THE NORTHCAP UNIVERSITY

POWERED BY Arizona State University

Operating System

Lab Manual

Department of Computer Science and Engineering
The NorthCap University, Gurugram

Operating System

CSL303

Author Name

Dr Priyanka Vashisht

Dr Anuradha Dhull

Ms Monika Yadav

Ms Jyoti Yadav

Ms Ishita

Ms Akansha Singh

Department of Computer Science and Engineering

NorthCap University, Gurugram- 122001, India

Session 2021-22

Published by:

School of Engineering and Technology

Department of Computer Science & Engineering

The NorthCap University Gurugram

Copying or facilitating copying of lab work comes under cheating and is considered as use of unfair means. Students indulging in copying or facilitating copying shall be awarded zero marks for that particular experiment. Frequent cases of copying may lead to disciplinary action. Attendance in lab classes is mandatory.

Labs are open up to 7 PM upon request. Students are encouraged to make full use of labs beyond normal lab hours.

PREFACE

Operating System Lab Manual is designed to meet the course and program requirements of NCU curriculum for B.Tech III year students of CSE branch. The concept of the lab work is to give brief practical experience for basic lab skills to students. It provides the space and scope for self-study so that students can come up with new and creative ideas.

The Lab manual is written on the basis of "teach yourself pattern" and expected that students who come with proper preparation should be able to perform the experiments without any difficulty. Brief introduction to each experiment with information about self-study material is provided. The laboratory exercises will include familiarization with LINUX system calls for process management and inter-process communication; Experiments on process scheduling and other operating system tasks through simulation/implementation. Students would require design process synchronization, CPU scheduling algorithms, memory management and disc management algorithms in high level languages like c, c++, python. Finally, the students would require applying the operating system concepts by experimenting on either xv6/minix operating systems. At the start of each experiment a question bank for preparation and practice is suggested which may be used to test the basic understanding of the students about the experiment. Students are expected to come thoroughly prepared for the lab. General disciplines, safety guidelines and report writing are also discussed.

The lab manual is a part of curriculum for the TheNorthCap University, Gurugram. Teacher's copy of the experimental results and answer for the questions are available as sample guidelines.

We hope that lab manual would be useful to students of CSE, IT, ECE and BSc branches and author requests the readers to kindly forward their suggestions / constructive criticism for further improvement of the work book.

Author expresses deep gratitude to Members, Governing Body-NCU for encouragement and motivation.

Authors

The NorthCap University
Gurugram, India

CONTENTS

SYLLABUS

1. Department:	**Department of CSE**		
2. Course Name: Operating Systems	**3. Course Code :**	**4. L- P**	**5. Credits**
	Code: CSL 303	3-2	4
6. Type of Course (Check one):	Programme Core ☐ Programme Elective ☐ Open Elective ☐		
7. Frequency of offering (check one): Odd ☐ Even ☐ Either Sem. ☐ Every Sem. ☐			
8. Brief Syllabus: This is an introductory course which briefs LINUX Operating System Concepts that forms an integral part of computer science engineering in development of software applications in many diverse areas, including Web Development, Windows Applications, Research, Analytics and Processing. It lays the foundation of Process Management & Scheduling, Memory Management, Deadlocks and other Operating system Concepts.			
9. Total lecture and Practical Hours for this course: 30 Hours The class size is maximum 30 learners.			
10. Course Outcomes (COs) Possible usefulness of this course after its completion i.e. how this course will be practically useful to him once it is completed			

CO 1	The students will be able to understand the basic architecture of Linux.
CO 2	The students will be able to understand the process management & scheduling of Linux.
CO 3	The students will be able to understand the memory management of Linux.
CO 4	The students will be able to understand the inter process communication of Linux.
CO 5	They will understand the main principles and techniques to handle the deadlocks.
CO6	They will understand the I/O device management & the VFS of Linux.

11. UNIT WISE DETAILS No. of Units: -05
Unit 1: Introduction to Linux OS Hours: 6 Introduction & overview: functions of operating systems, Overview of various Operating Systems, Linux architecture, Boot strap loader of Linux, Tasks of the kernel, implementation strategies of kernel, System Calls.
Unit II: Process Management & Scheduling Hours: 6 Process priorities, process life cycle of Linux, process representation: process types, process identification numbers, process management system calls, kernel thread, overview of different scheduling algorithms, Linux scheduler: priority and completely fair share scheduling algorithm.
Unit III: Process Synchronization and Memory Management Hours: 8 Implementation of Producer- Consumer problem, implementation of semaphores, Page-Replacement Algorithms.
Unit IV: Deadlocks Hours: 6 Implementation of Banker's Algorithm,
Unit V: Virtual File System Hours: 4 Disk scheduling algorithms, Introduction to VFS File System types, Common File model, Structure of the VFS
12. Guided Project (No. of Hours): Case Study on Windows OS **13. Unguided Project (No. of Hours):** Case Study of Linux, Window, MAC OS
14. Brief Description of Self-learning component by students (through books/resource material etc.): Topics: Linux syntax for shell scripting, revise c/c++/Python and data structure concepts from previous semesters
15. Suggested Readings GNU/Linux Command−Line Tools Summary [eBook] http://www.tldp.org/LDP/Bash-Beginners-Guide/Bash-Beginners-Guide.pdf **websites:** ● https://www.linuxjournal.com/ ● https://www.omgubuntu.co.uk/

1. INTRODUCTION

That 'learning is a continuous process' cannot be over emphasized. The theoretical knowledge gained during lecture sessions need to be strengthened through practical experimentation. Thus practical makes an integral part of a learning process.

The purpose of conducting experiments can be stated as follows:

- To familiarize the students with the basic concepts, programming skill development and the take home laboratory assignments mainly implementation-oriented which have to be coded in high level language. The lab sessions will be based on exploring the concepts discussed in class.
- Observing basic structure and characteristics of Operating Systems
- Reporting and analyzing the complexities.
- Hands on experience on the experimental setup and software tools

2. LAB REQUIREMENTS

S.No.	Requirements	Details
1	**Software Requirements**	Linux's Shell, Python/c/c++
2	**Operating System**	Linux Operating System
3	**Hardware Requirements**	Windows and Linux: Intel 64/32 or AMD Athlon 64/32, or AMD Opteron processor 2 GB RAM 80 GB hard disk space
4	**Required Bandwidth**	NA

3. GENERAL INSTRUCTIONS

3.1 General discipline in the lab

- Students must turn up in time and contact concerned faculty for the experiment they are supposed to perform.
- Students will not be allowed to enter late in the lab.
- Students will not leave the class till the period is over.
- Students should come prepared for their experiment.
- Experimental results should be entered in the lab report format and certified/signed by concerned faculty/ lab Instructor.
- Students must get the connection of the hardware setup verified before switching on the power supply.
- Students should maintain silence while performing the experiments. If any necessity arises for discussion amongst them, they should discuss with a very low pitch without disturbing the adjacent groups.
- Violating the above code of conduct may attract disciplinary action.
- Damaging lab equipment or removing any component from the lab may invite penalties and strict disciplinary action.

3.2 Attendance

- Attendance in the lab class is compulsory.
- Students should not attend a different lab group/section other than the one assigned at the beginning of the session.
- On account of illness or some family problems, if a student misses his/her lab classes, he/she may be assigned a different group to make up the losses in consultation with the concerned faculty / lab instructor. Or he/she may work in the lab during spare/extra hours to complete the experiment. No attendance will be granted for such case**.**

3.3 Preparation and Performance

- Students should come to the lab thoroughly prepared on the experiments they are assigned to perform on that day. Brief introduction to each experiment with information about self study reference is provided on LMS.
- Students must bring the lab report during each practical class with written records of the last experiments performed complete in all respect.

- Each student is required to write a complete report of the experiment he has performed and bring to lab class for evaluation in the next working lab. Sufficient space in work book is provided for independent writing of theory, observation, calculation and conclusion.
- Students should follow the Zero tolerance policy for copying / plagiarism. Zero marks will be awarded if found copied. If caught further, it will lead to disciplinary action.
- Refer **Annexure 1** for Lab Report Format

4. LIST OF EXPERIMENTS

Exp. No.	Division of Experiments	List of Experiments	Software Used	Unit Covered	CO Covered	Time Required
1	Basics of Linux	Explain the structure of Linux Operating System	**Cygwin**	1	CO1	2 Hrs
		Installation of Ubuntu Operating system	Cygwin	1	CO1	2 Hrs
1	Shell Programs	Write a shell program to find factorial of a number.		1	CO1	2 Hrs
2		Write a shell program to find gross salary of an employee.	Cygwin	1	CO1	2 Hrs
3		Write a shell program to display the menu and execute instructions accordingly (i)List of file (ii)Process Status (iii) Date (iv) users in program (v) Quit	Cygwin	1	CO1	2 Hrs
4		Write a shell program to find Fibonacci series.	Cygwin	1	CO1	2 Hrs
5		Write a shell program to find largest of three numbers.	Cygwin	1	CO1	2 Hrs
6		Write a shell program to find average of N numbers	Cygwin	1	CO1	2 Hrs
7	CPU Scheduling Algorithms	Write a C program to simulate the following non-preemptive CPU scheduling algorithms to find	C++/ Python/ Java	2	CO2	2 Hrs

		turnaround time and waiting time. a) FCFS b) SJF c) Round Robin (pre-emptive) d) Priority				
8		*Write a C program to simulate multi-level queue scheduling algorithm considering the following scenario. All the processes in the system are divided into two categories – system processes and user processes. System processes are to be given higher priority than user processes. Use FCFS scheduling for the processes in each queue.	C++/ Python/ Java	2	CO2	2 Hrs
		Implement the following CPU scheduling Algorithms. i) Round Robin	C++/ Python/ Java	2	CO2	2 Hrs

		ii) Priority Based				
9	Deadlock Management Technique	Write a program to simulate Bankers algorithm for the purpose of deadlock avoidance	C++/ Python/ Java	3	CO3	2 Hrs
10	Page Replacement Algorithms	Write a C program to simulate page replacement algorithms a) FIFO b) LRU c) LFU	C++/ Python/ Java	4	CO4	2 Hrs
11		Write a C program to simulate page replacement algorithms a) Optimal	C++/ Python/ Java	4	CO4	2 Hrs

5. LIST OF FLIP EXPERIMENTS

5.1 Execute the **who** command written in a file to instruct the shell to read input from a file called "myfile1" instead of from the keyboard. Use the **more** command to see the contents of myfile1.

