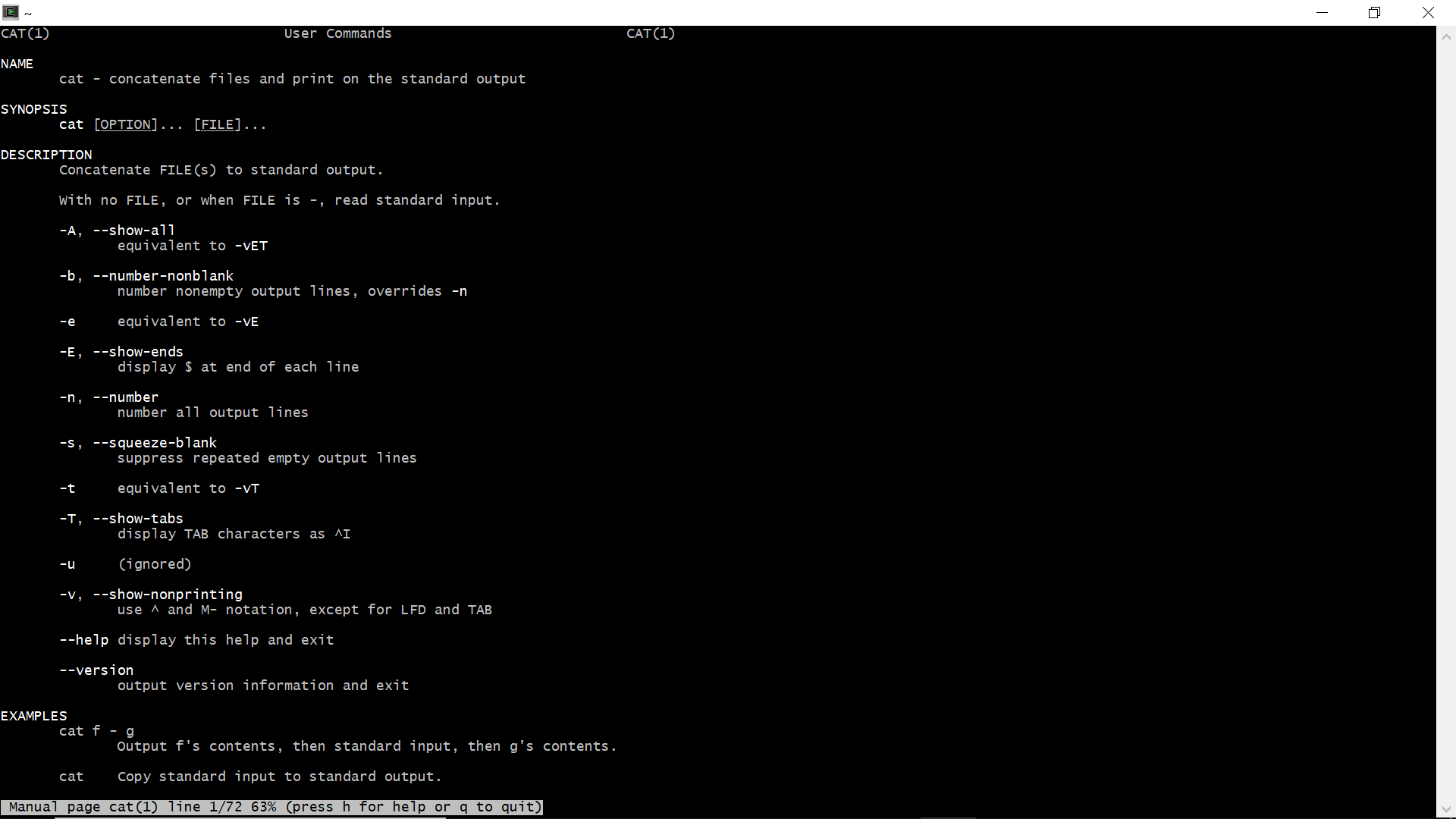
Software used: Cygwin

OS: Windows 10

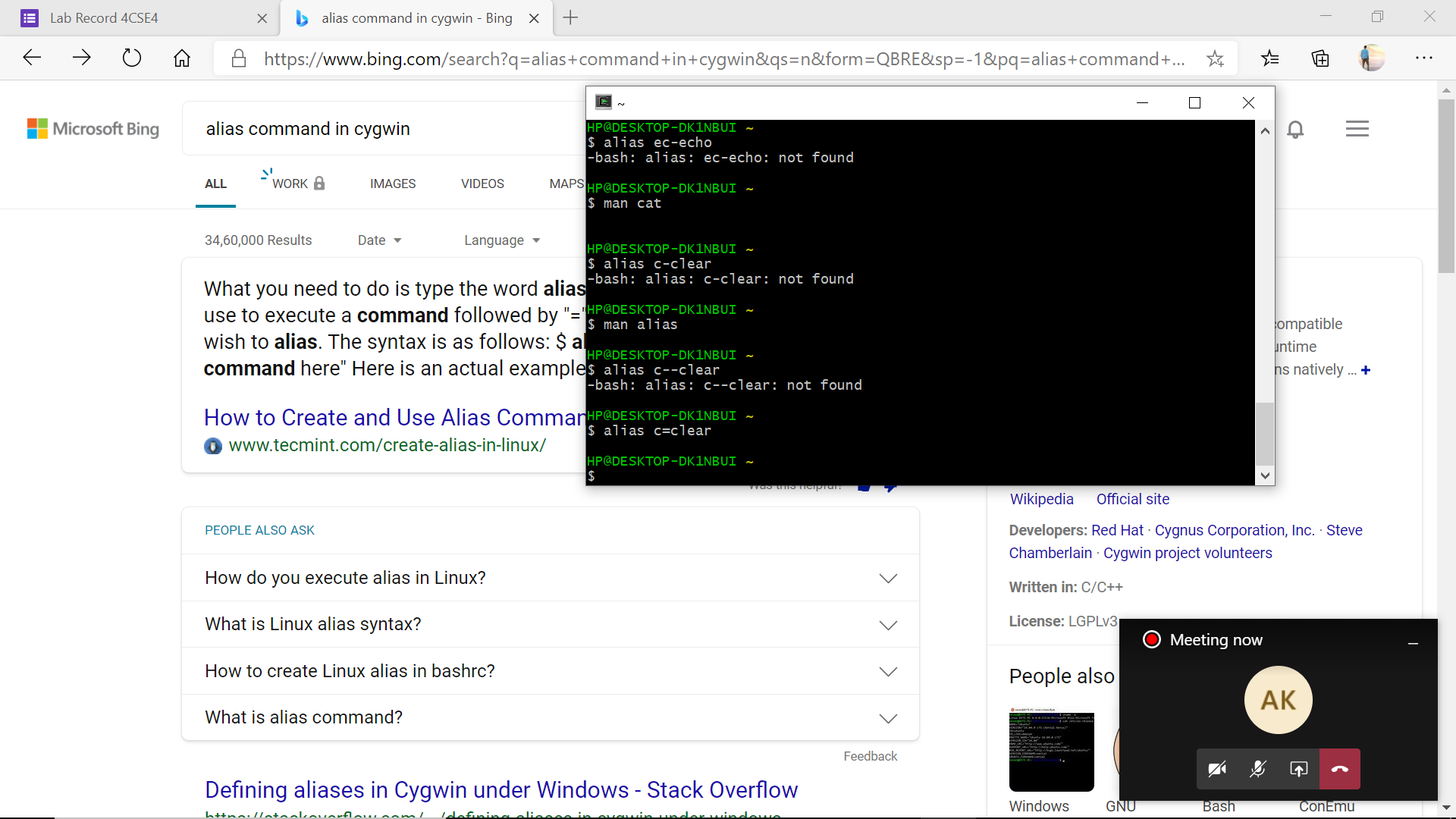
Theory:

1. man: man command in is used to **display the user manual** of any command that we can run on the terminal.

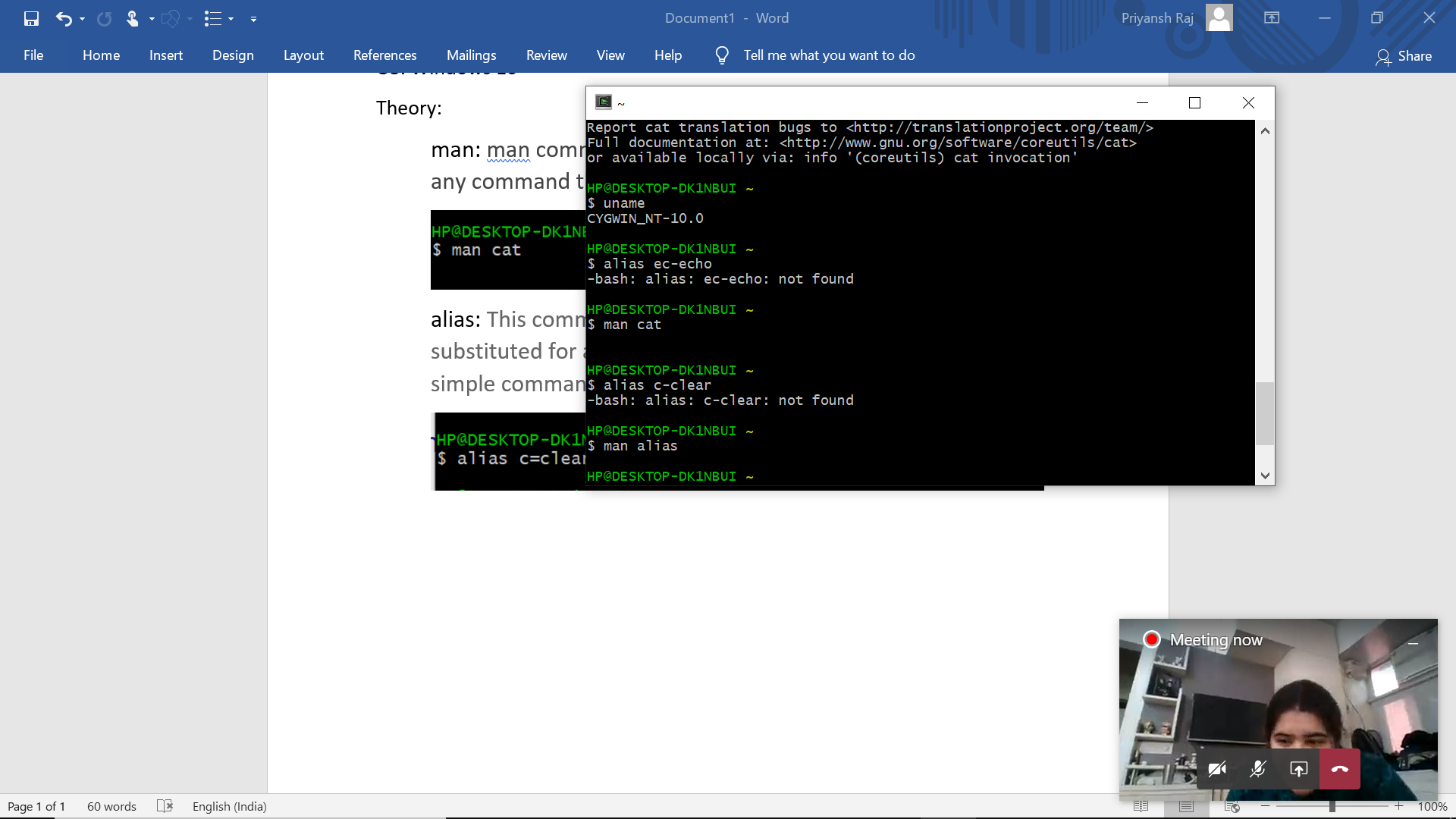




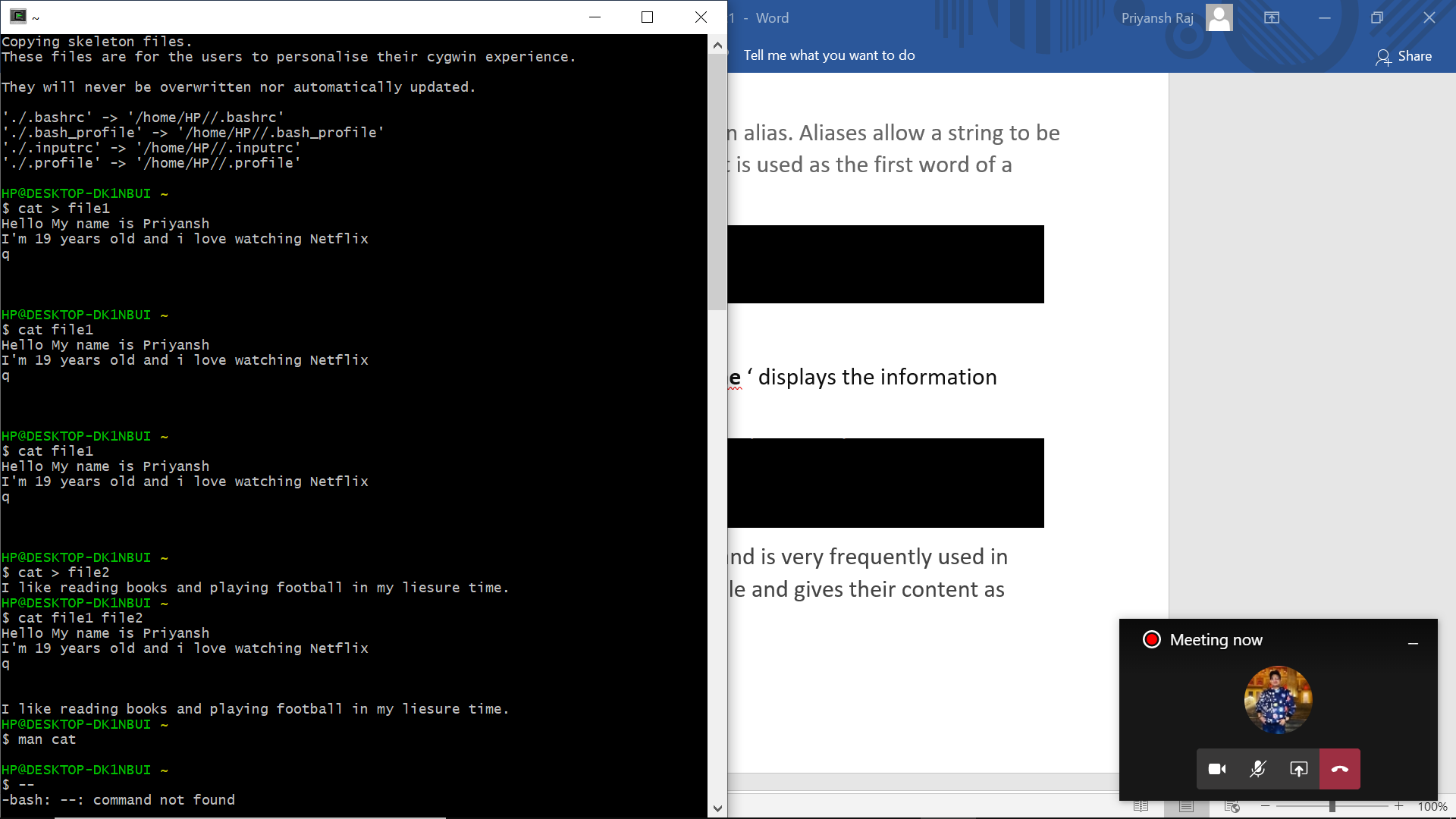
1. alias: This command creates an alias. Aliases allow a string to be substituted for a word when it is used as the first word of a simple command.



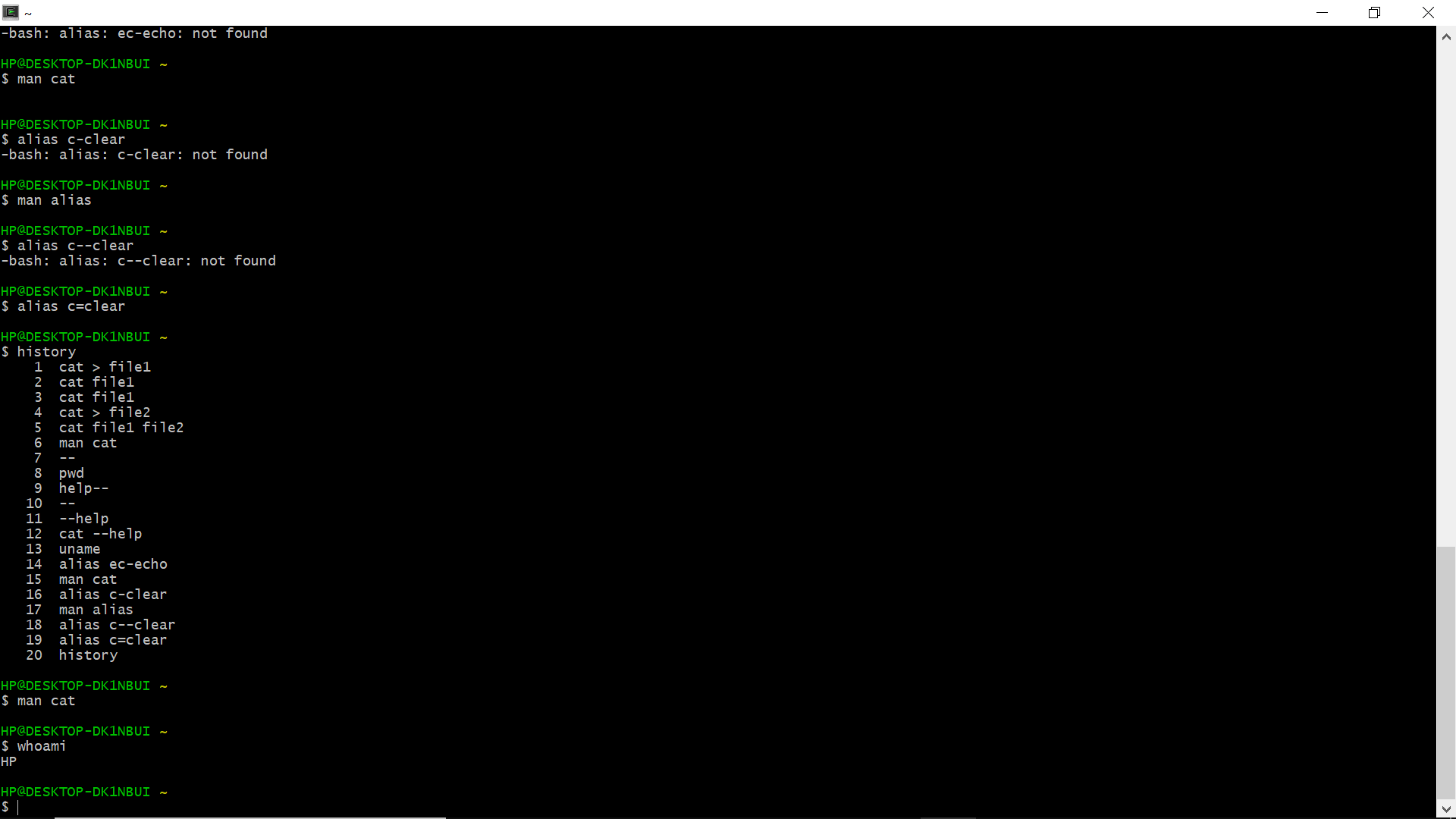
1. uname: The **command** ‘ **uname** ‘ displays the information about the system.



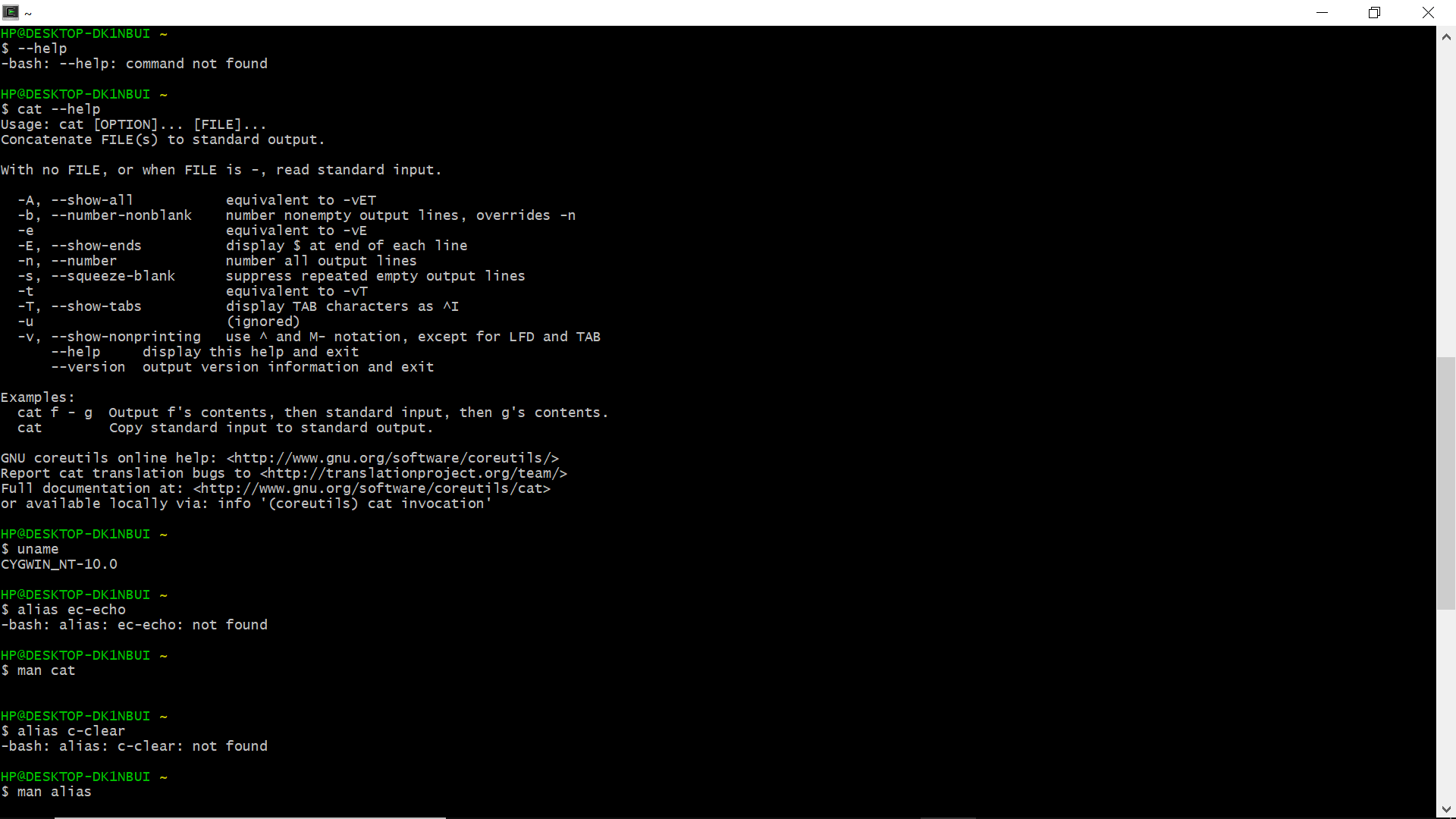
1. Cat: Cat (concatenate) command is very frequently used in Linux. It reads data from the file and gives their content as output. It helps us to create, view, concatenate files. So let us see some frequently used cat commands.



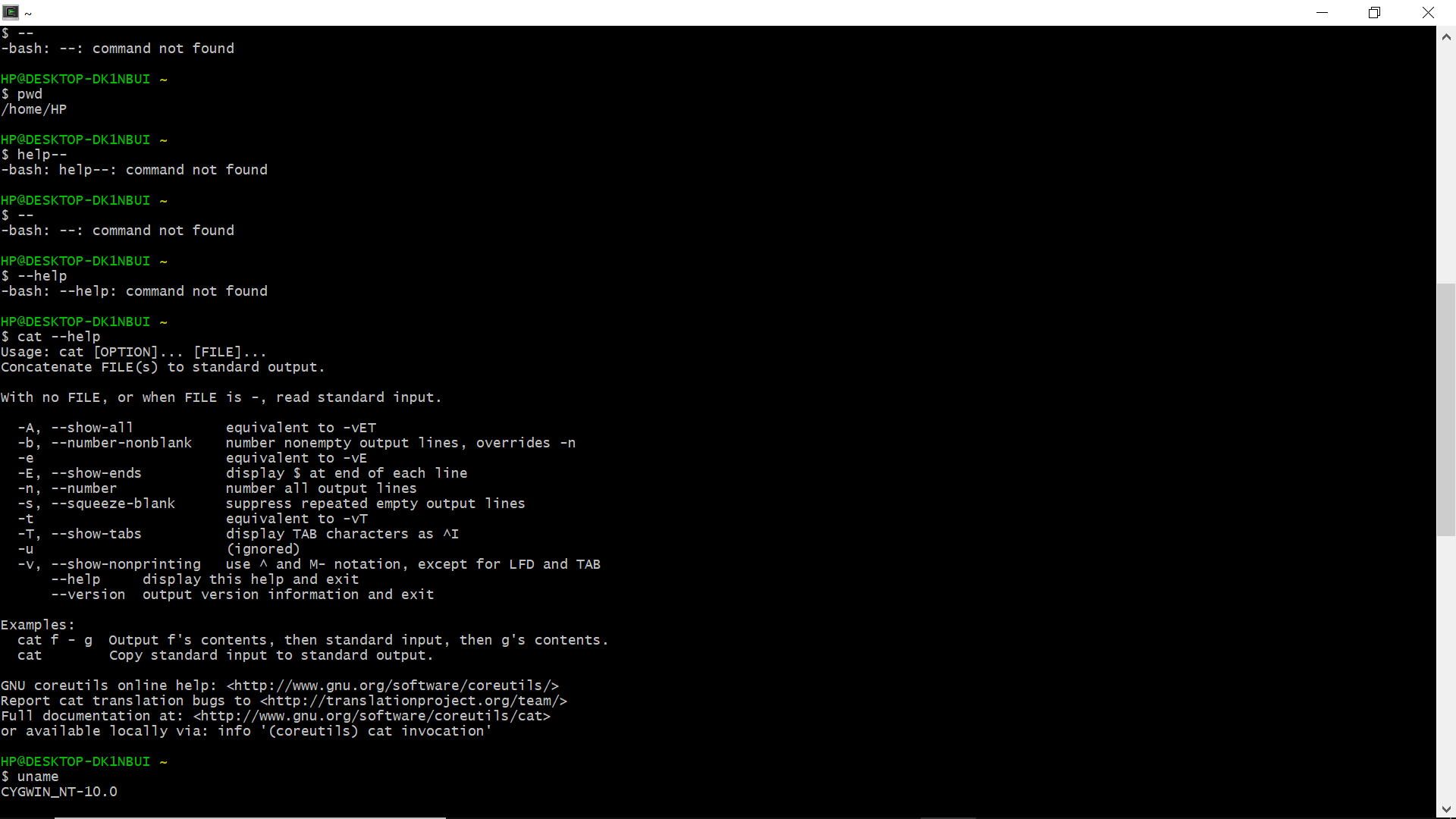
1. Whoami: The who command will always display the account that you used to login (the real user info). The whoami command will show your effective user.



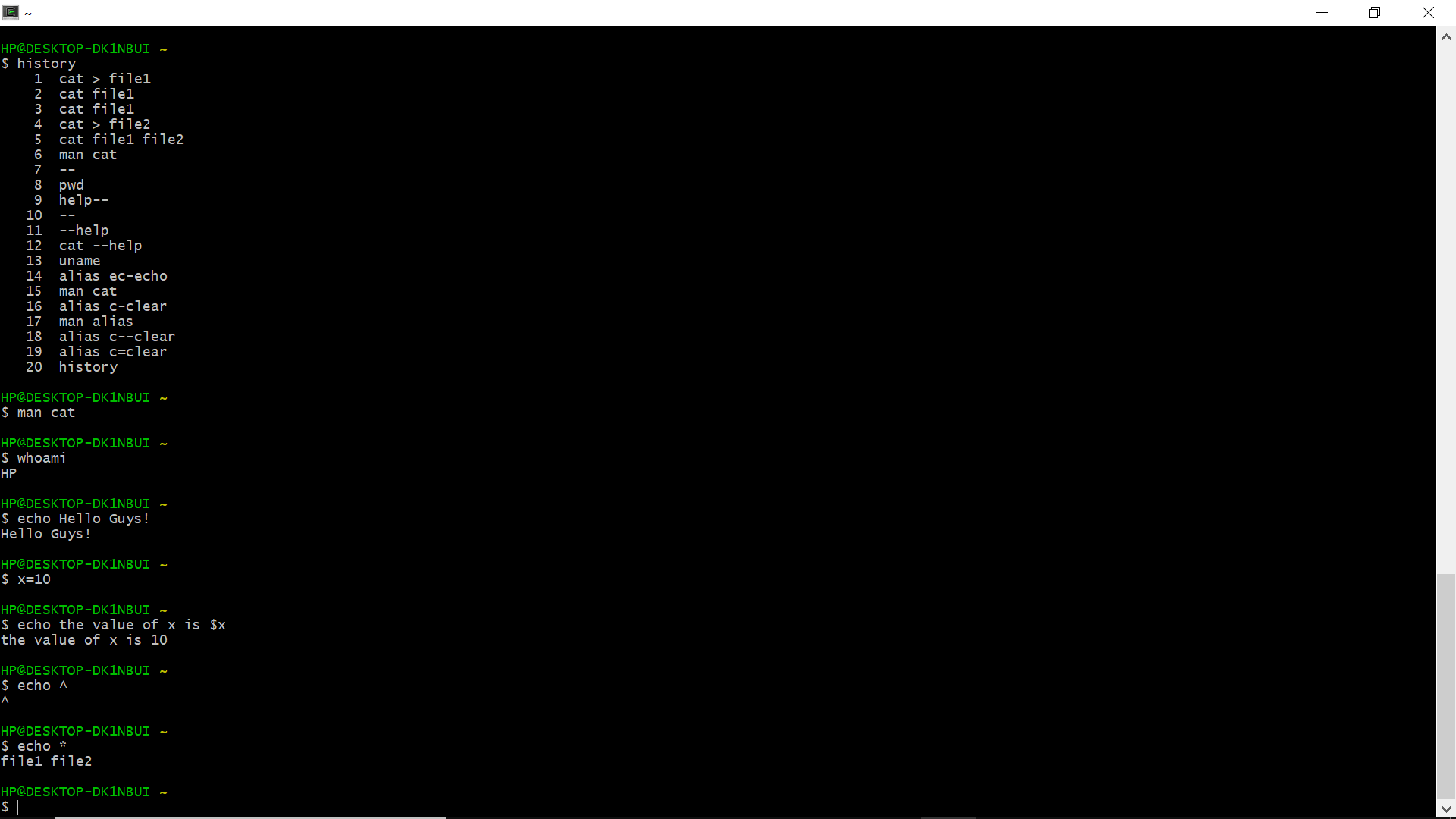
1. Help: cat - -help



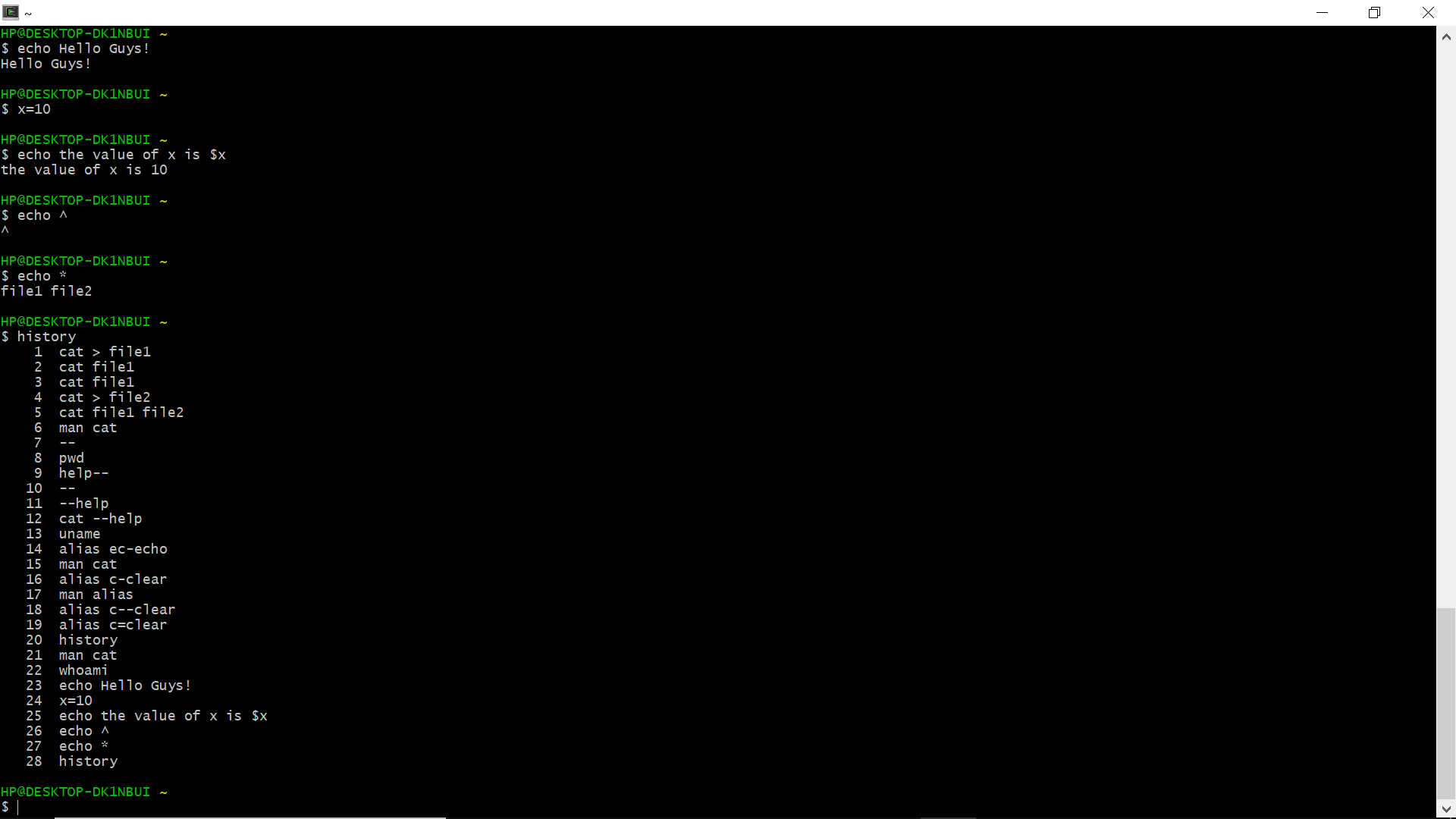
1. pwd: (Print word directory) pwd command writes the full pathname of the current working directory to the standard output



1. echo: echo command in Linux is one of the widely used command in day-to-day operations task. The echo command is a built-in command-line tool that prints the text or string to the standard output or redirect output to a file. The command is usually used in a bash shell or other shells to print the output from a command.



1. history: The list of previously used commands is then written to the terminal window. The commands are numbered, with the most recently used (those with the highest numbers) at the end of the list.



Result: Executed basic Linux commands successfully.