5.2 Use the date and who commands in sequence (in one line) such that the output of date will display on the screen and the output of who will be redirected to a file called myfile2. Use the more command to check the contents of myfile2

5.3 Write a sed command that swaps the first and second words in each line in a file.

5.4 Write a shell script that takes a command –line argument and reports on whether it is directory, a file, or something else.

5.5 Write a shell script that accepts one or more file name as arguments and converts all of them to uppercase, provided they exist in the current directory.

5.6 Write a shell script that determines the period for which a specified user is working on the system.

5.7 Write a shell script that accepts a file name, starting and ending line numbers as arguments and displays all the lines between the given line numbers.

5.8 Write a shell script that deletes all lines containing a specified word in one or more files supplied as arguments to it.

5.9 Write a shell script to perform the following string operations:

i. To extract a sub-string from a given string

ii. To find the length of a given string

6. LIST OF PROJECTS

1. Case Study of Window OS
2. Case Study of Linux OS
3. Case Study of MAC OS

7. RUBRICS

Marks Distribution	
Continuous Evaluation(50 Marks)	**End Semester Exam (20 Marks)**
Each experiment shall be evaluated for 10 marks and at the end of the semester proportional marks shall be awarded out of 50.	End semester practical evaluation including Mini project (if any) carries 20 marks.
Following is the breakup of 10 marks for each **4 Marks**: Observation & conduct of experiment. Teacher may ask questions about experiment. **3 Marks:** For report writing **3 Marks:** For the 15 minutes quiz to be conducted in every lab.	

Annexure 1

Operating System

(CSL 303)

Lab Practical Report

Faculty name Dr. Priyanka Vashisht

Student name : Namit Kumar

Roll No.:19CSU185

Semester:V

Group:A1

Department of Computer Science and Engineering

NorthCap University, Gurugram- 122001, India

Session 2021-22

INDEX

	d) Write a shell program to type number 1 to 7 and then print its corresponding day of week						
5	Implement the following CPU scheduling Algorithms. i. FCFS with Arrival time ii. FCFS without Arrival time	46	8th September	8th September		CO2	
6	Implement the following CPU scheduling Algorithms. • SJF (Non-Preemptive) • SJTF (shortest remaining time first -Preemptive SJF)	51	8th September	8th September		CO2	
7	Implement the priority scheduling.	60	6th October	6th October		CO2	
8	Implement the Round Robin Algorithm.	66	13th October	13th October		CO2	
9	Write a program to implement reader/writer problem using semaphore	72	20th October	27th October		CO3	
10	Write a program to implement Dining Philosopher problem using semaphore	79	10th November	10th November		CO3	
11	Write a program to implement Banker's algorithm for deadlock avoidance.	88	17th November			CO3	
12	CASE STUDY	94	17th November				

Experiment No: 1

Student Name and Roll Number: Namit Kumar 19CSU185	
Semester /Section: V/FS-A-1	
Link to Code: https://github.com/NamitKumar16/OS	
Date: 6th August 2021	
Faculty Signature:	
Marks:	

Objective(s):
To familiarize the students to Linux interface.

Outcome:
- The students will understand commands used in Linux.

Problem Statement:

Implement the following things:

- Cygwin Installation
- Basic Linux commands

Background Study:

Cygwin is a open source tool which provides that functionality of the Linux in windows Operating System. Cygwin is a large collection of GNU and Open Source tools which provide functionality similar to a Linux distribution on Windows. It is a DLL (cygwin1.dll) which provides substantial POSIX API functionality.

Question Bank:

1. **What is Linux?**

2. How will you List files from a directory?
3. How files in a directory can be removed?
4. How to find out a word in a file?
5. What are wildcards?

Student Work Area

Algorithm/Flowchart/Code/Sample Outputs

Q1. Linux is a Unix-like, open source and community-developed operating system for computers, servers, mainframes, mobile devices and embedded devices. It is supported on almost every major computer platform including x86, ARM and SPARC, making it one of the most widely supported operating systems.

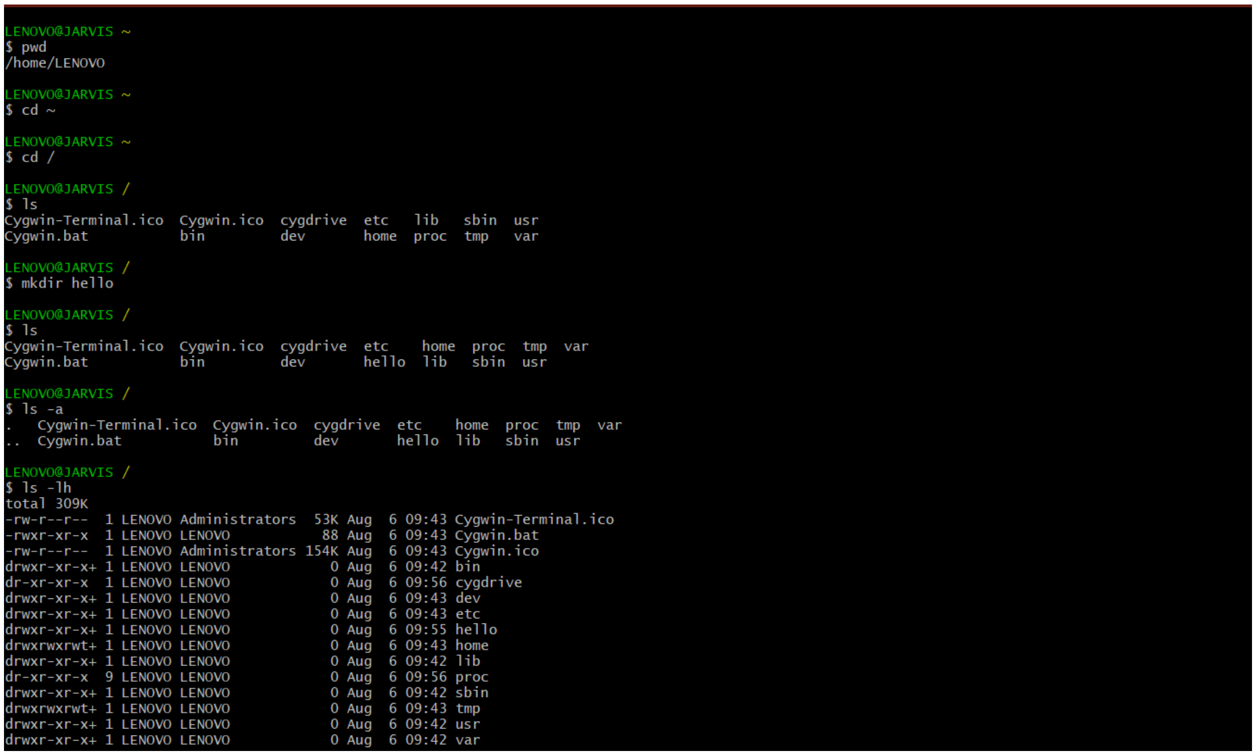
Q2. The ls command is used to list files or directories in Linux and other Unix-based operating systems.

Q3. Files can be removed using the rm command.