**Experiment 2**

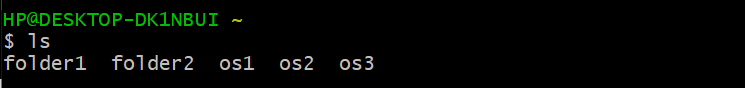
**Date:** 24 December, 2020

**Objective:** To explore some more Linux ls commands

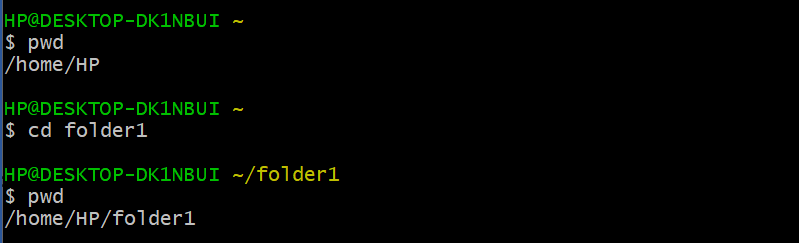
**Software used:** Cygwin Terminal

**Theory: -**

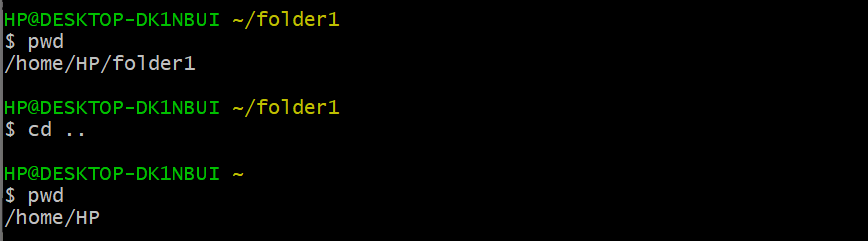
1. ls – To list the files in the current directory



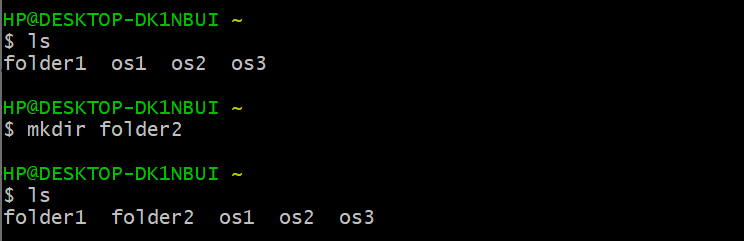
1. cd – To change to a specific directory



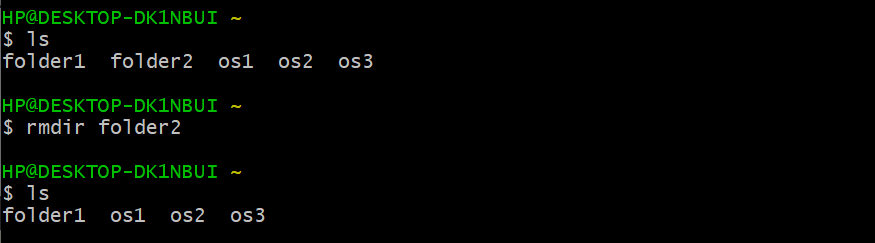
1. cd .. – To come out of a directory



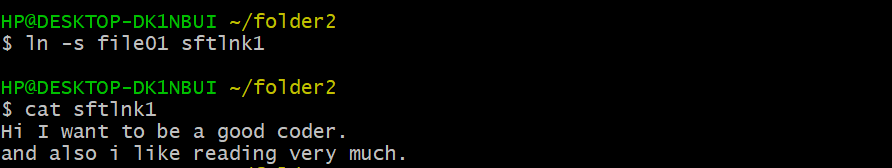
1. mkdir – To create a new directory

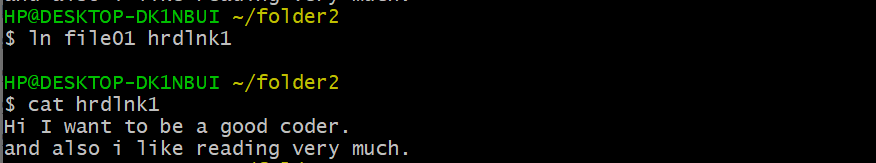


1. rmdir – To remove an empty directory



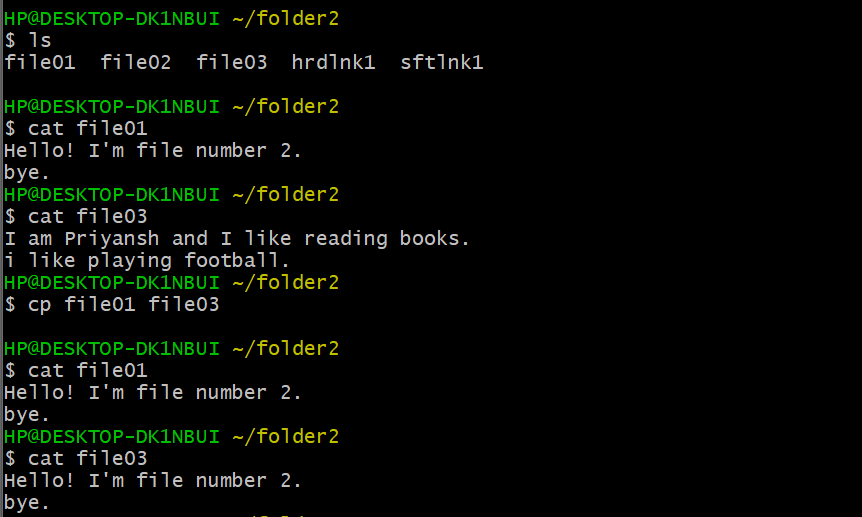
1. ln – To make links ([ln -s] for soft link and [ln] for hard link)



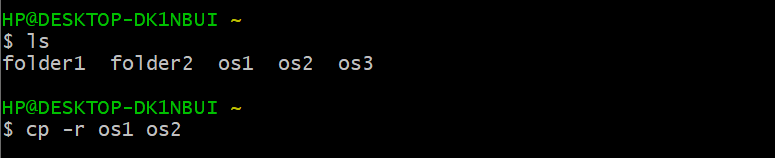


1. cp – To copy a file or directory to another file or directory

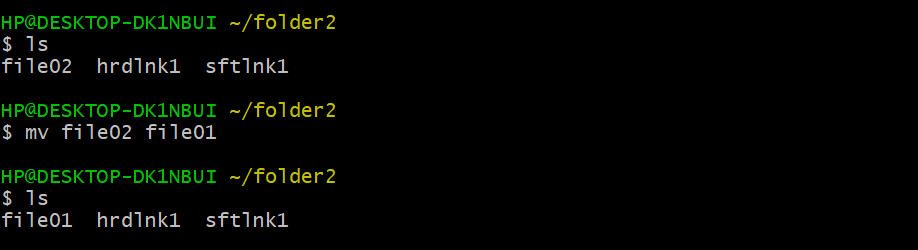
Copying files:



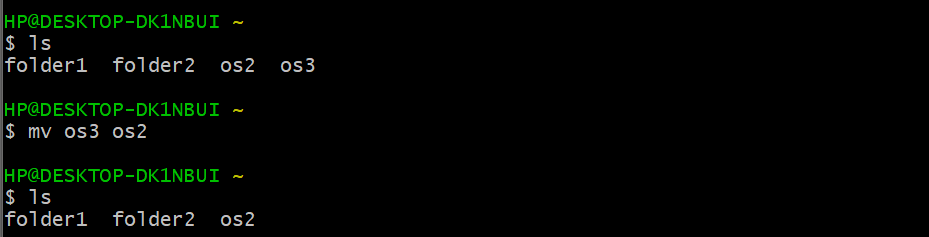
Copying directories:



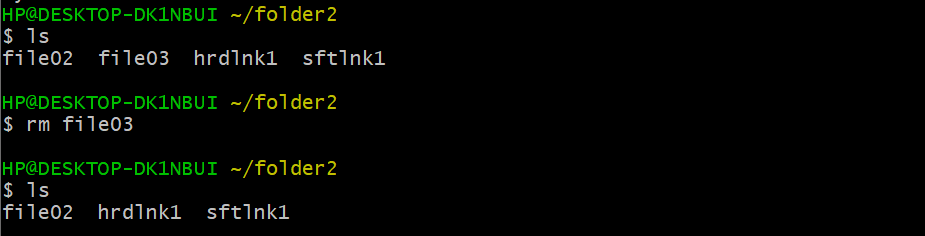
1. mv – To move or rename a file



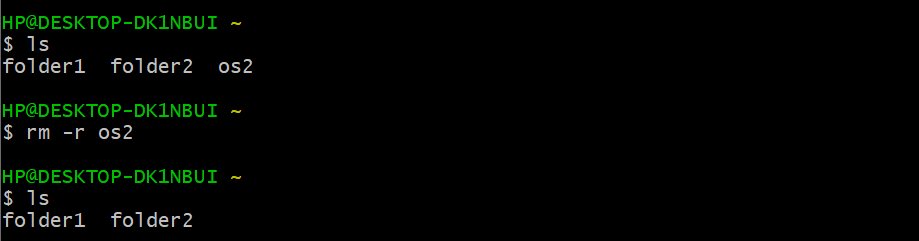
Moving directories:



1. rm – To remove a file or directory



Removing directories:



**Result:** Executed Linux file commands successfully.

**Experiment-3**

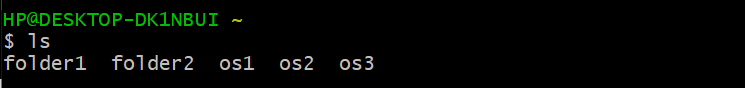
**Date:** 07 January, 2021

**Objective:** To explore Vi editor Linux commands

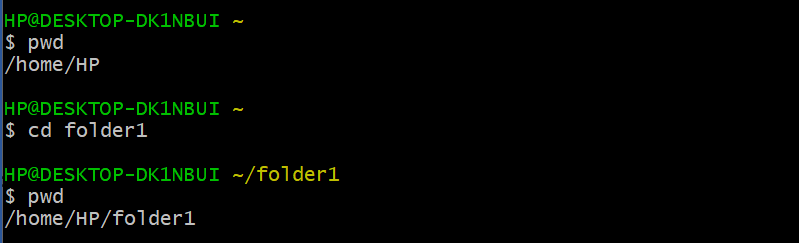
**Software used:** Cygwin Terminal

**Theory: -**

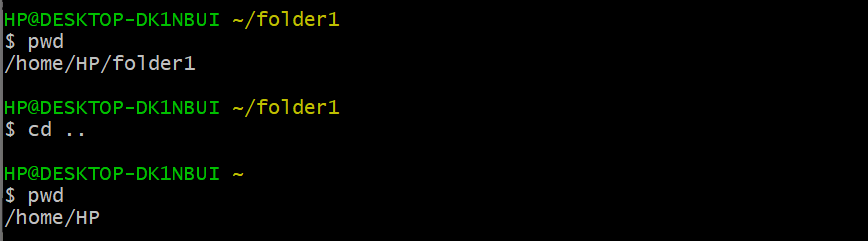
1. ls – To list the files in the current directory



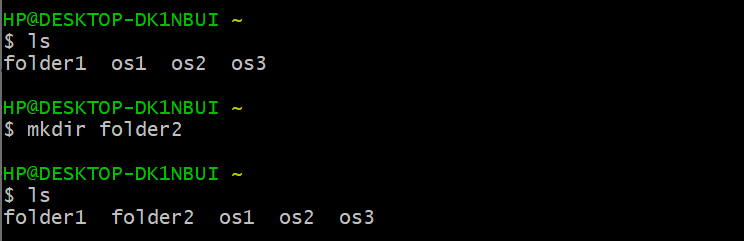
1. cd – To change to a specific directory



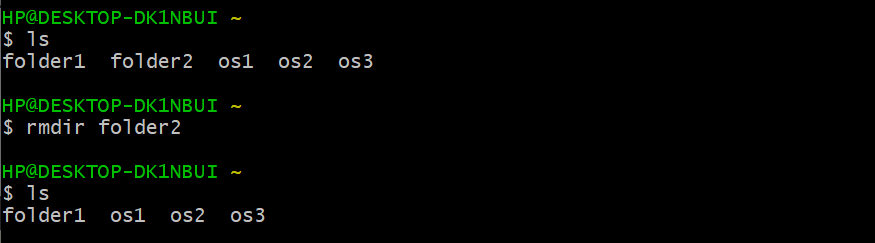
1. cd .. – To come out of a directory



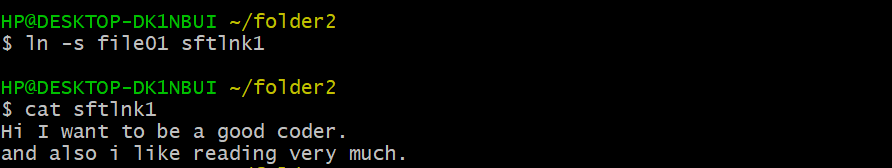
1. mkdir – To create a new directory

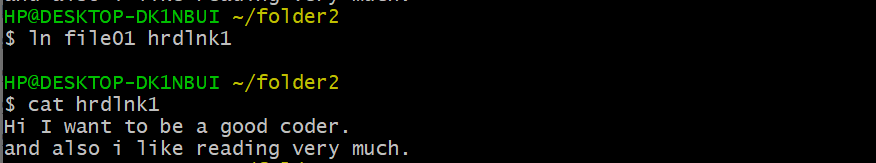


1. rmdir – To remove an empty directory



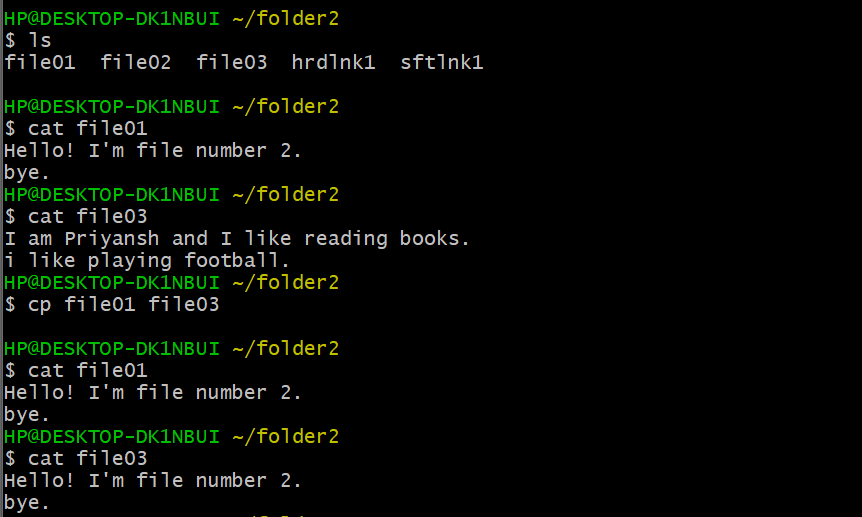
1. ln – To make links ([ln -s] for soft link and [ln] for hard link)



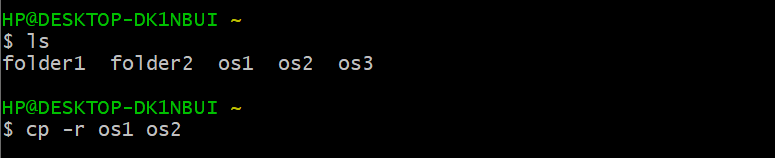


1. cp – To copy a file or directory to another file or directory

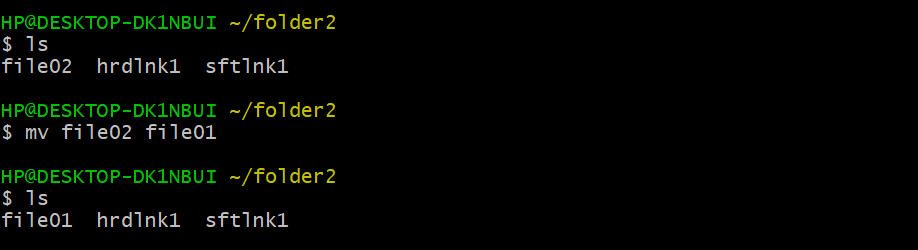
Copying files:



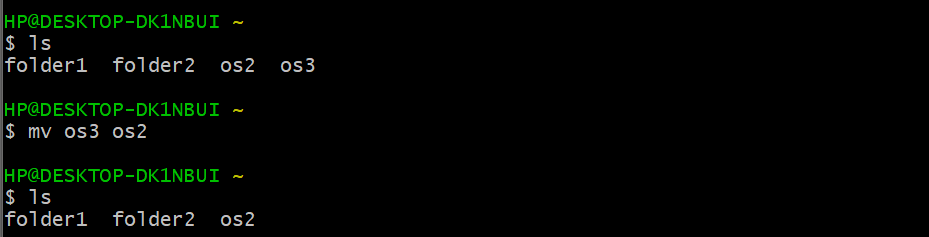
Copying directories:



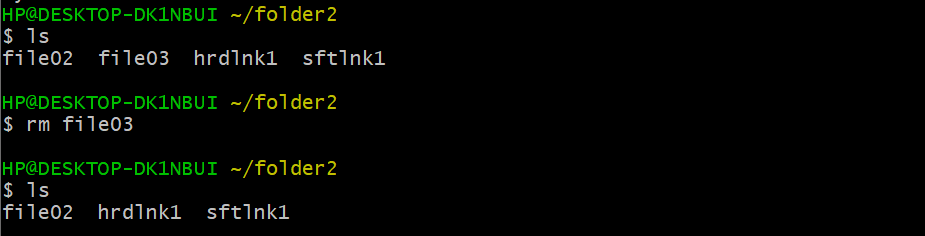
1. mv – To move or rename a file



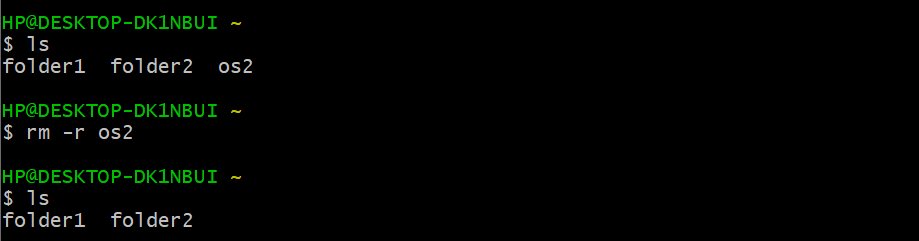
Moving directories:



1. rm – To remove a file or directory



Removing directories:



**Result:** Executed Linux file commands successfully.

**Experiment-04**

Date: 14/01/2021

Objective: To execute Linux commands on terminal.

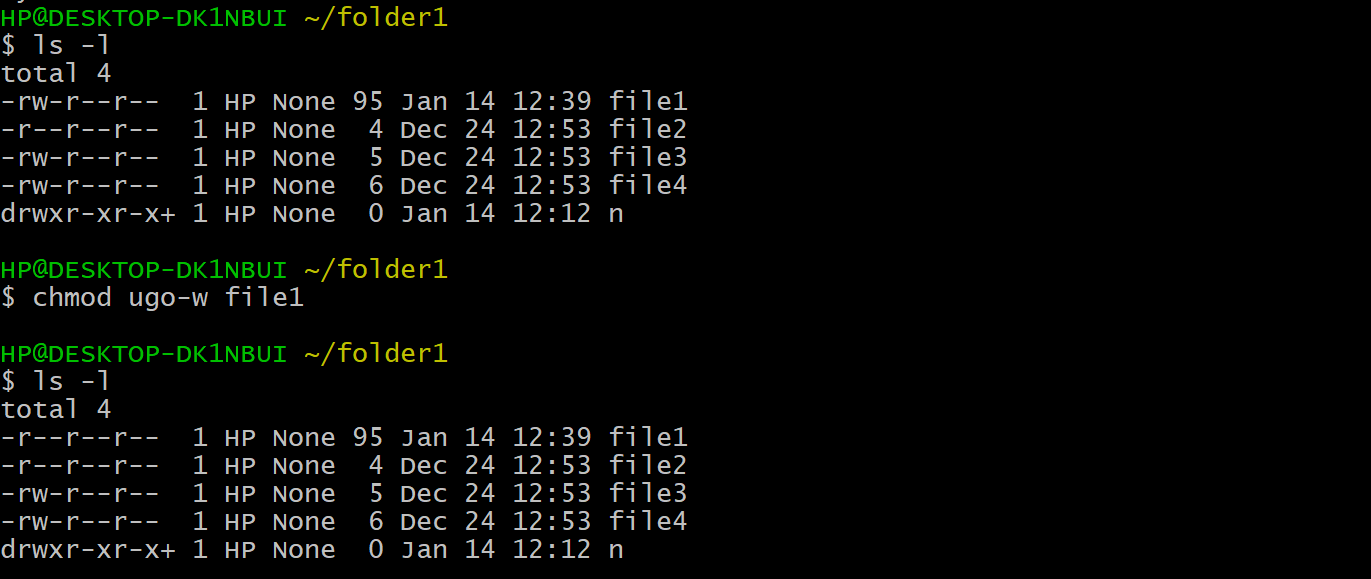
Software used: Cygwin

OS: Windows 10

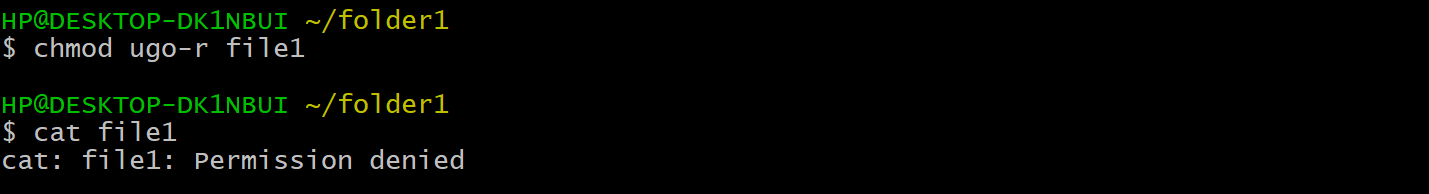
Theory:

1. **chmod**: it is the command and system call which is used to change the access permissions of file system objects.

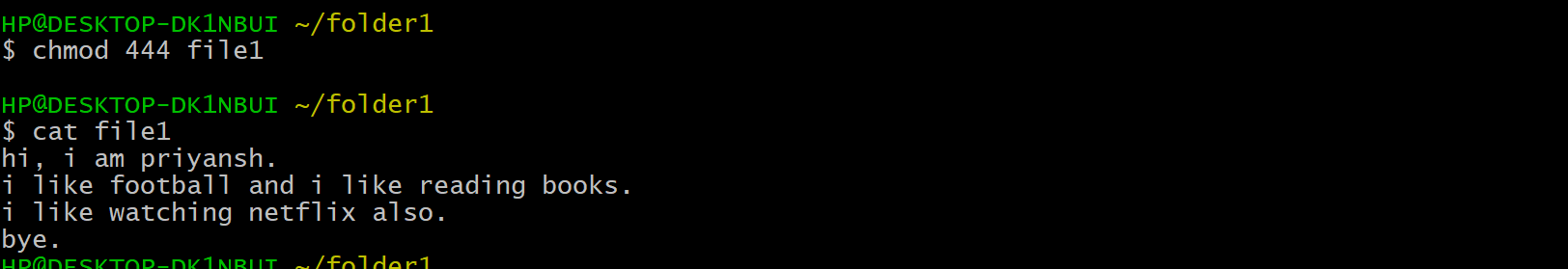
WRITE

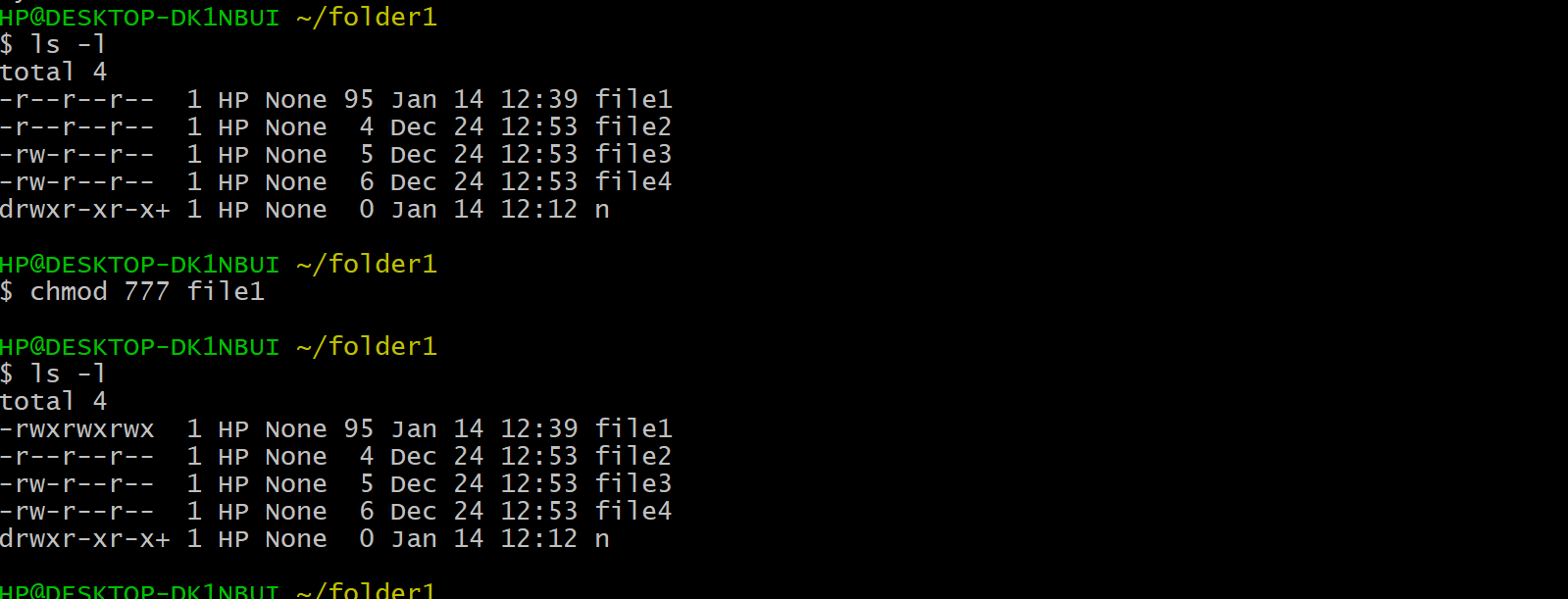


READ

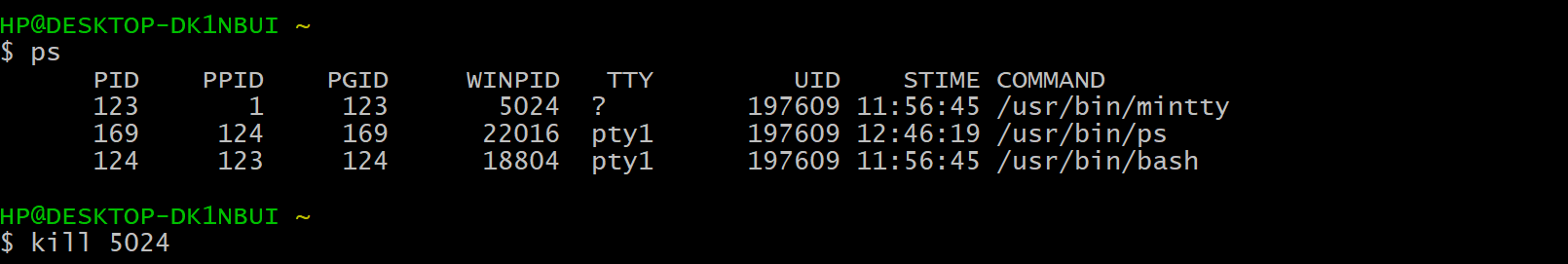


NUMERIC

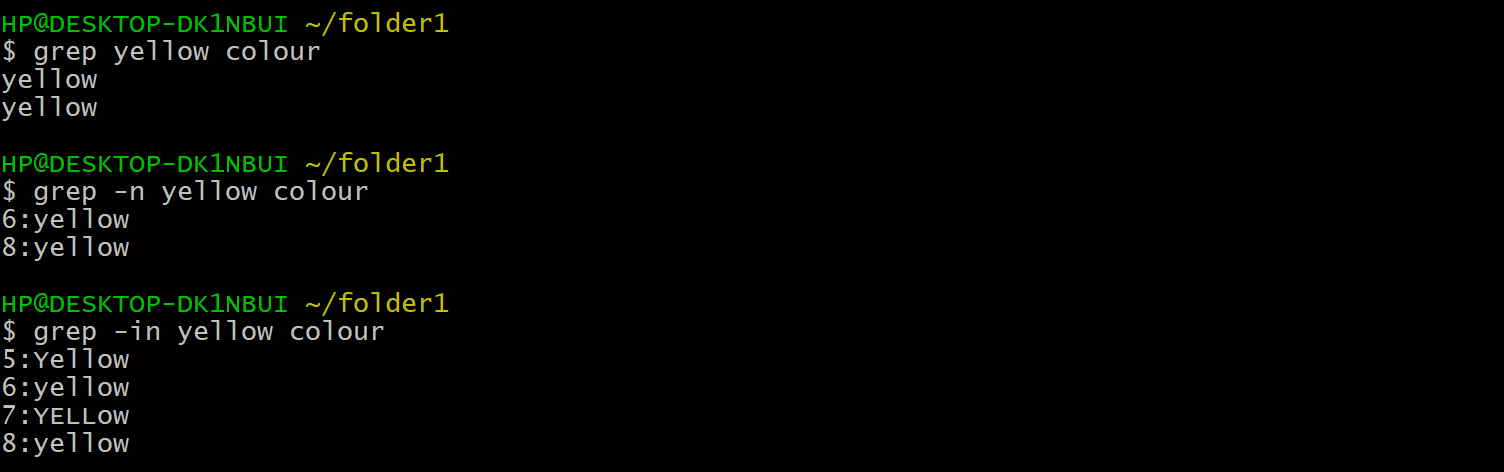


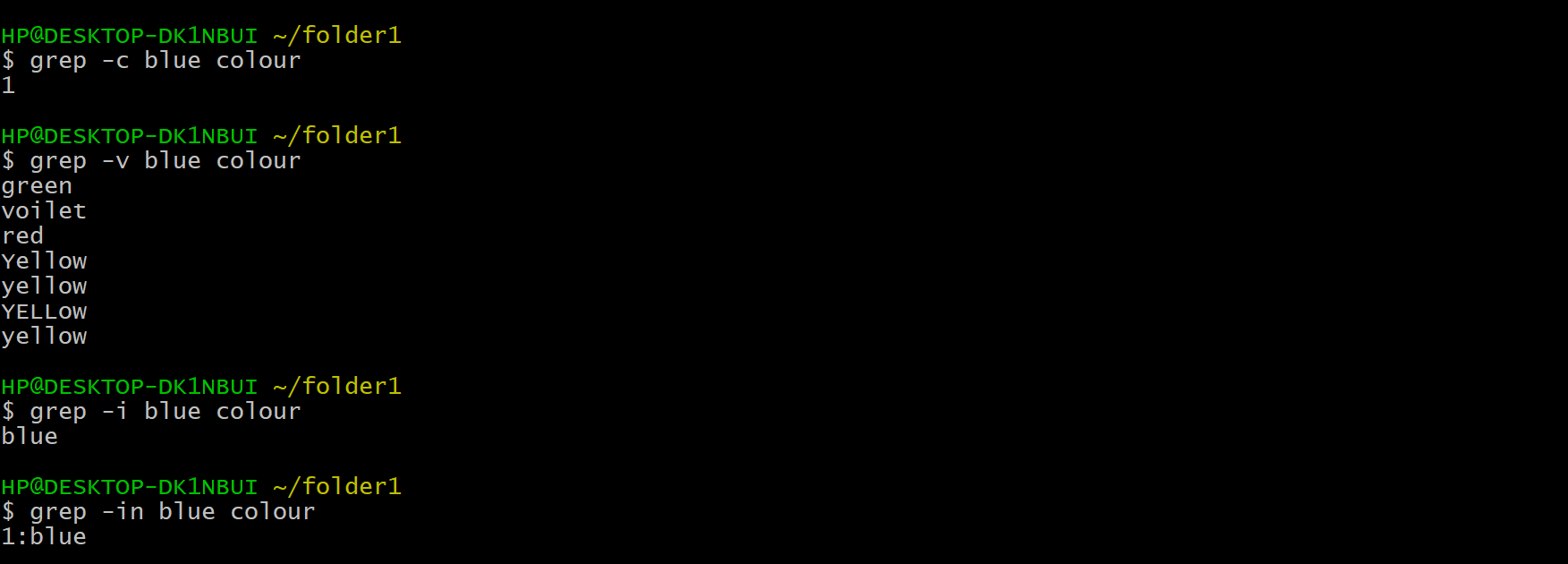


1. **ps** and **kill** command: the ps command displays the currently running processes and kill command sends a signal to a process which terminates the process.

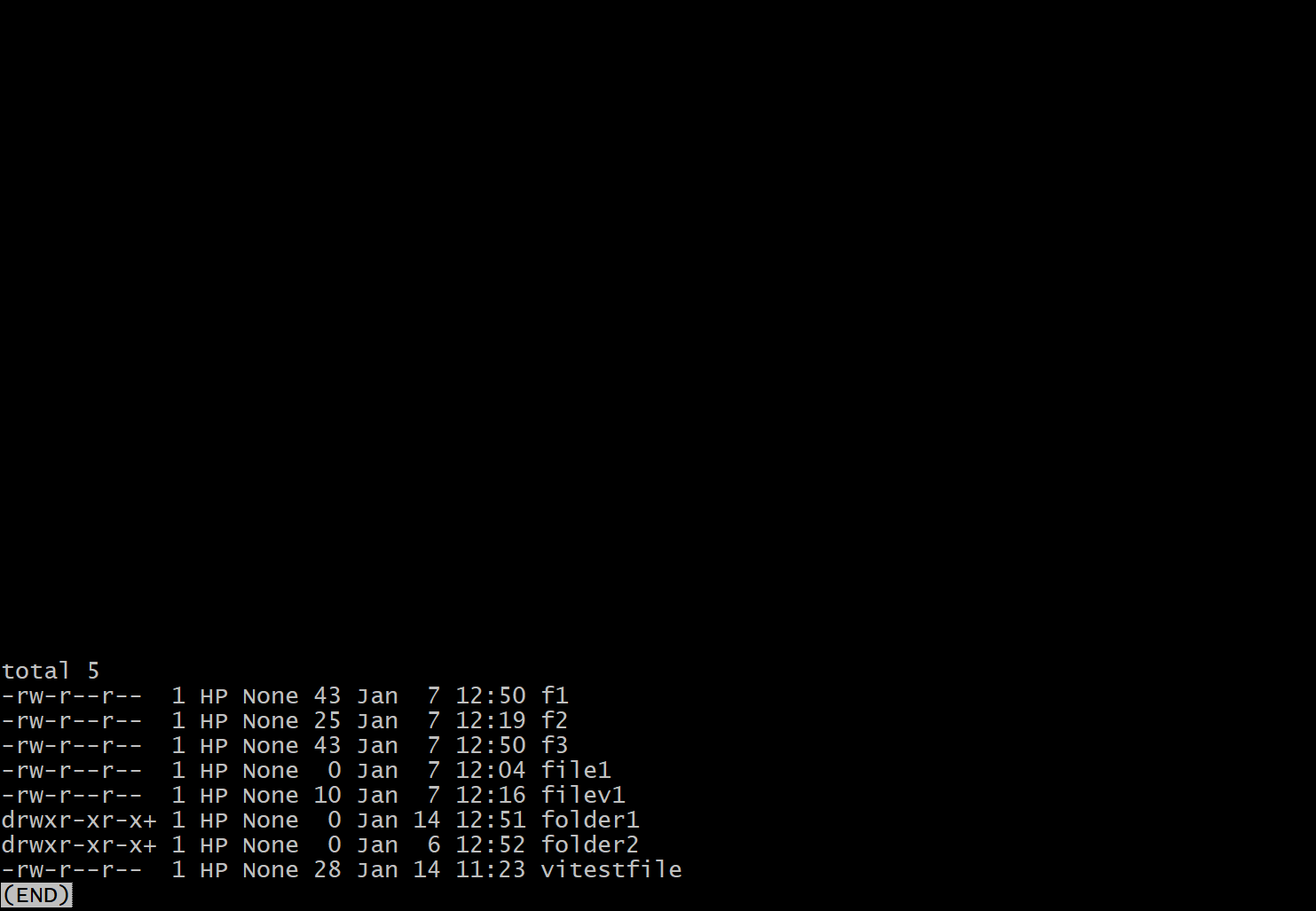
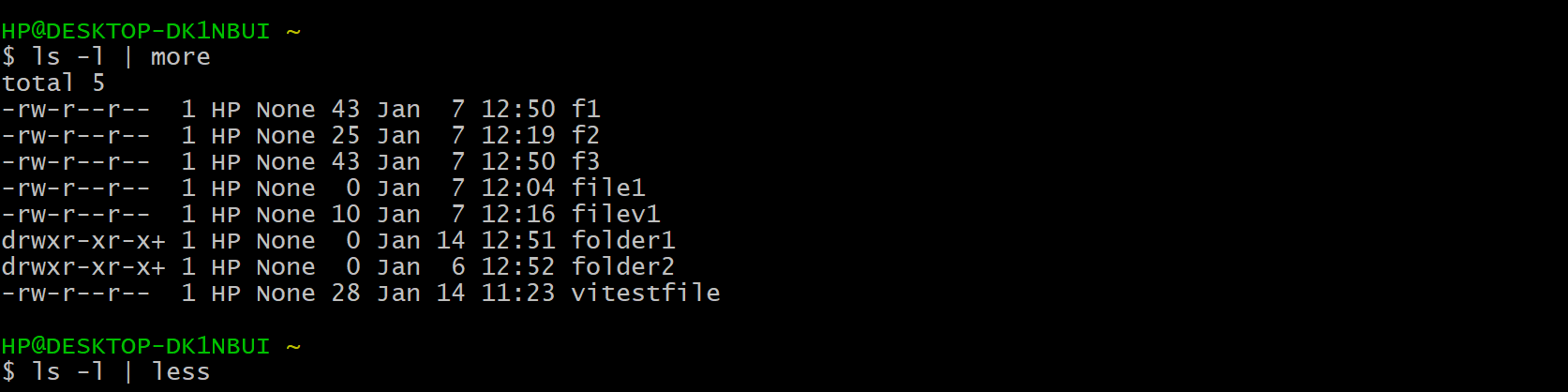


1. **grep** command: **grep searches the named input FILEs (or standard input if no files are named, or if a single hyphen-minus (-) is given as file name) for lines containing a match to the given PATTERN.**By default, grep prints the matching lines.