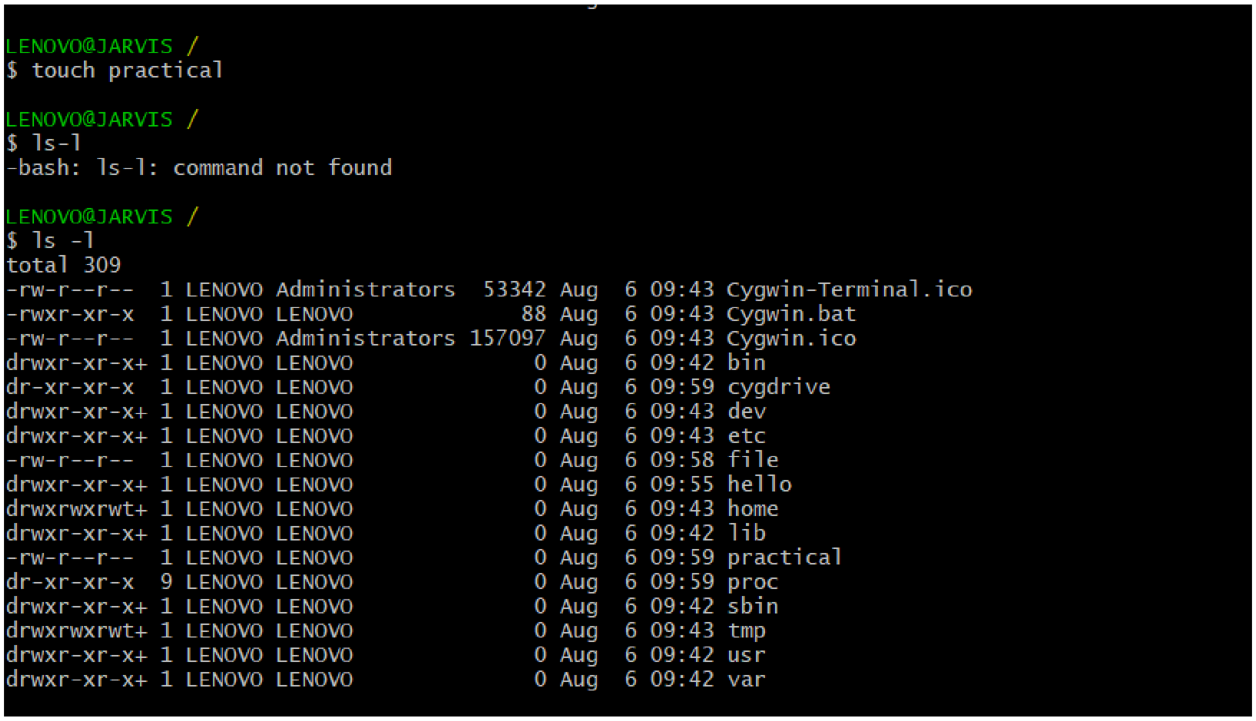
Q4. Grep is a Linux / Unix command-line tool used to search for a string of characters in a specified file. The text search pattern is called a regular expression. When it finds a match, it prints the line with the result.

Q5. A wildcard in Linux is a symbol or a set of symbols that stands in for other characters. It can be used to substitute for any other character or characters in a string. For example, you can use a wildcard to get a list of all files in a directory that begin with the letter O

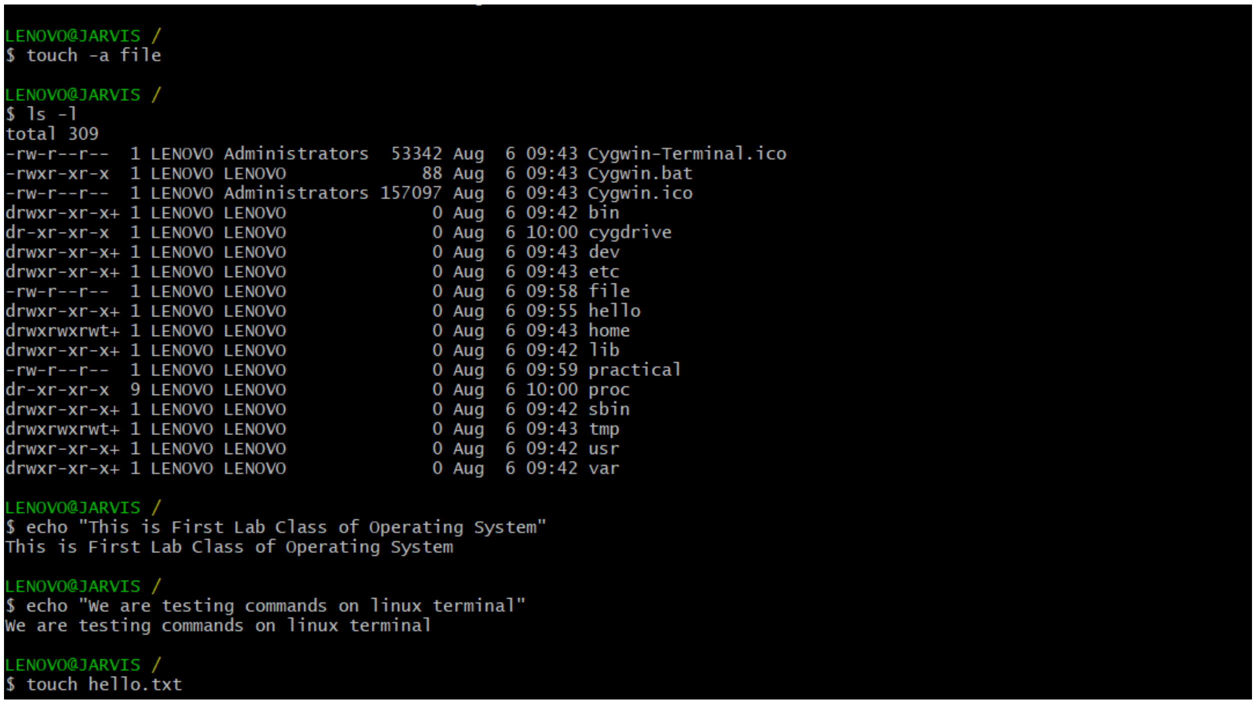
Screenshots



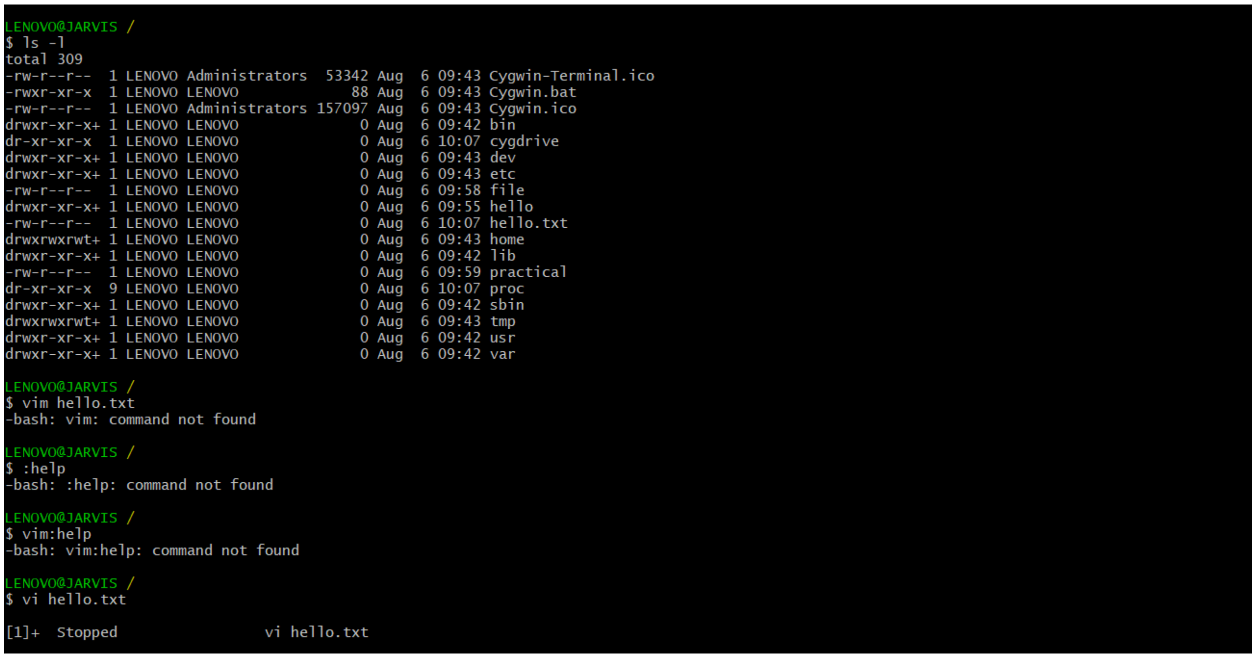
```
LENOVO@JARVIS ~
$ pwd
/home/LENOVO

LENOVO@JARVIS ~
$ cd ~

LENOVO@JARVIS ~
$ cd /

LENOVO@JARVIS /
$ ls
Cygwin-Terminal.ico  Cygwin.ico  cygdrive  etc   lib   sbin  usr
Cygwin.bat           bin         dev       home  proc  tmp   var

LENOVO@JARVIS /
$ mkdir hello

LENOVO@JARVIS /
$ ls
Cygwin-Terminal.ico  Cygwin.ico  cygdrive  etc    home  proc  tmp  var
Cygwin.bat           bin         dev       hello  lib   sbin  usr

LENOVO@JARVIS /
$ ls -a
.   Cygwin-Terminal.ico  Cygwin.ico  cygdrive  etc    home  proc  tmp  var
..  Cygwin.bat           bin         dev       hello  lib   sbin  usr

LENOVO@JARVIS /
$ ls -lh
total 309K
-rw-r--r--  1 LENOVO Administrators  53K Aug  6 09:43 Cygwin-Terminal.ico
-rwxr-xr-x  1 LENOVO LENOVO           88 Aug  6 09:43 Cygwin.bat
-rw-r--r--  1 LENOVO Administrators 154K Aug  6 09:43 Cygwin.ico
drwxr-xr-x+ 1 LENOVO LENOVO            0 Aug  6 09:42 bin
dr-xr-xr-x  1 LENOVO LENOVO            0 Aug  6 09:56 cygdrive
drwxr-xr-x+ 1 LENOVO LENOVO            0 Aug  6 09:43 dev
drwxr-xr-x+ 1 LENOVO LENOVO            0 Aug  6 09:43 etc
drwxr-xr-x+ 1 LENOVO LENOVO            0 Aug  6 09:55 hello
drwxrwxrwt+ 1 LENOVO LENOVO            0 Aug  6 09:43 home
drwxr-xr-x+ 1 LENOVO LENOVO            0 Aug  6 09:42 lib
dr-xr-xr-x  9 LENOVO LENOVO            0 Aug  6 09:56 proc
drwxr-xr-x+ 1 LENOVO LENOVO            0 Aug  6 09:42 sbin
drwxrwxrwt+ 1 LENOVO LENOVO            0 Aug  6 09:43 tmp
drwxr-xr-x+ 1 LENOVO LENOVO            0 Aug  6 09:42 usr
drwxr-xr-x+ 1 LENOVO LENOVO            0 Aug  6 09:42 var
```