1. **pipes**: The Pipe is a command in Linux that lets you use two or more commands such that output of one command serves as input to the next. In short, the output of each process directly as input to the next one like a pipeline.



Result: Commands executed successfully.

**Experiment-05**

Date: 21/01/2021

Objective: To execute Linux Scripts on terminal.

Software used: Cygwin

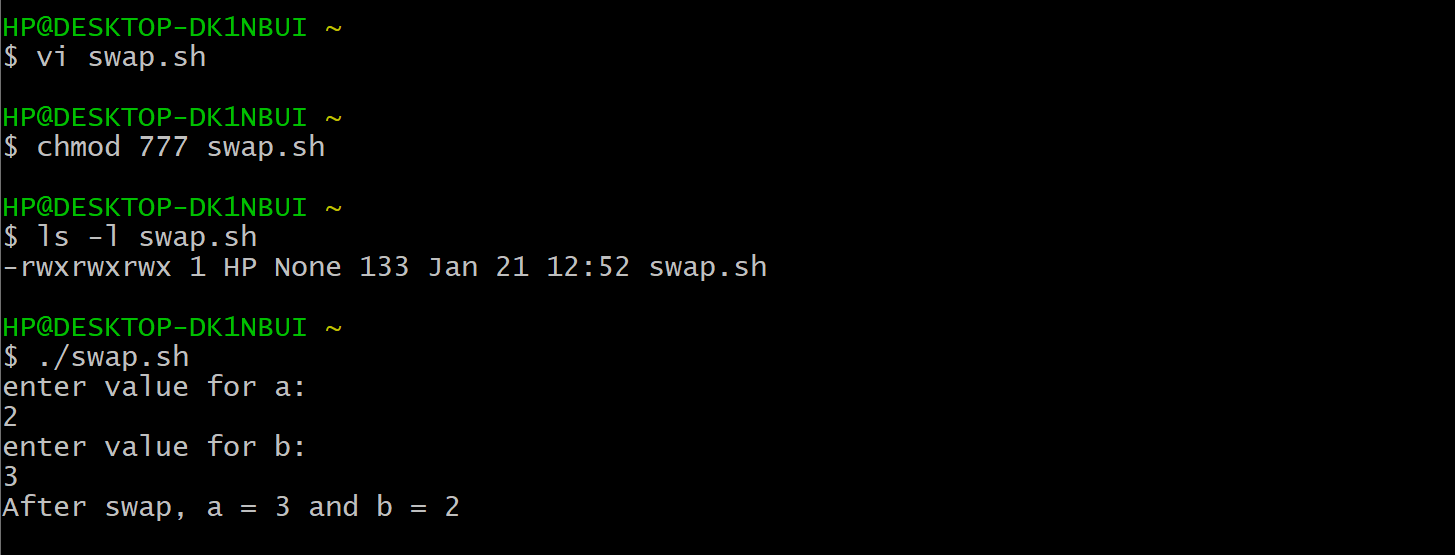
OS: Windows 10

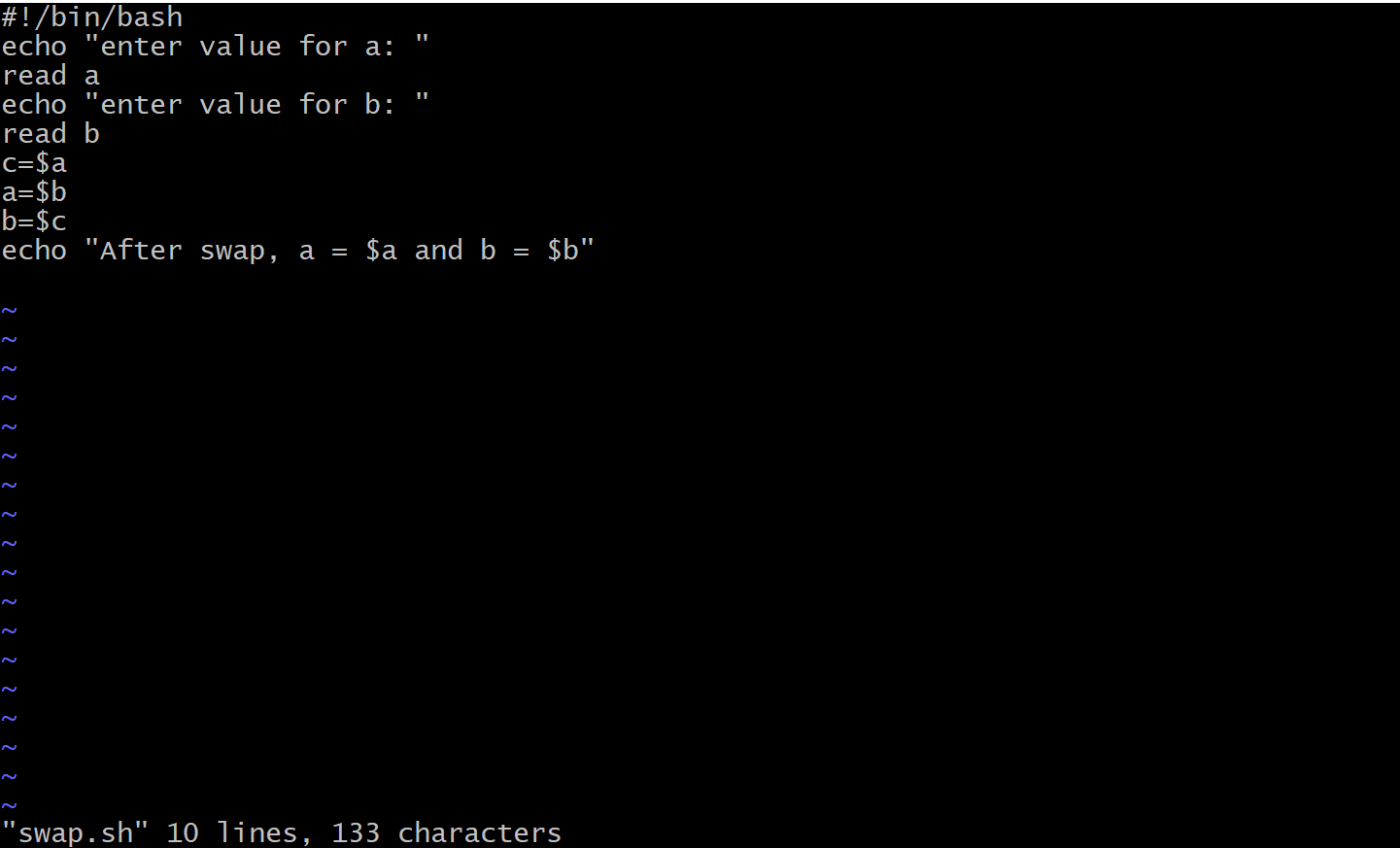
**Shell :** A shell is the user interface to the operating system

**Shell script :** A shell script is a regular text file that contains shell or commands. Before running it must have execute permission.

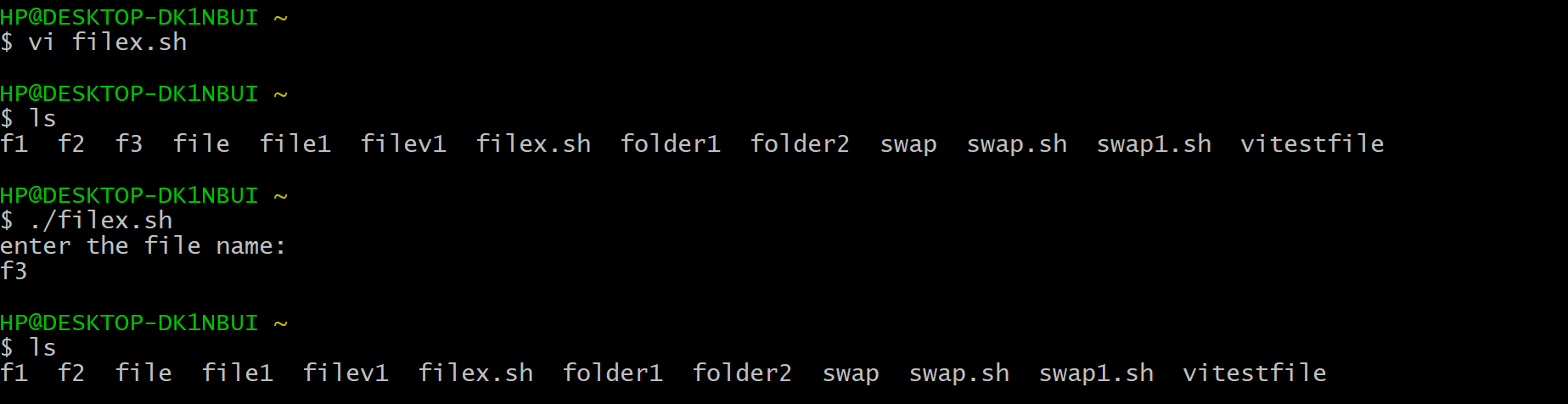
Theory:

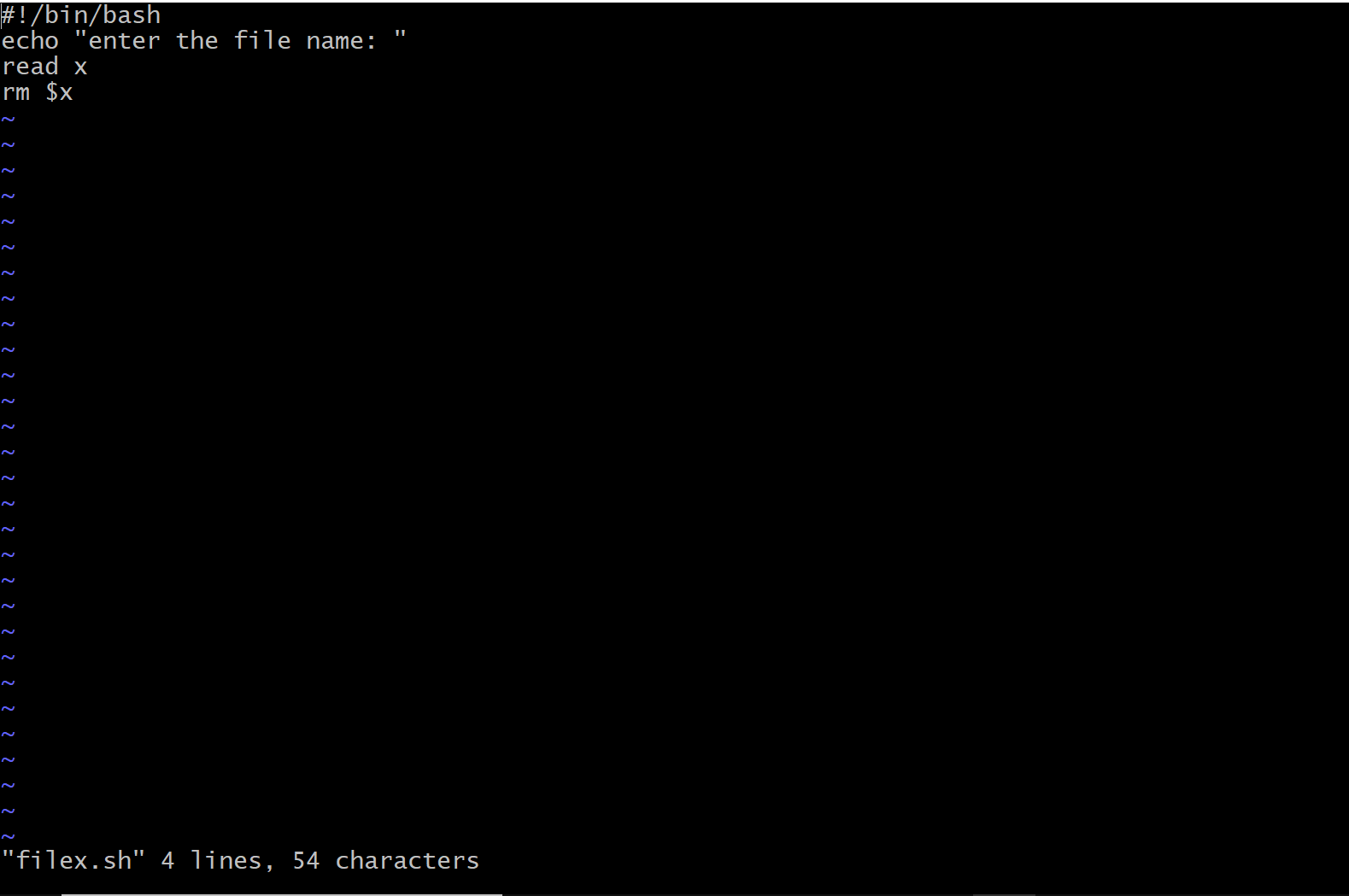
1. **Write a script to swap two numbers**



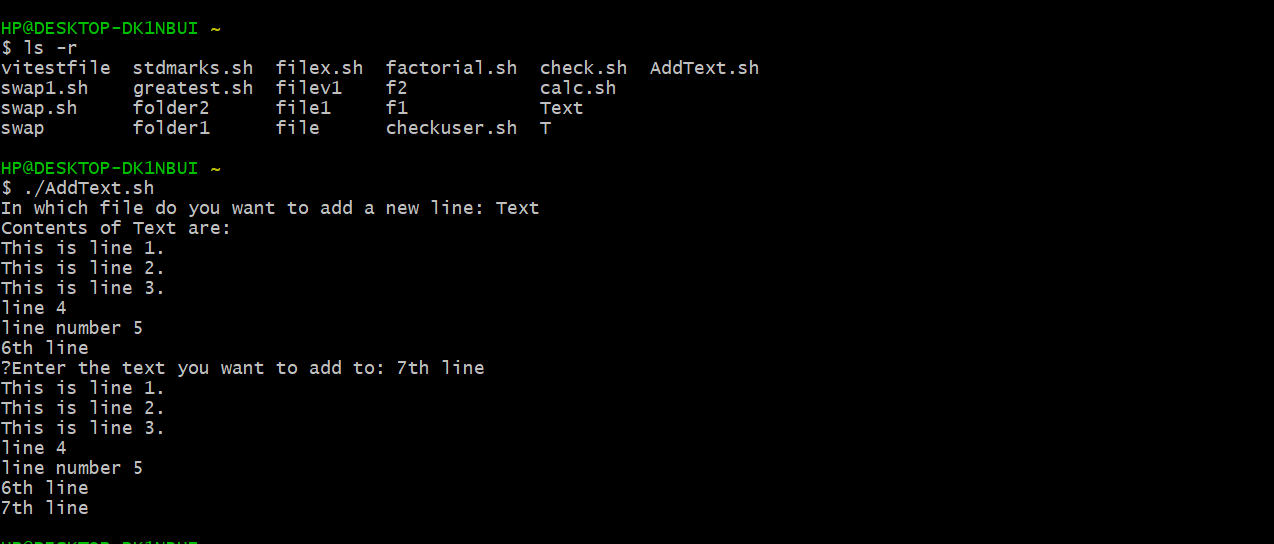


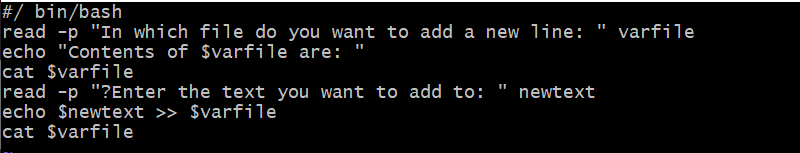
1. **Write a script to delete a file**

****

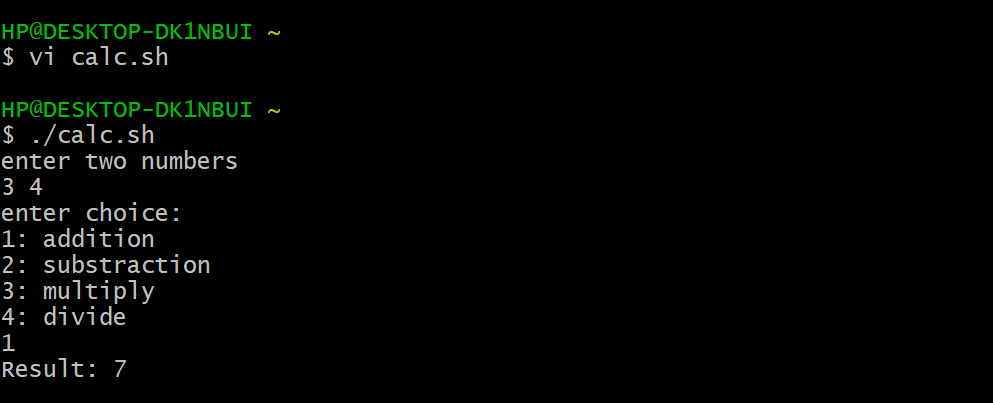
****

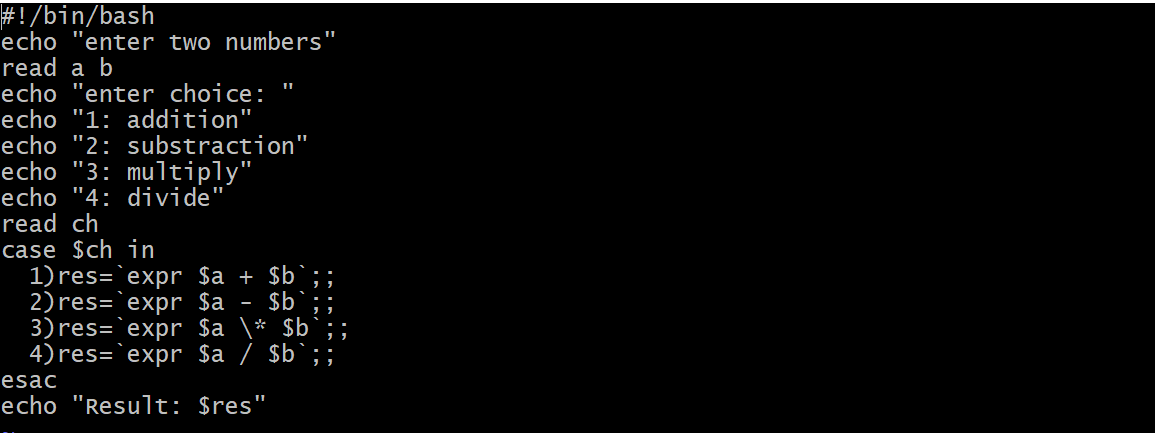
1. **Write a script to add text in an already existing file**

****



1. **Write a script to show different arithmetic operations**

****



Result: Scripts executed successfully.

**Experiment-06**

Date: 4/02/2021

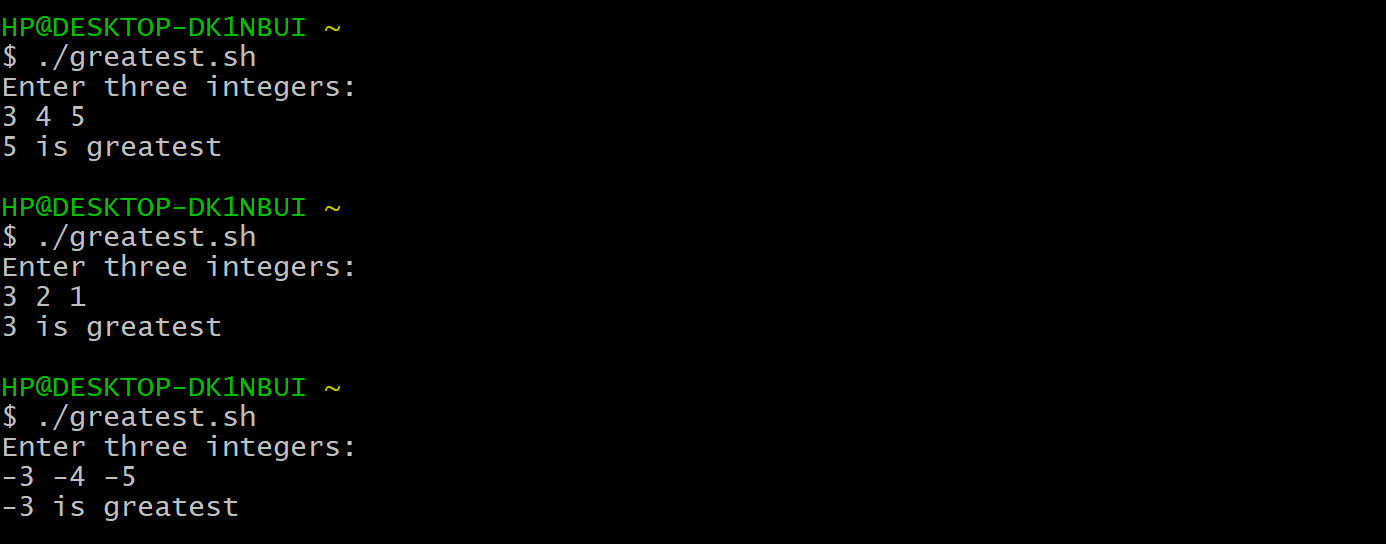
Objective: To execute Linux Scripts on terminal.

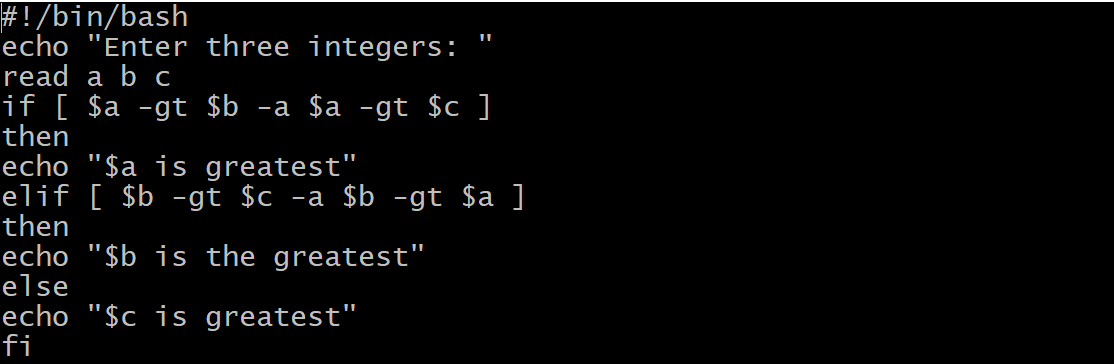
Software used: Cygwin

OS: Windows 10

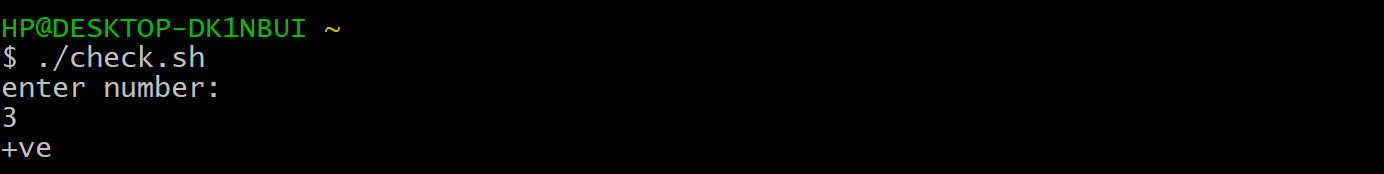
Theory:

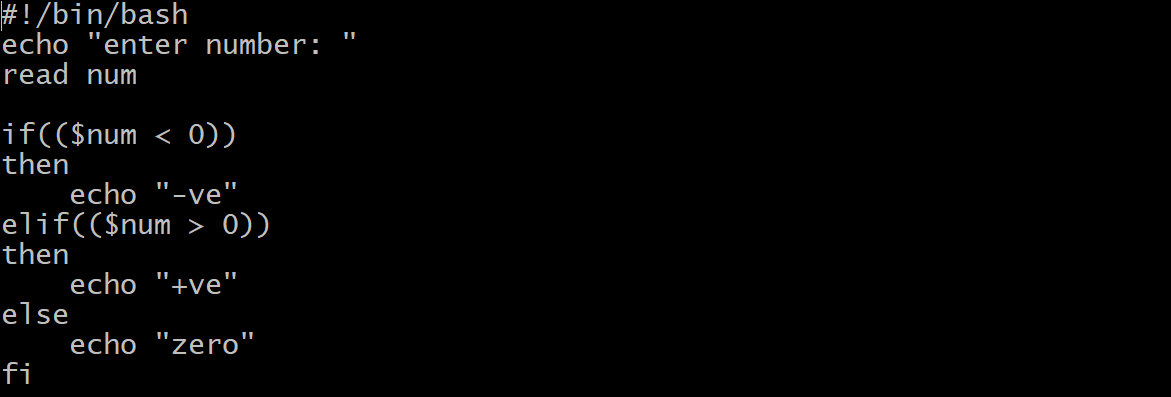
1. **Greatest num of three**



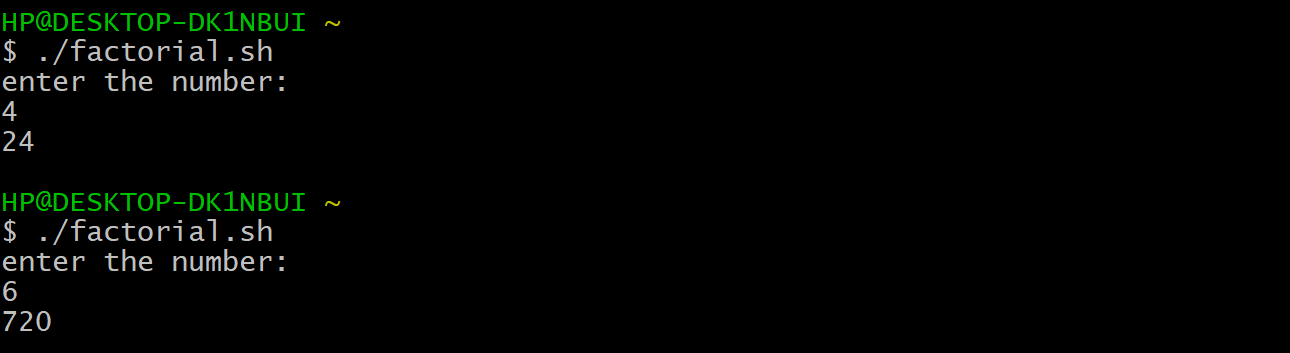


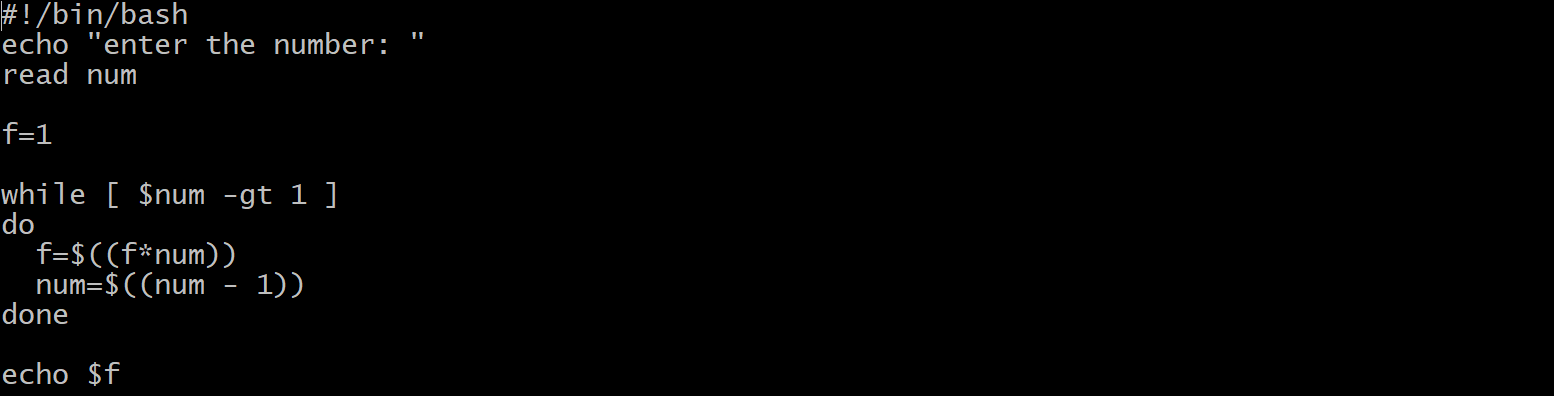
1. **Find positive or negative**



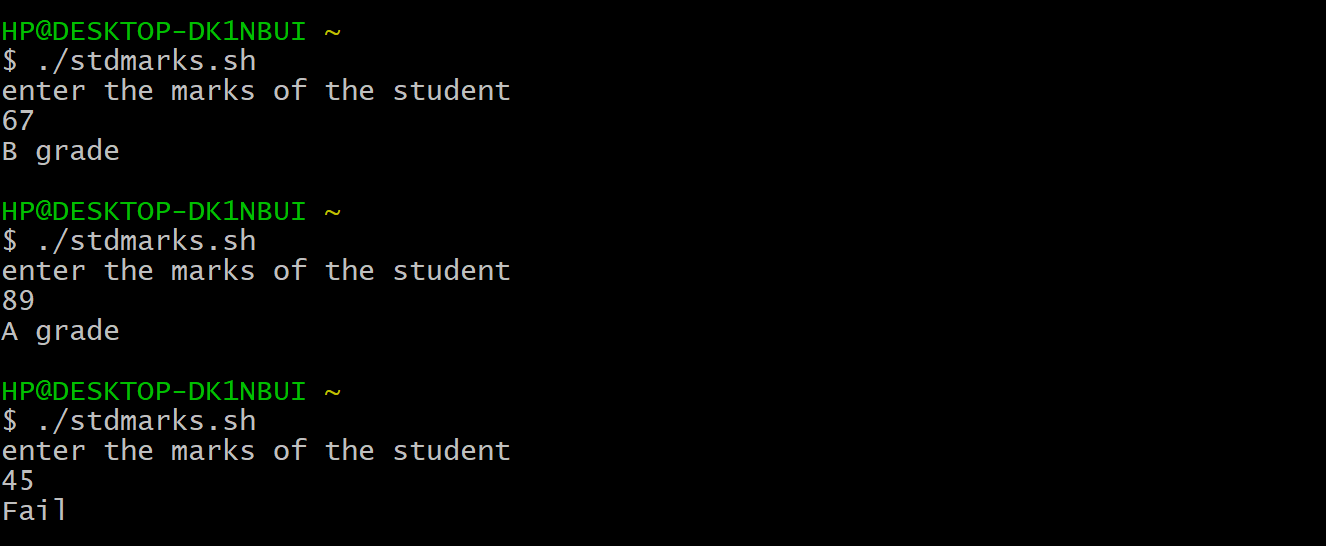


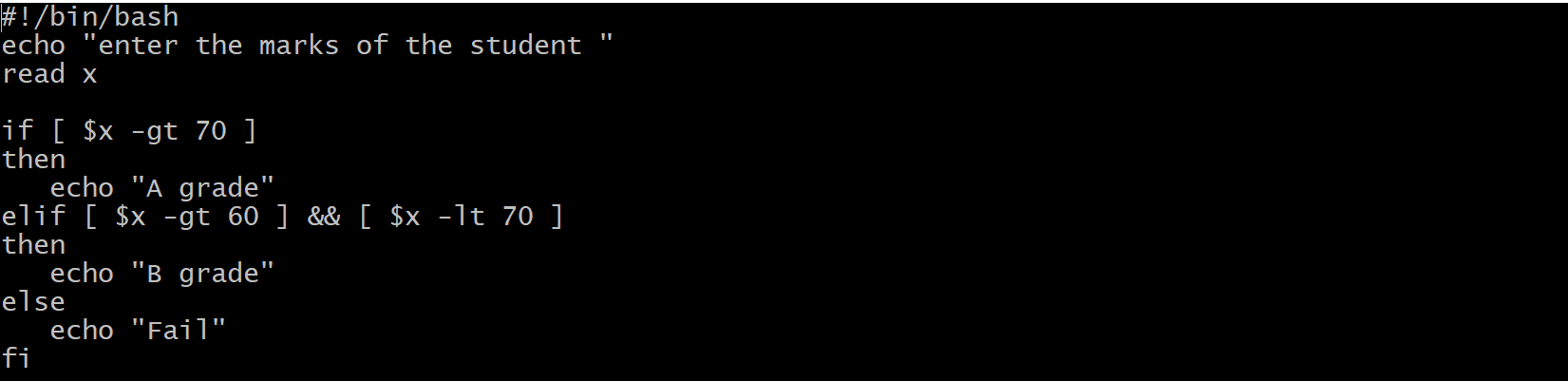
1. **Factorial**



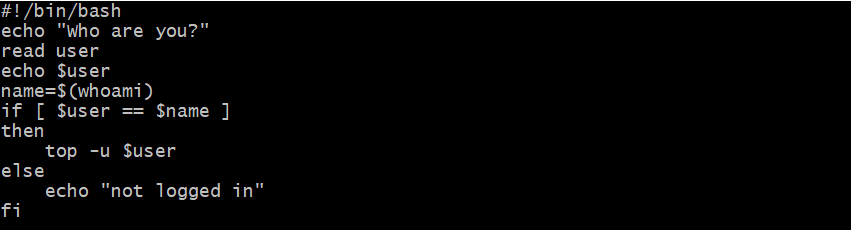


1. **Student marks**





1. **Checkuser**



Result: Scripts executed successfully.

**Experiment-07**

Date: 11/02/2021

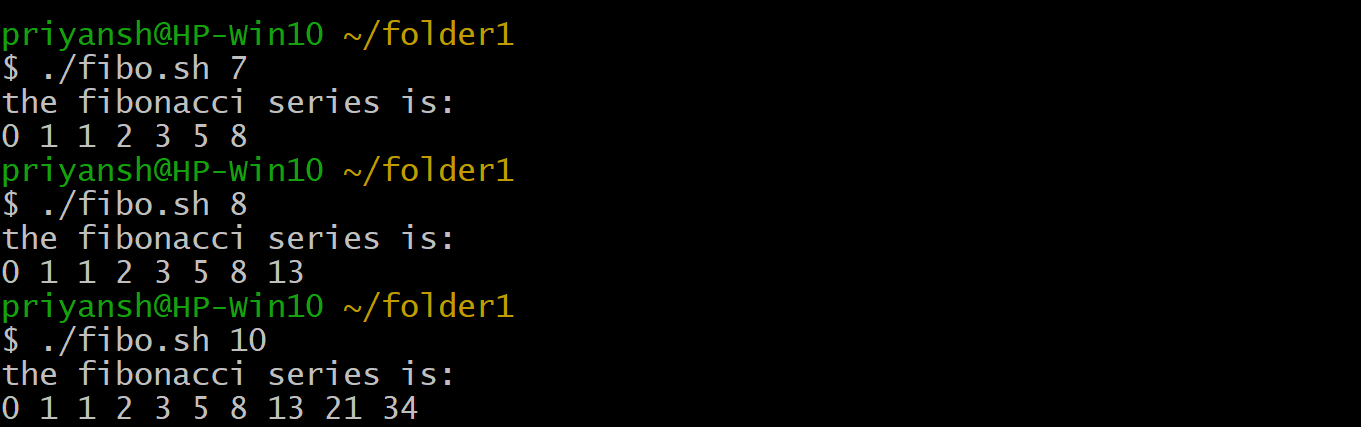
Objective: To execute Linux Scripts on terminal.