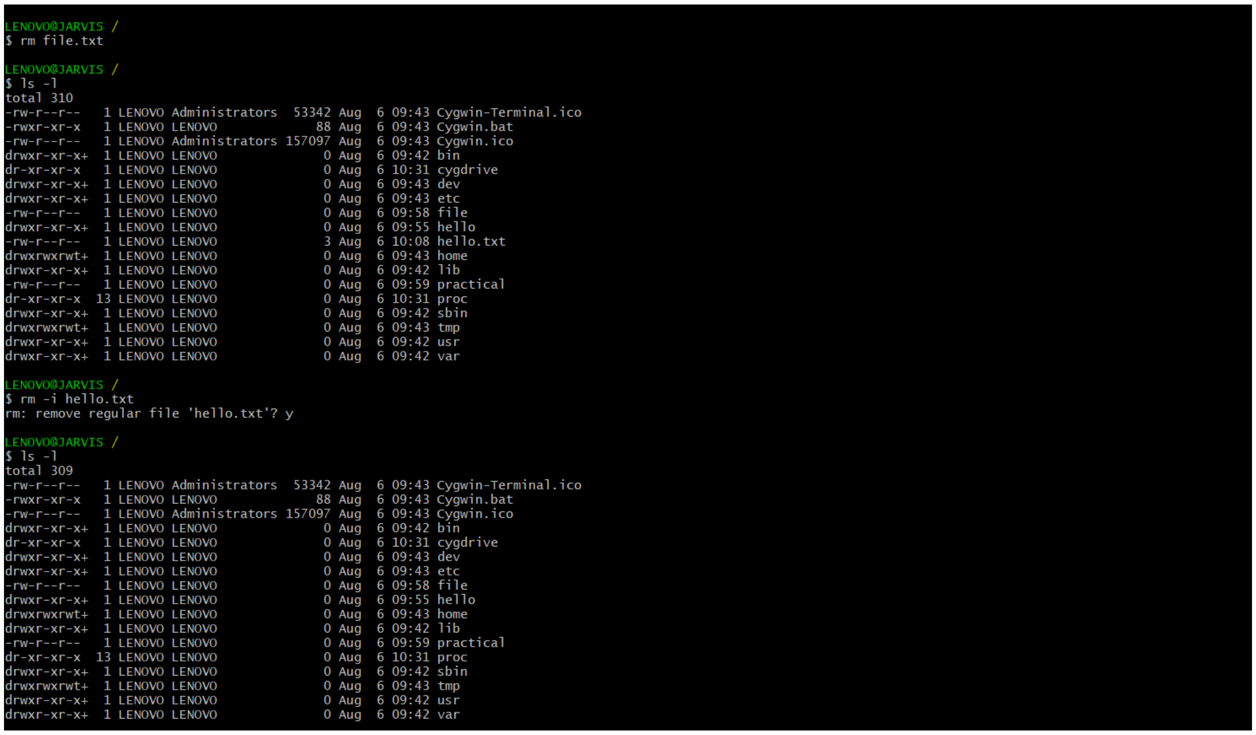
```
LENOVO@JARVIS /
$ touch practical

LENOVO@JARVIS /
$ ls-l
-bash: ls-l: command not found

LENOVO@JARVIS /
$ ls -l
total 309
-rw-r--r--  1 LENOVO Administrators  53342 Aug  6 09:43 Cygwin-Terminal.ico
-rwxr-xr-x  1 LENOVO LENOVO             88 Aug  6 09:43 Cygwin.bat
-rw-r--r--  1 LENOVO Administrators 157097 Aug  6 09:43 Cygwin.ico
drwxr-xr-x+ 1 LENOVO LENOVO              0 Aug  6 09:42 bin
dr-xr-xr-x  1 LENOVO LENOVO              0 Aug  6 09:59 cygdrive
drwxr-xr-x+ 1 LENOVO LENOVO              0 Aug  6 09:43 dev
drwxr-xr-x+ 1 LENOVO LENOVO              0 Aug  6 09:43 etc
-rw-r--r--  1 LENOVO LENOVO              0 Aug  6 09:58 file
drwxr-xr-x+ 1 LENOVO LENOVO              0 Aug  6 09:55 hello
drwxrwxrwt+ 1 LENOVO LENOVO              0 Aug  6 09:43 home
drwxr-xr-x+ 1 LENOVO LENOVO              0 Aug  6 09:42 lib
-rw-r--r--  1 LENOVO LENOVO              0 Aug  6 09:59 practical
dr-xr-xr-x  9 LENOVO LENOVO              0 Aug  6 09:59 proc
drwxr-xr-x+ 1 LENOVO LENOVO              0 Aug  6 09:42 sbin
drwxrwxrwt+ 1 LENOVO LENOVO              0 Aug  6 09:43 tmp
drwxr-xr-x+ 1 LENOVO LENOVO              0 Aug  6 09:42 usr
drwxr-xr-x+ 1 LENOVO LENOVO              0 Aug  6 09:42 var
```



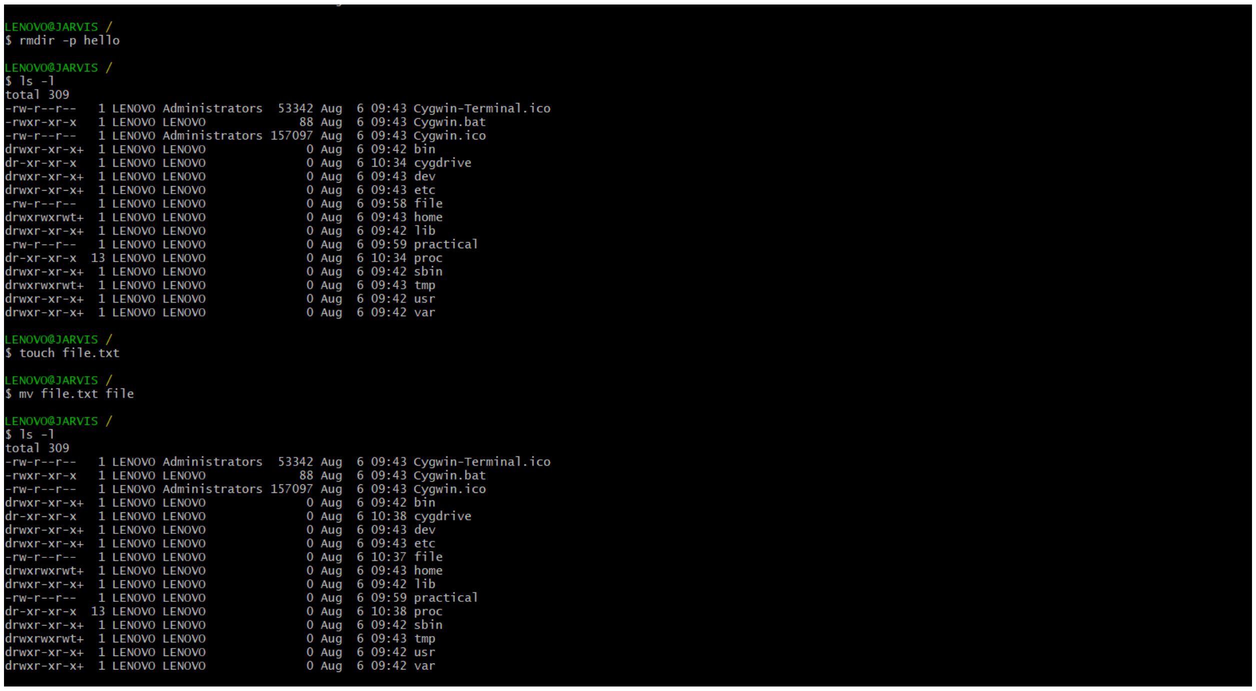
```
LENOVO@JARVIS /
$ touch -a file

LENOVO@JARVIS /
$ ls -l
total 309
-rw-r--r--  1 LENOVO Administrators  53342 Aug  6 09:43 Cygwin-Terminal.ico
-rwxr-xr-x  1 LENOVO LENOVO             88 Aug  6 09:43 Cygwin.bat
-rw-r--r--  1 LENOVO Administrators 157097 Aug  6 09:43 Cygwin.ico
drwxr-xr-x+ 1 LENOVO LENOVO              0 Aug  6 09:42 bin
dr-xr-xr-x  1 LENOVO LENOVO              0 Aug  6 10:00 cygdrive
drwxr-xr-x+ 1 LENOVO LENOVO              0 Aug  6 09:43 dev
drwxr-xr-x+ 1 LENOVO LENOVO              0 Aug  6 09:43 etc
-rw-r--r--  1 LENOVO LENOVO              0 Aug  6 09:58 file
drwxr-xr-x+ 1 LENOVO LENOVO              0 Aug  6 09:55 hello
drwxrwxrwt+ 1 LENOVO LENOVO              0 Aug  6 09:43 home
drwxr-xr-x+ 1 LENOVO LENOVO              0 Aug  6 09:42 lib
-rw-r--r--  1 LENOVO LENOVO              0 Aug  6 09:59 practical
dr-xr-xr-x  9 LENOVO LENOVO              0 Aug  6 10:00 proc
drwxr-xr-x+ 1 LENOVO LENOVO              0 Aug  6 09:42 sbin
drwxrwxrwt+ 1 LENOVO LENOVO              0 Aug  6 09:43 tmp
drwxr-xr-x+ 1 LENOVO LENOVO              0 Aug  6 09:42 usr
drwxr-xr-x+ 1 LENOVO LENOVO              0 Aug  6 09:42 var

LENOVO@JARVIS /
$ echo "This is First Lab Class of Operating System"
This is First Lab Class of Operating System

LENOVO@JARVIS /
$ echo "We are testing commands on linux terminal"
We are testing commands on linux terminal

LENOVO@JARVIS /
$ touch hello.txt
```



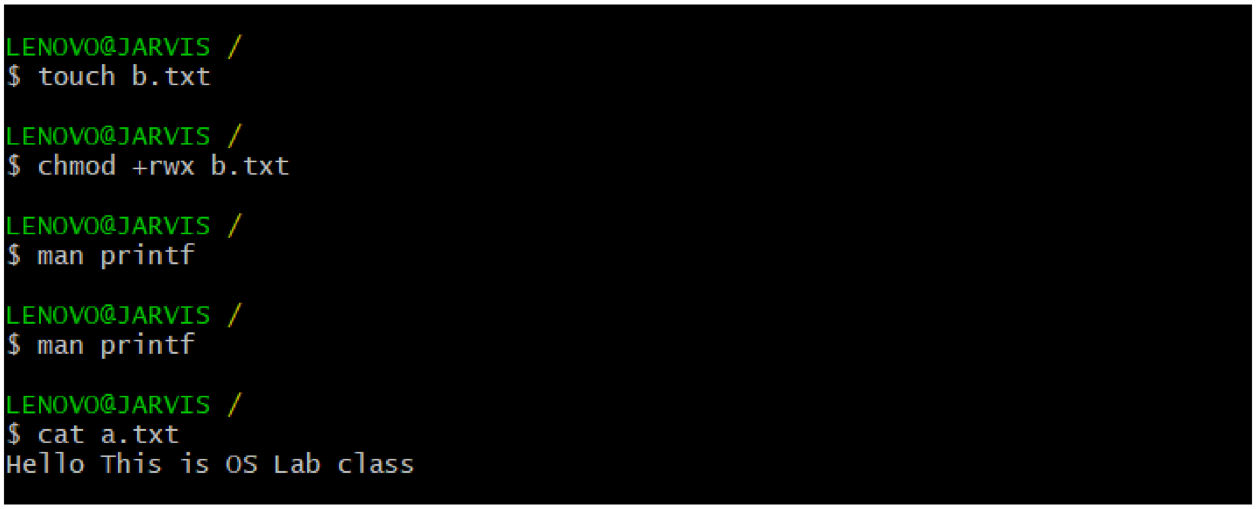
```
LENOVO@JARVIS /
$ ls -l
total 309
-rw-r--r--  1 LENOVO Administrators  53342 Aug  6 09:43 Cygwin-Terminal.ico
-rwxr-xr-x  1 LENOVO LENOVO             88 Aug  6 09:43 Cygwin.bat
-rw-r--r--  1 LENOVO Administrators 157097 Aug  6 09:43 Cygwin.ico
drwxr-xr-x+ 1 LENOVO LENOVO              0 Aug  6 09:42 bin
dr-xr-xr-x  1 LENOVO LENOVO              0 Aug  6 10:07 cygdrive
drwxr-xr-x+ 1 LENOVO LENOVO              0 Aug  6 09:43 dev
drwxr-xr-x+ 1 LENOVO LENOVO              0 Aug  6 09:43 etc
-rw-r--r--  1 LENOVO LENOVO              0 Aug  6 09:58 file
drwxr-xr-x+ 1 LENOVO LENOVO              0 Aug  6 09:55 hello
-rw-r--r--  1 LENOVO LENOVO              0 Aug  6 10:07 hello.txt
drwxrwxrwt+ 1 LENOVO LENOVO              0 Aug  6 09:43 home
drwxr-xr-x+ 1 LENOVO LENOVO              0 Aug  6 09:42 lib
-rw-r--r--  1 LENOVO LENOVO              0 Aug  6 09:59 practical
dr-xr-xr-x  9 LENOVO LENOVO              0 Aug  6 10:07 proc
drwxr-xr-x+ 1 LENOVO LENOVO              0 Aug  6 09:42 sbin
drwxrwxrwt+ 1 LENOVO LENOVO              0 Aug  6 09:43 tmp
drwxr-xr-x+ 1 LENOVO LENOVO              0 Aug  6 09:42 usr
drwxr-xr-x+ 1 LENOVO LENOVO              0 Aug  6 09:42 var

LENOVO@JARVIS /
$ vim hello.txt
-bash: vim: command not found

LENOVO@JARVIS /
$ :help
-bash: :help: command not found

LENOVO@JARVIS /
$ vim:help
-bash: vim:help: command not found

LENOVO@JARVIS /
$ vi hello.txt

[1]+  Stopped                 vi hello.txt
```