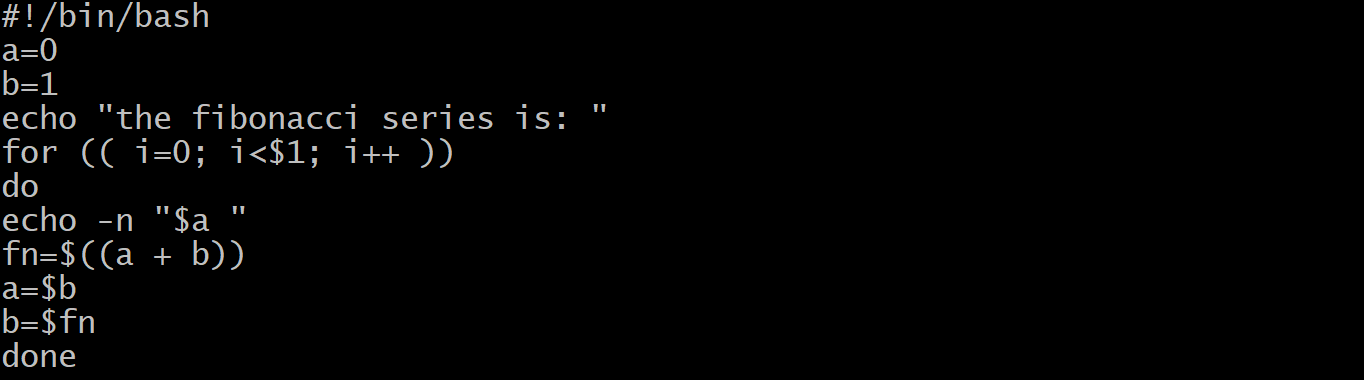
Software used: Cygwin

OS: Windows 10

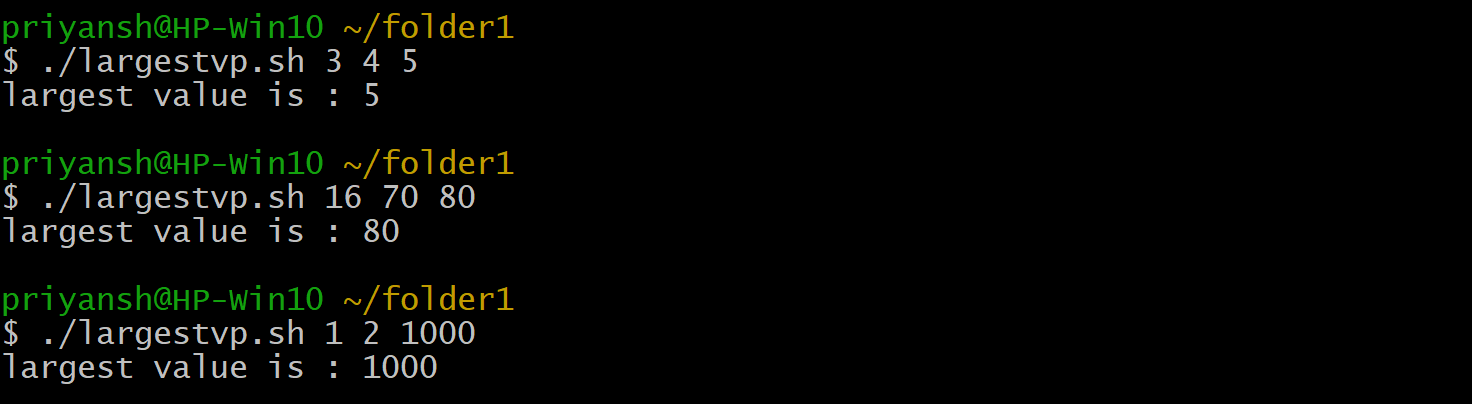
Theory:

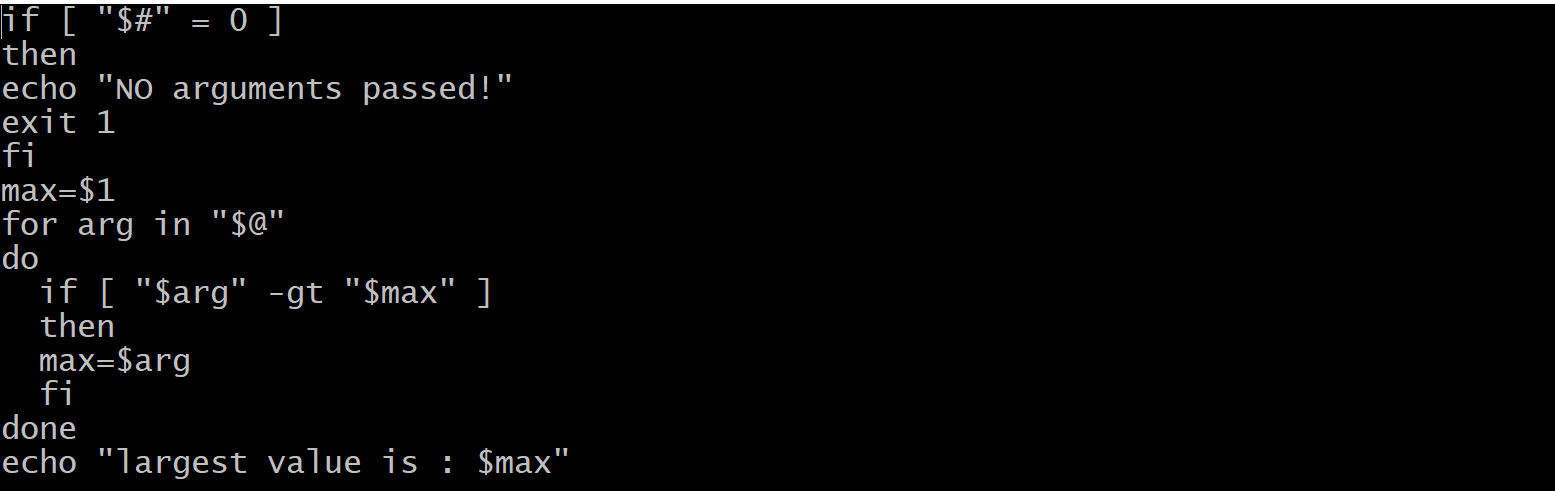
1. **Fibonacci Series**



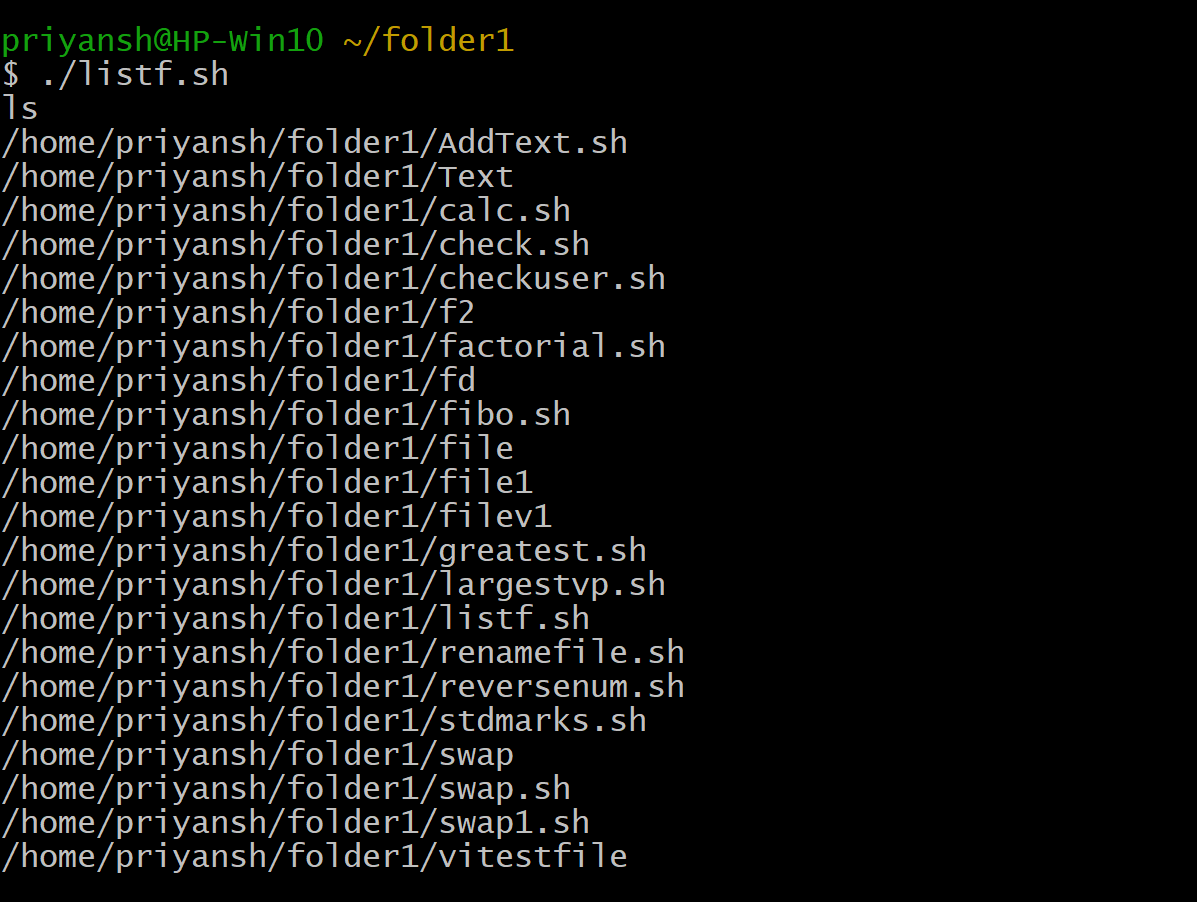


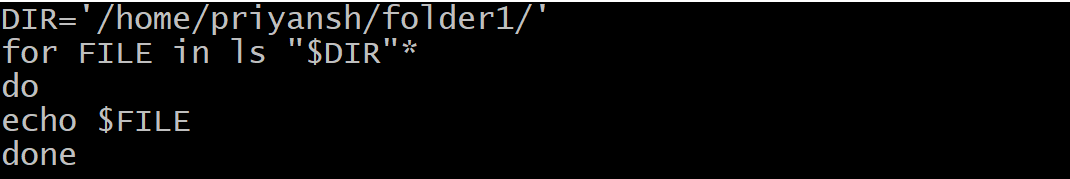
1. **Largest Value Passed**



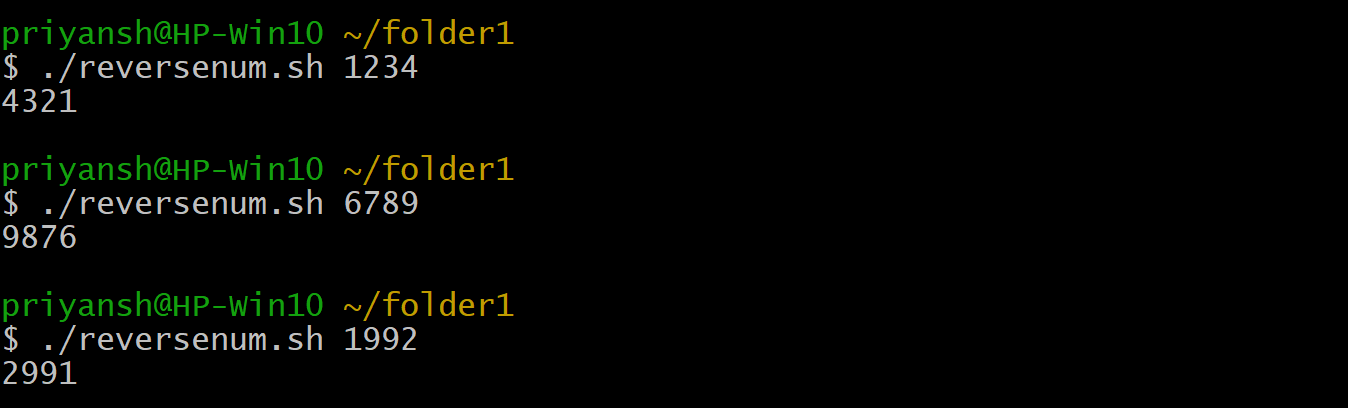


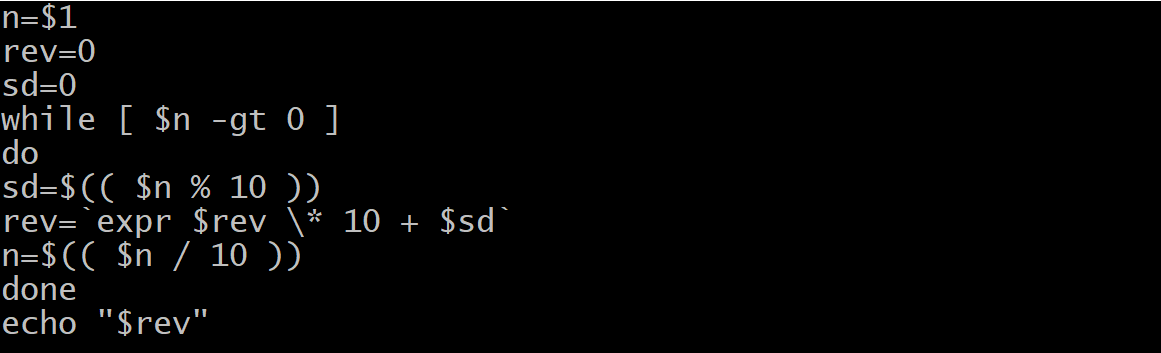
1. **List Files**



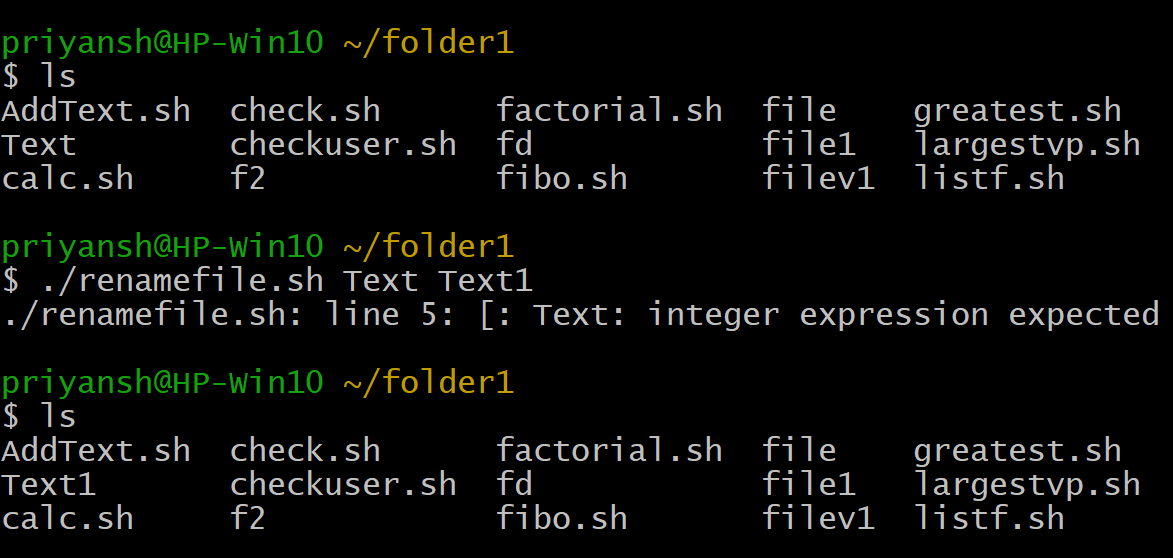


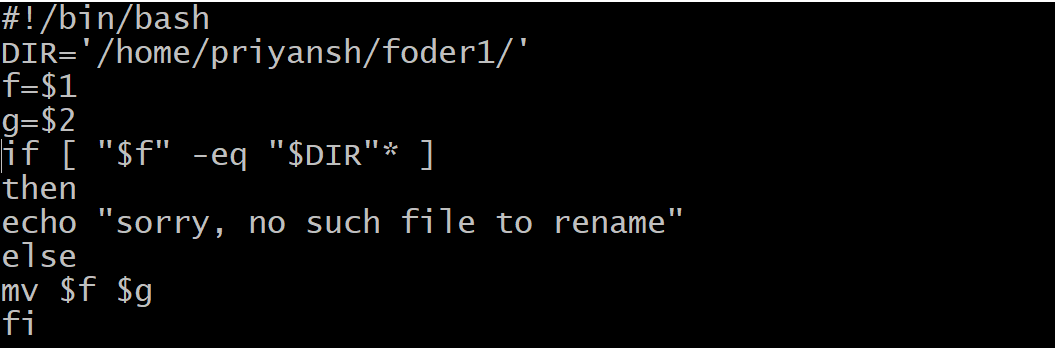
1. **Reverse number**





1. **Rename File**





Result : Scripts executed successfully.

**Experiment-08**

Date: 18/02/2021

Objective: Write a program to simulate FCFS CPU Scheduling algorithm.