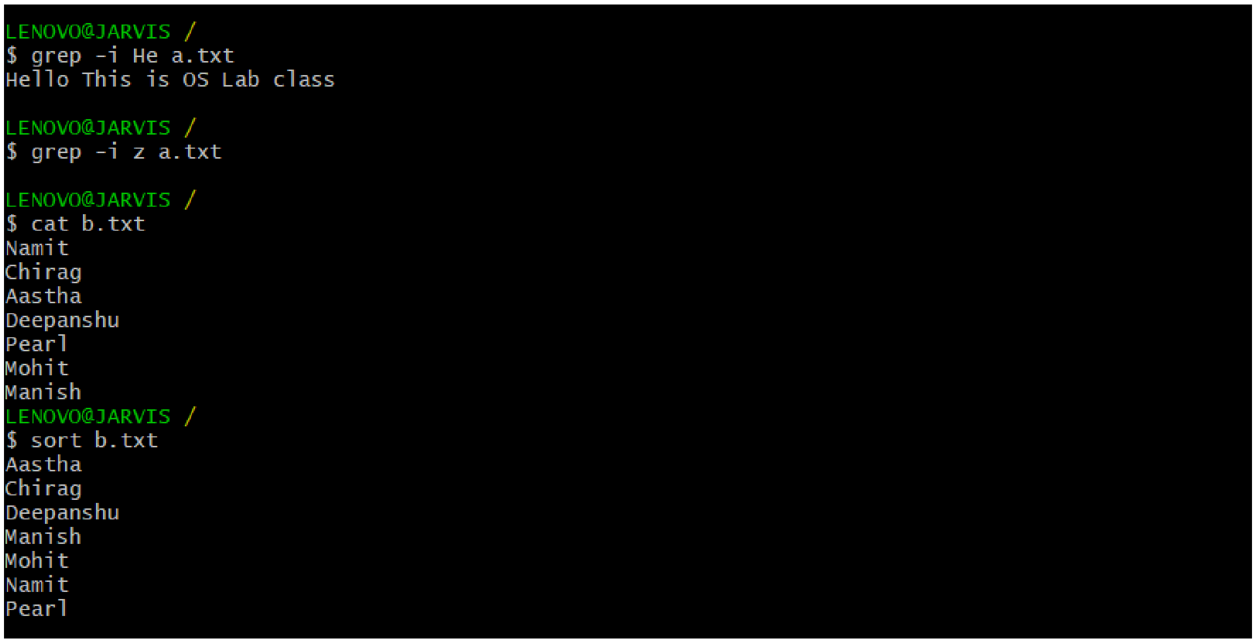
```
LENOVO@JARVIS /
$ rm file.txt

LENOVO@JARVIS /
$ ls -l
total 310
-rw-r--r--   1 LENOVO Administrators  53342 Aug  6 09:43 Cygwin-Terminal.ico
-rwxr-xr-x   1 LENOVO LENOVO             88 Aug  6 09:43 Cygwin.bat
-rw-r--r--   1 LENOVO Administrators 157097 Aug  6 09:43 Cygwin.ico
drwxr-xr-x+  1 LENOVO LENOVO              0 Aug  6 09:42 bin
dr-xr-xr-x   1 LENOVO LENOVO              0 Aug  6 10:31 cygdrive
drwxr-xr-x+  1 LENOVO LENOVO              0 Aug  6 09:43 dev
drwxr-xr-x+  1 LENOVO LENOVO              0 Aug  6 09:43 etc
-rw-r--r--   1 LENOVO LENOVO              0 Aug  6 09:58 file
drwxr-xr-x+  1 LENOVO LENOVO              0 Aug  6 09:55 hello
-rw-r--r--   1 LENOVO LENOVO              3 Aug  6 10:08 hello.txt
drwxrwxrwt+  1 LENOVO LENOVO              0 Aug  6 09:43 home
drwxr-xr-x+  1 LENOVO LENOVO              0 Aug  6 09:42 lib
-rw-r--r--   1 LENOVO LENOVO              0 Aug  6 09:59 practical
dr-xr-xr-x  13 LENOVO LENOVO              0 Aug  6 10:31 proc
drwxr-xr-x+  1 LENOVO LENOVO              0 Aug  6 09:42 sbin
drwxrwxrwt+  1 LENOVO LENOVO              0 Aug  6 09:43 tmp
drwxr-xr-x+  1 LENOVO LENOVO              0 Aug  6 09:42 usr
drwxr-xr-x+  1 LENOVO LENOVO              0 Aug  6 09:42 var

LENOVO@JARVIS /
$ rm -i hello.txt
rm: remove regular file 'hello.txt'? y

LENOVO@JARVIS /
$ ls -l
total 309
-rw-r--r--   1 LENOVO Administrators  53342 Aug  6 09:43 Cygwin-Terminal.ico
-rwxr-xr-x   1 LENOVO LENOVO             88 Aug  6 09:43 Cygwin.bat
-rw-r--r--   1 LENOVO Administrators 157097 Aug  6 09:43 Cygwin.ico
drwxr-xr-x+  1 LENOVO LENOVO              0 Aug  6 09:42 bin
dr-xr-xr-x   1 LENOVO LENOVO              0 Aug  6 10:31 cygdrive
drwxr-xr-x+  1 LENOVO LENOVO              0 Aug  6 09:43 dev
drwxr-xr-x+  1 LENOVO LENOVO              0 Aug  6 09:43 etc
-rw-r--r--   1 LENOVO LENOVO              0 Aug  6 09:58 file
drwxr-xr-x+  1 LENOVO LENOVO              0 Aug  6 09:55 hello
drwxrwxrwt+  1 LENOVO LENOVO              0 Aug  6 09:43 home
drwxr-xr-x+  1 LENOVO LENOVO              0 Aug  6 09:42 lib
-rw-r--r--   1 LENOVO LENOVO              0 Aug  6 09:59 practical
dr-xr-xr-x  13 LENOVO LENOVO              0 Aug  6 10:31 proc
drwxr-xr-x+  1 LENOVO LENOVO              0 Aug  6 09:42 sbin
drwxrwxrwt+  1 LENOVO LENOVO              0 Aug  6 09:43 tmp
drwxr-xr-x+  1 LENOVO LENOVO              0 Aug  6 09:42 usr
drwxr-xr-x+  1 LENOVO LENOVO              0 Aug  6 09:42 var
```



```
LENOVO@JARVIS /
$ rmdir -p hello

LENOVO@JARVIS /
$ ls -l
total 309
-rw-r--r--   1 LENOVO Administrators  53342 Aug  6 09:43 Cygwin-Terminal.ico
-rwxr-xr-x   1 LENOVO LENOVO             88 Aug  6 09:43 Cygwin.bat
-rw-r--r--   1 LENOVO Administrators 157097 Aug  6 09:43 Cygwin.ico
drwxr-xr-x+  1 LENOVO LENOVO              0 Aug  6 09:42 bin
dr-xr-xr-x   1 LENOVO LENOVO              0 Aug  6 10:34 cygdrive
drwxr-xr-x+  1 LENOVO LENOVO              0 Aug  6 09:43 dev
drwxr-xr-x+  1 LENOVO LENOVO              0 Aug  6 09:43 etc
-rw-r--r--   1 LENOVO LENOVO              0 Aug  6 09:58 file
drwxrwxrwt+  1 LENOVO LENOVO              0 Aug  6 09:43 home
drwxr-xr-x+  1 LENOVO LENOVO              0 Aug  6 09:42 lib
-rw-r--r--   1 LENOVO LENOVO              0 Aug  6 09:59 practical
dr-xr-xr-x  13 LENOVO LENOVO              0 Aug  6 10:34 proc
drwxr-xr-x+  1 LENOVO LENOVO              0 Aug  6 09:42 sbin
drwxrwxrwt+  1 LENOVO LENOVO              0 Aug  6 09:43 tmp
drwxr-xr-x+  1 LENOVO LENOVO              0 Aug  6 09:42 usr
drwxr-xr-x+  1 LENOVO LENOVO              0 Aug  6 09:42 var

LENOVO@JARVIS /
$ touch file.txt

LENOVO@JARVIS /
$ mv file.txt file

LENOVO@JARVIS /
$ ls -l
total 309
-rw-r--r--   1 LENOVO Administrators  53342 Aug  6 09:43 Cygwin-Terminal.ico
-rwxr-xr-x   1 LENOVO LENOVO             88 Aug  6 09:43 Cygwin.bat
-rw-r--r--   1 LENOVO Administrators 157097 Aug  6 09:43 Cygwin.ico
drwxr-xr-x+  1 LENOVO LENOVO              0 Aug  6 09:42 bin
dr-xr-xr-x   1 LENOVO LENOVO              0 Aug  6 10:38 cygdrive
drwxr-xr-x+  1 LENOVO LENOVO              0 Aug  6 09:43 dev
drwxr-xr-x+  1 LENOVO LENOVO              0 Aug  6 09:43 etc
-rw-r--r--   1 LENOVO LENOVO              0 Aug  6 10:37 file
drwxrwxrwt+  1 LENOVO LENOVO              0 Aug  6 09:43 home
drwxr-xr-x+  1 LENOVO LENOVO              0 Aug  6 09:42 lib
-rw-r--r--   1 LENOVO LENOVO              0 Aug  6 09:59 practical
dr-xr-xr-x  13 LENOVO LENOVO              0 Aug  6 10:38 proc
drwxr-xr-x+  1 LENOVO LENOVO              0 Aug  6 09:42 sbin
drwxrwxrwt+  1 LENOVO LENOVO              0 Aug  6 09:43 tmp
drwxr-xr-x+  1 LENOVO LENOVO              0 Aug  6 09:42 usr
drwxr-xr-x+  1 LENOVO LENOVO              0 Aug  6 09:42 var
```

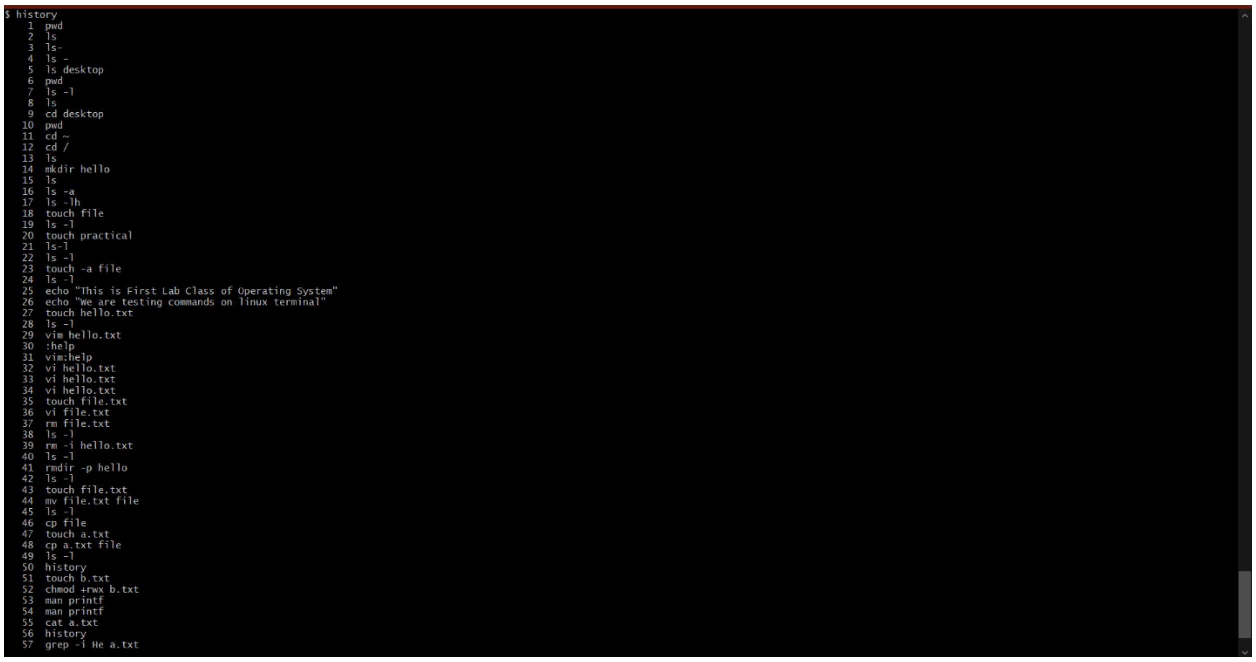


```
LENOVO@JARVIS /
$ touch b.txt

LENOVO@JARVIS /
$ chmod +rwx b.txt

LENOVO@JARVIS /
$ man printf

LENOVO@JARVIS /
$ man printf

LENOVO@JARVIS /
$ cat a.txt
Hello This is OS Lab class
```



```
LENOVO@JARVIS /
$ grep -i He a.txt
Hello This is OS Lab class

LENOVO@JARVIS /
$ grep -i z a.txt

LENOVO@JARVIS /
$ cat b.txt
Namit
Chirag
Aastha
Deepanshu
Pearl
Mohit
Manish
LENOVO@JARVIS /
$ sort b.txt
Aastha
Chirag
Deepanshu
Manish
Mohit
Namit
Pearl
```

History Command to show all the commands used :-



```
$ history
    1  pwd
    2  ls
    3  ls-
    4  ls -
    5  ls desktop
    6  pwd
    7  ls -l
    8  ls
    9  cd desktop
   10  pwd
   11  cd ~
   12  cd /
   13  ls
   14  mkdir hello
   15  ls
   16  ls -a
   17  ls -lh
   18  touch file
   19  ls -l
   20  touch practical
   21  ls-l
   22  ls -l
   23  touch -a file
   24  ls -l
   25  echo "This is First Lab Class of Operating System"
   26  echo "We are testing commands on linux terminal"
   27  touch hello.txt
   28  ls -l
   29  vim hello.txt
   30  :help
   31  vim:help
   32  vi hello.txt
   33  vi hello.txt
   34  vi hello.txt
   35  touch file.txt
   36  vi file.txt
   37  rm file.txt
   38  ls -l
   39  rm -i hello.txt
   40  ls -l
   41  rmdir -p hello
   42  ls -l
   43  touch file.txt
   44  mv file.txt file
   45  ls -l
   46  cp file
   47  touch a.txt
   48  cp a.txt file
   49  ls -l
   50  history
   51  touch b.txt
   52  chmod +rwx b.txt
   53  man printf
   54  man printf
   55  cat a.txt
   56  history
   57  grep -i He a.txt
```

EXPERIMENT NO. 2

Student Name and Roll Number: Namit Kumar 19CSU185
Semester /Section: V/FSA1
Link to Code: https://github.com/NamitKumar16/OS
Date: 10th August 2021
Faculty Signature:
Marks:

Objective:
To write the shell programming code for the following.

Outcome:
Student is able to write code in shell programming

Problem Statement:
a) Write A Shell Program of Hello World

b) Write a shell program to find factorial of a number.

c) Write a shell program to find gross salary of an employee.

d) Write a shell program to display the menu and execute instructions accordingly

(i) List of files **(ii)**Process Status **(iii)** Date **(iv)** users in program **(v)** Quit

Background Study:

A shell script is a file with a set of commands in it. The shell reads this file and executes the instructions as if they were input directly on the command line.

A shell is a command-line interpreter and operations such as file manipulation, program execution and text printing is performed by shell script. So, we will use vi editor to edit our files.

Question Bank:

1. What is a shell?
2. What is the significance of $#?
3. What are the different types of commonly used shells on a typical Linux system?
4. How will you pass and access arguments to a script in Linux?
5. Use sed command to replace the content of the file (emulate tac command)

Student Work Area

Algorithm/Flowchart/Code/Sample Outputs

Algorithm/Flowchart/Code/Sample Outputs

Q1 Shell is a program that takes commands from the keyboard and gives them to the operating system to perform.

Q2 $# shows the count of the arguments passed to the script.

Q3 Bash, Zsh, Korn, Tcsh, Fish are commonly used shells

Q4 Arguments can be passed to the script when it is executed, by writing them as a space-delimited list following the script file name.

Inside the script, the $1 variable references the first argument in the command line, $2 the second argument and so forth. The variable $0 references to the current script.

Q5 sed ‘1! G; h;$!d’ file1

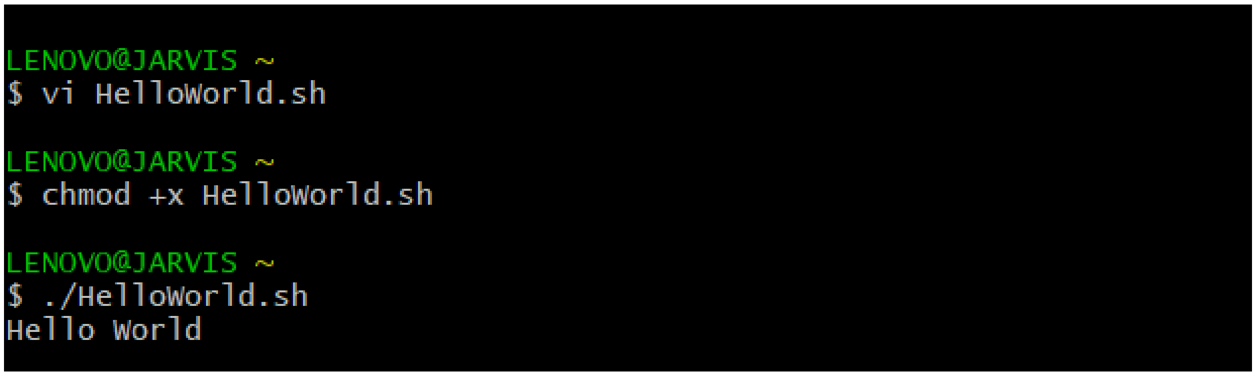
Here G command appends to the pattern space,

h command copies pattern buffer to hold buffer

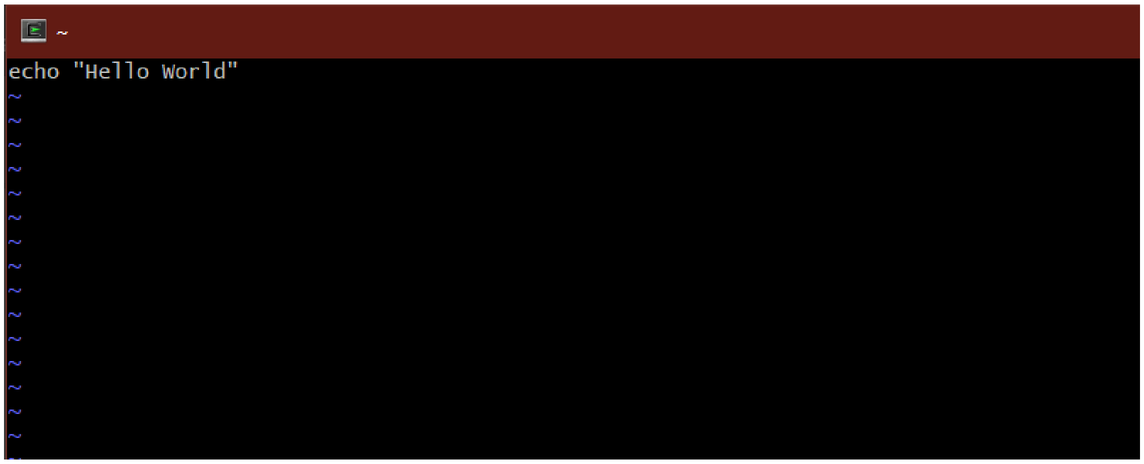
and d command deletes the current pattern space.

Screenshots

a) Write A Shell Program of Hello World

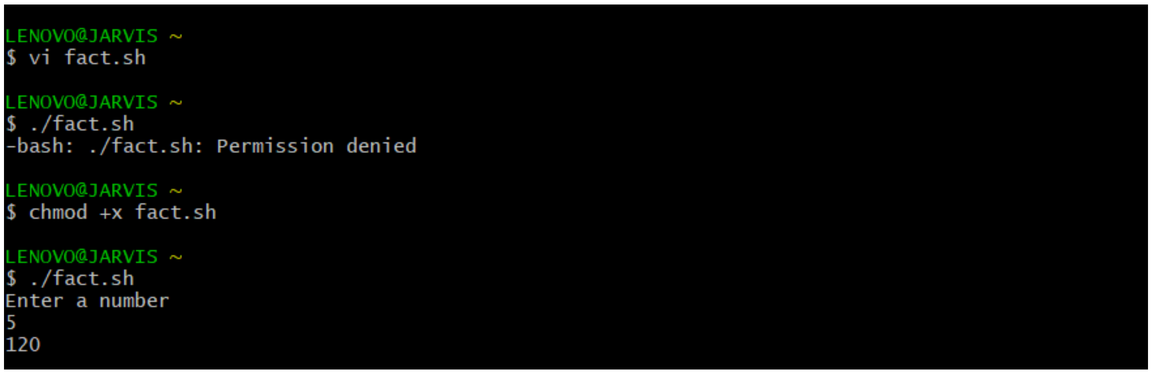


```
LENOVO@JARVIS ~
$ vi HelloWorld.sh

LENOVO@JARVIS ~
$ chmod +x HelloWorld.sh

LENOVO@JARVIS ~
$ ./HelloWorld.sh
Hello World
```

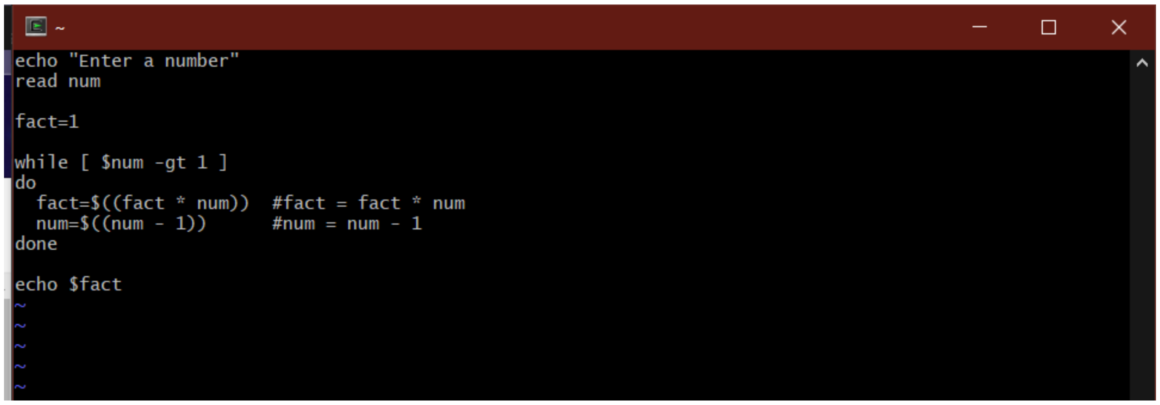


```
echo "Hello World"
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
```

b) Write a shell program to find factorial of a number.