Software used: Online Compiler JAVA

OS: Windows 10

**Code**:

import java.util.\*;

public class FCFS {

public static void main(String args[])

{

Scanner sc = new Scanner(System.in);

System.out.println("enter no of process: ");

int n = sc.nextInt();

int pid[] = new int[n];

int ar[] = new int[n];

int bt[] = new int[n];

int ct[] = new int[n];

int ta[] = new int[n];

int wt[] = new int[n];

int temp;

float avgwt=0,avgta=0;

for(int i = 0; i < n; i++)

{

System.out.println("enter process " + (i+1) + " arrival time: ");

ar[i] = sc.nextInt();

System.out.println("enter process " + (i+1) + " brust time: ");

bt[i] = sc.nextInt();

pid[i] = i+1;

}

for(int i = 0 ; i <n; i++)

{

for(int j=0; j < n-(i+1) ; j++)

{

if( ar[j] > ar[j+1] )

{

temp = ar[j];

ar[j] = ar[j+1];

ar[j+1] = temp;

temp = bt[j];

bt[j] = bt[j+1];

bt[j+1] = temp;

temp = pid[j];

pid[j] = pid[j+1];

pid[j+1] = temp;

}

}

}

for(int i = 0 ; i < n; i++)

{

if( i == 0)

{

ct[i] = ar[i] + bt[i];

}

else

{

if( ar[i] > ct[i-1])

{

ct[i] = ar[i] + bt[i];

}

else

ct[i] = ct[i-1] + bt[i];

}

ta[i] = ct[i] - ar[i] ;

wt[i] = ta[i] - bt[i] ;

avgwt += wt[i] ;

avgta += ta[i] ;

}

System.out.println("\npid arrival brust complete turn waiting");

for(int i = 0 ; i< n; i++)

{

System.out.println(pid[i] + " \t " + ar[i] + "\t" + bt[i] + "\t" + ct[i] + "\t" + ta[i] + "\t" + wt[i] ) ;

}

sc.close();

System.out.println("Gnatt Chart: ");

for(int i = 0; i<n; i++){

System.out.println("P"+ pid[i] + " ");

}

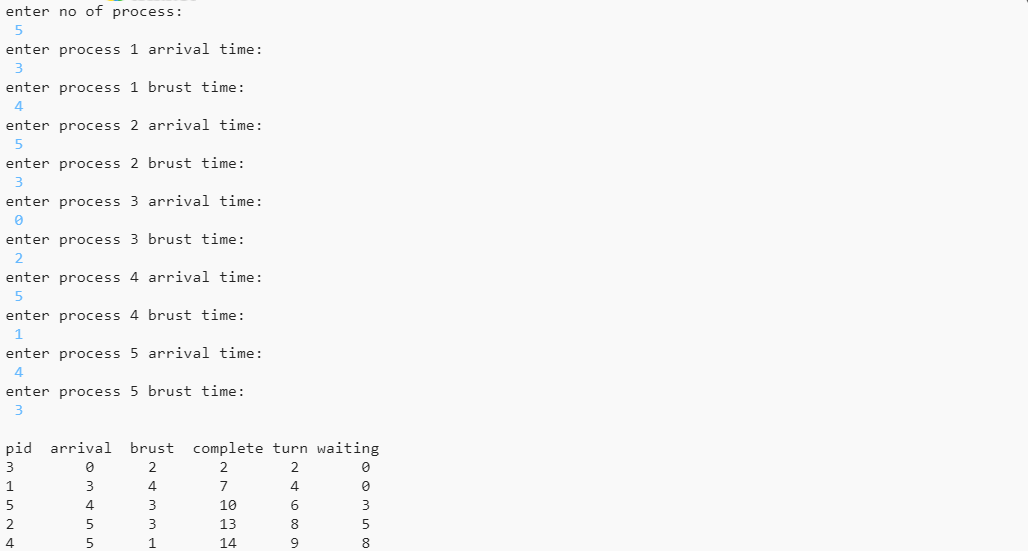
System.out.println("\naverage waiting time: "+ (avgwt/n));

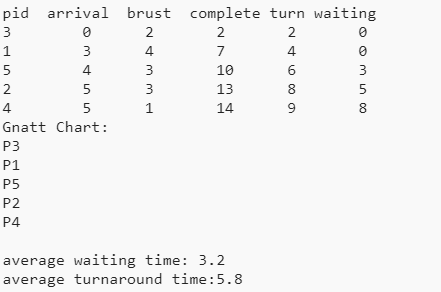
System.out.println("average turnaround time:"+(avgta/n));

}

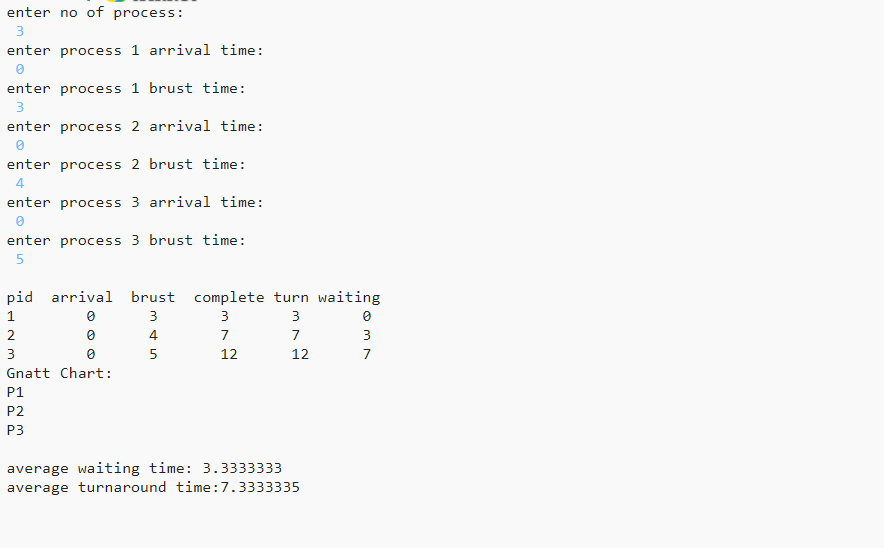
}

**Output-1**





**Output 2:**

****

Result: Code executed successfully.

**Experiment-09**

Date: 04/03/2021

Objective: Write a program to simulate pre-emptive and non-pre-emptive SJF CPU Scheduling algorithm.

Software used: Online Compiler JAVA

OS: Windows 10

**Code**:

1. **Non-preemptive**

import java.util.\*;

public class SJF {

public static void main(String args[])

{

Scanner sc = new Scanner(System.in);

System.out.println ("enter no of process:");

int n = sc.nextInt();

int pid[] = new int[n];

int at[] = new int[n]; // at means arrival time

int bt[] = new int[n]; // bt means burst time

int ct[] = new int[n]; // ct means complete time

int ta[] = new int[n]; // ta means turn around time

int wt[] = new int[n]; //wt means waiting time

int f[] = new int[n]; // f means it is flag it checks process is completed or not

int st=0, tot=0;

float avgwt=0, avgta=0;

for(int i=0;i<n;i++)

{

System.out.println ("enter process " + (i+1) + " arrival time:");

at[i] = sc.nextInt();

System.out.println ("enter process " + (i+1) + " brust time:");

bt[i] = sc.nextInt();

pid[i] = i+1;

f[i] = 0;

}

boolean a = true;

while(true)

{

int c=n, min=999;

if (tot == n) // total no of process = completed process loop will be terminated

break;

for (int i=0; i<n; i++)

{

/\*

\* If i'th process arrival time <= system time and its flag=0 and burst<min

\* That process will be executed first

\*/

if ((at[i] <= st) && (f[i] == 0) && (bt[i]<min))

{

min=bt[i];

c=i;

}

}

/\* If c==n means c value can not updated because no process arrival time< system time so we increase the system time \*/

if (c==n)

st++;

else

{

ct[c]=st+bt[c];

st+=bt[c];

ta[c]=ct[c]-at[c];

wt[c]=ta[c]-bt[c];

f[c]=1;

tot++;

}

}

System.out.println("\npid arrival brust complete turn waiting");

for(int i=0;i<n;i++)

{

avgwt+= wt[i];

avgta+= ta[i];

System.out.println(pid[i]+"\t"+at[i]+"\t"+bt[i]+"\t"+ct[i]+"\t"+ta[i]+"\t"+wt[i]);

}

System.out.println ("\naverage tat is "+ (float)(avgta/n));

System.out.println ("average wt is "+ (float)(avgwt/n));

sc.close();

}

}

1. **preemptive**

import java.util.\*;

public class SRTF {

public static void main (String args[])

{

Scanner sc=new Scanner(System.in);

System.out.println ("enter no of process:");

int n= sc.nextInt();

int pid[] = new int[n]; // it takes pid of process

int at[] = new int[n]; // at means arrival time

int bt[] = new int[n]; // bt means burst time

int ct[] = new int[n]; // ct means complete time

int ta[] = new int[n];// ta means turn around time

int wt[] = new int[n];  // wt means waiting time

int f[] = new int[n];  // f means it is flag it checks process is completed or not

int k[]= new int[n];   // it is also stores brust time

    int i, st=0, tot=0;

    float avgwt=0, avgta=0;

    for (i=0;i<n;i++)

    {

     pid[i]= i+1;

     System.out.println ("enter process " +(i+1)+ " arrival time:");

     at[i]= sc.nextInt();

     System.out.println("enter process " +(i+1)+ " burst time:");

     bt[i]= sc.nextInt();

     k[i]= bt[i];

     f[i]= 0;

    }

    while(true){

     int min=99,c=n;

     if (tot==n)

     break;

     for ( i=0;i<n;i++)

     {

     if ((at[i]<=st) && (f[i]==0) && (bt[i]<min))

     {

     min=bt[i];

     c=i;

     }

     }

     if (c==n)

     st++;

     else

     {

     bt[c]--;

     st++;

     if (bt[c]==0)

     {

     ct[c]= st;

     f[c]=1;

     tot++;

     }

     }

    }

    for(i=0;i<n;i++)

    {

     ta[i] = ct[i] - at[i];

     wt[i] = ta[i] - k[i];

     avgwt+= wt[i];

     avgta+= ta[i];

    }

    System.out.println("pid  arrival  burst  complete turn waiting");

    for(i=0;i<n;i++)

    {

     System.out.println(pid[i] +"\t"+ at[i]+"\t"+ k[i] +"\t"+ ct[i] +"\t"+ ta[i] +"\t"+ wt[i]);

    }

    System.out.println("\naverage tat is "+ (float)(avgta/n));

    System.out.println("average wt is "+ (float)(avgwt/n));

    sc.close();

}

Result: Code executed successfully.

**Experiment-10**

Date: 18/03/2021

Objective: Write a program to simulate round robin CPU Scheduling algorithm.

Software used: C Compiler

OS: Windows 10

**Code**:

**Round Robin**

#include<stdio.h>

int main()

{

int count,j,n,time,remain,flag=0,time\_quantum;

int wait\_time = 0,turnaround\_time = 0,at[10],bt[10],rt[10];

printf("Enter Total Process: ");

scanf("%d",&n);

remain = n;

for(count = 0; count < n; count++)

{

printf("Enter Arrival Time and Burst Time for Process Process Number %d :", count + 1);

scanf("%d", &at[count]);

scanf("%d", &bt[count]);

rt[count] = bt[count];

}

printf("Enter Time Quantum: ");

scanf("%d", &time\_quantum);

printf("\n\nProcess\t | Turnaround Time | Waiting Time\n\n");

for(time = 0,count = 0; remain != 0;)

{

if(rt[count] <= time\_quantum && rt[count] > 0)

{

time += rt[count];

rt[count] = 0;

flag = 1;

}

else if(rt[count]>0)

{

rt[count] -= time\_quantum;

time += time\_quantum;

}

if(rt[count] ==0 && flag ==1)

{

remain--;

printf("P[%d]\t | \t%d\t | \t%d\n",count+1,time-at[count],time-at[count]-bt[count]);

wait\_time += time - at[count] - bt[count];

turnaround\_time += time - at[count];

flag = 0;

}

if(count == n-1)

count = 0;

else if(at[count+1] <= time)

count++;

else

count = 0;

}

printf("\nAverage Waiting Time = %f\n",wait\_time\*1.0/n);

printf("Avg Turnaround Time = %f",turnaround\_time\*1.0/n);

return 0;

}

Result: Code executed successfully.

**Experiment-11**

Date: 25/03/2021

Objective: Write a program to simulate banker’s algorithm.

Software used: C++ Compiler

OS: Windows 10

**Code**:

**Banker’s Algo**

#include <iostream>

#include <stdlib.h>

using namespace std;

void print(int x[][10], int n, int m)

{

int i, j;

for (i = 0; i < n; i++)

{

cout << "\n";

for (j = 0; j < m; j++)

{

cout << x[i][j] << "\t";

}

}

}

void res\_request(int A[10][10], int N[10][10], int AV[10][10], int pid, int m)

{

int reqmat[1][10];

int i;

cout << "\n Enter additional request: \n";

for (i = 0; i < m; i++)

{

cout << "Request for resource " << i + 1 << ": ";

cin >> reqmat[0][i];

}

for (i = 0; i < m; i++)

if (reqmat[0][i] > N[pid][i])

{

cout << "\n Error encountered.\n";

exit(0);

}

for (i = 0; i < m; i++)

if (reqmat[0][i] > AV[0][i])

{

cout << "\n Resources unavailable.\n";

exit(0);

}

for (i = 0; i < m; i++)

{

AV[0][i] -= reqmat[0][i];

A[pid][i] += reqmat[0][i];

N[pid][i] -= reqmat[0][i];

}

}

int safety(int A[][10], int N[][10], int AV[1][10], int n, int m, int a[])

{

int i, j, k, x = 0;

int F[10], W[1][10];

int pflag = 0, flag = 0;

for (i = 0; i < n; i++)

F[i] = 0;

for (i = 0; i < m; i++)

W[0][i] = AV[0][i];

for (k = 0; k < n; k++)

{

for (i = 0; i < n; i++)

{

if (F[i] == 0)

{

flag = 0;

for (j = 0; j < m; j++)

{

if (N[i][j] > W[0][j])

flag = 1;

}

if (flag == 0 && F[i] == 0)

{

for (j = 0; j < m; j++)

W[0][j] += A[i][j];

F[i] = 1;

pflag++;

a[x++] = i;

}

}

}

if (pflag == n)

return 1;

}

return 0;

}

void accept(int A[][10], int N[][10], int M[10][10], int W[1][10], int \*n, int \*m)

{

int i, j;

cout << "\n Enter total no. of processes: ";

cin >> \*n;

cout << "\n Enter total no. of resources: ";

cin >> \*m;

for (i = 0; i < \*n; i++)

{

cout << "\n Process " << i + 1 << " :\n";

for (j = 0; j < \*m; j++)

{

cout << " Allocation for resource " << j + 1 << ": ";

cin >> A[i][j];

cout << " Maximum for resource " << j + 1 << ": ";

cin >> M[i][j];

}

}

cout << "\n Available resources: \n";

for (i = 0; i < \*m; i++)

{

cout << " Resource " << i + 1 << ": ";

cin >> W[0][i];

}

for (i = 0; i < \*n; i++)

for (j = 0; j < \*m; j++)

N[i][j] = M[i][j] - A[i][j];

cout << "\n Allocation Matrix";

print(A, \*n, \*m);

cout << "\n Maximum Requirement Matrix";

print(M, \*n, \*m);

cout << "\n Need Matrix";

print(N, \*n, \*m);

}

int banker(int A[][10], int N[][10], int W[1][10], int n, int m)

{

int j, i, a[10];

j = safety(A, N, W, n, m, a);

if (j != 0)

{

cout << "\n\n";

for (i = 0; i < n; i++)

cout << "P " << a[i];

cout << "\nA safety sequence has been detected.\n";

return 1;

}

else

{

cout << "\nDeadlock has occured.\n";

return 0;

}

}

int main()

{

int ret;

int A[10][10];

int M[10][10];

int N[10][10];

int W[1][10];

int n, m, pid, ch;

cout << "\n DEADLOCK AVOIDANCE USING BANKER'S ALGORITHM\n";

accept(A, N, M, W, &n, &m);

ret = banker(A, N, W, n, m);

if (ret != 0)

{

cout << "\n Do you want make an additional request ? (1=Yes|0=No)";

cin >> ch;

if (ch == 1)

{

cout << "\n Enter process no. : ";

cin >> pid;

res\_request(A, N, W, pid - 1, m);

ret = banker(A, N, W, n, m);

if (ret == 0)

exit(0);

}

}

else

exit(0);

return 0;

}

Result: Code executed successfully.

**Experiment-12**

Date: 01/04/2021

Objective: Write a program to simulate Shortest Seek Time First(SSTF) Disk Scheduling algorithm.

Software used: C++ Compiler

OS: Windows 10

**Code**:

**SSTF Algorithm**

#include<bits/stdc++.h>

using namespace std;

int main(){

int i,j,k,n,m,sum=0,x,y,h;

cout<<"Enter the size of disk\n";

cin>>m;

cout<<"Enter number of requests\n";

cin>>n;

cout<<"Enter the requests\n";

vector <int> a(n),b;

//creating a map to store the count of each element

//in the array a.

map <int,int> mp;

**for**(i=0;i<n;i++){

cin>>a[i];

mp[a[i]]++;

}

**for**(i=0;i<n;i++){

**if**(a[i]>m){

cout<<"Error, Unknown position "<<a[i]<<"\n";

**return** 0;

}

}

cout<<"Enter the head position\n";

cin>>h;

int temp=h;

int ele;

b.push\_back(h);

int count=0;

**while**(count<n){

//initially taking diff to be very large.

int diff=999999;

//traversing in map to find the least difference

**for**(auto q:mp){

**if**(abs(q.first-temp)<diff){

ele=q.first;

diff=abs(q.first-temp);

}

}

//deleting the element that has the least

//difference from the map

mp[ele]--;

**if**(mp[ele]==0){

mp.erase(ele);

}

//adding that element to our output array.

b.push\_back(ele);

temp=ele;

count++;

}

//printing the output array

cout<<b[0];

temp=b[0];

**for**(i=1;i<b.size();i++){

cout<<" -> "<<b[i];

sum+=abs(b[i]-temp);

temp=b[i];

}

cout<<'\n';

cout<<"Total head movements = "<< sum<<'\n';

cout<<"Average head movement = "<<(float)sum/n<<'\n';

**return** 0;

}

Result: Code executed successfully.