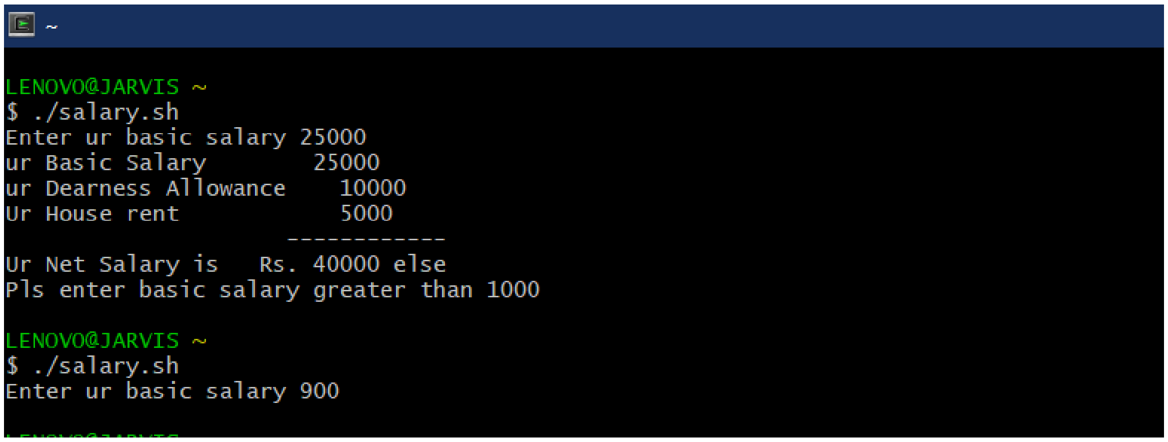


```
LENOVO@JARVIS ~
$ vi fact.sh

LENOVO@JARVIS ~
$ ./fact.sh
-bash: ./fact.sh: Permission denied

LENOVO@JARVIS ~
$ chmod +x fact.sh

LENOVO@JARVIS ~
$ ./fact.sh
Enter a number
5
120
```

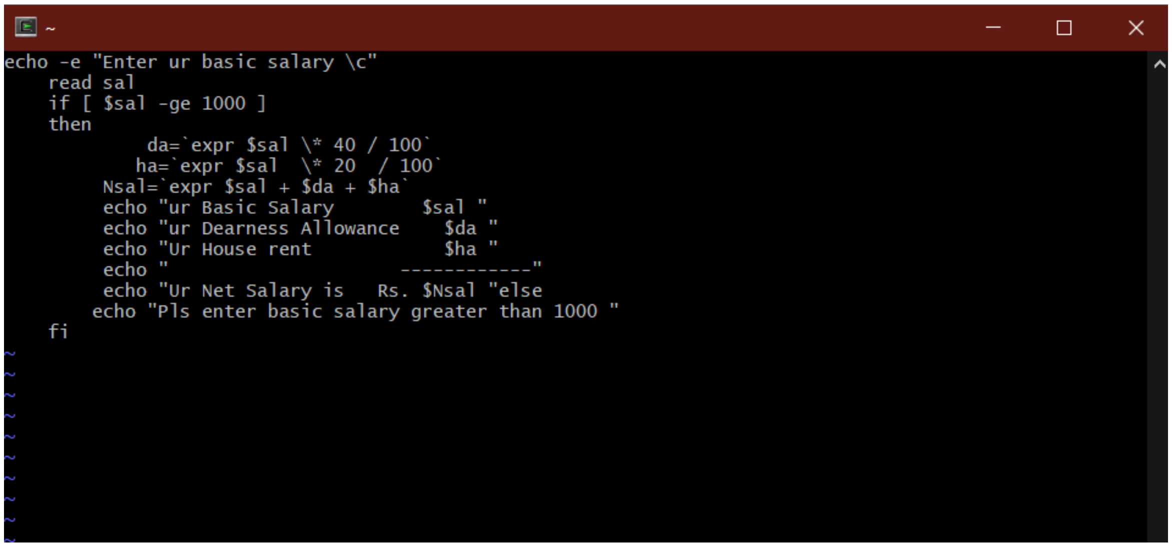


```
echo "Enter a number"
read num

fact=1

while [ $num -gt 1 ]
do
  fact=$((fact * num))  #fact = fact * num
  num=$((num - 1))      #num = num - 1
done

echo $fact
~
~
~
~
~
```

c) Write a shell program to find gross salary of an employee.

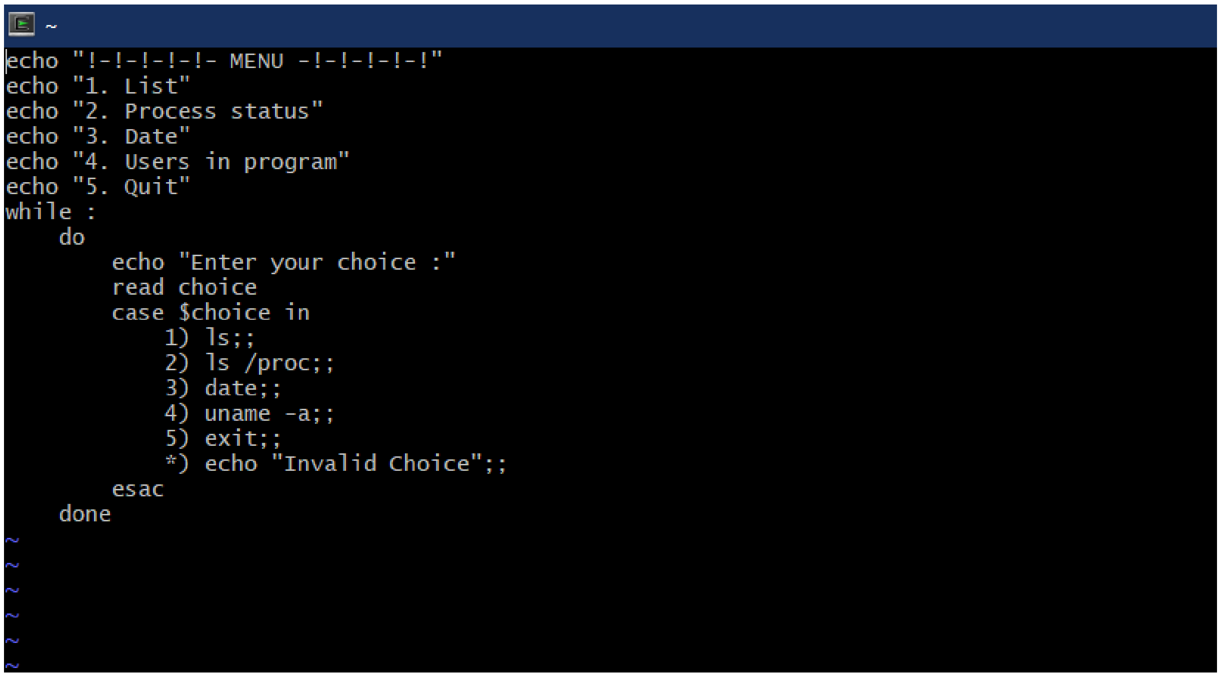


```
LENOVO@JARVIS ~
$ ./salary.sh
Enter ur basic salary 25000
ur Basic Salary         25000
ur Dearness Allowance     10000
Ur House rent             5000
                      ------------
Ur Net Salary is    Rs. 40000 else
Pls enter basic salary greater than 1000

LENOVO@JARVIS ~
$ ./salary.sh
Enter ur basic salary 900
```

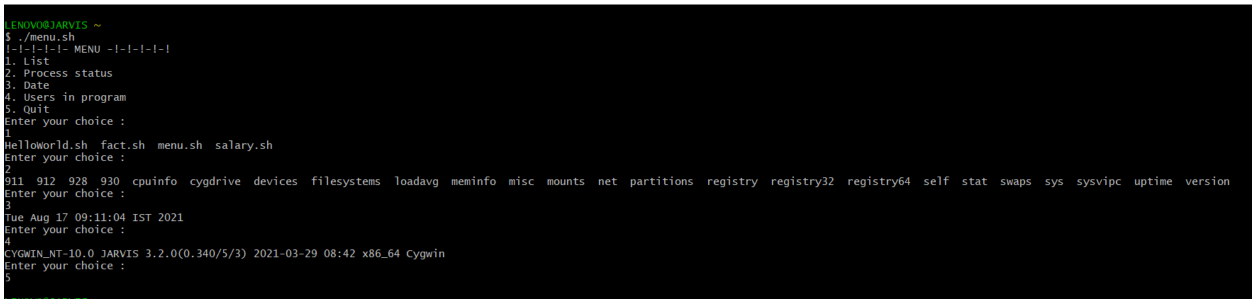


```
echo -e "Enter ur basic salary \c"
    read sal
    if [ $sal -ge 1000 ]
    then
             da=`expr $sal \* 40 / 100`
            ha=`expr $sal  \* 20  / 100`
         Nsal=`expr $sal + $da + $ha`
         echo "ur Basic Salary        $sal "
         echo "ur Dearness Allowance    $da "
         echo "Ur House rent            $ha "
         echo "                 ------------"
         echo "Ur Net Salary is   Rs. $Nsal "else
        echo "Pls enter basic salary greater than 1000 "
    fi
~
~
~
~
~
~
~
~
~
~
```

d) Write a shell program to display the menu and execute instructions accordingly **(i)**List of files **(ii)**Process Status **(iii)** Date **(iv)** users in program **(v)** Quit



```
echo "!-!-!-!-!- MENU -!-!-!-!-!"
echo "1. List"
echo "2. Process status"
echo "3. Date"
echo "4. Users in program"
echo "5. Quit"
while :
    do
        echo "Enter your choice :"
        read choice
        case $choice in
            1) ls;;
            2) ls /proc;;
            3) date;;
            4) uname -a;;
            5) exit;;
            *) echo "Invalid Choice";;
        esac
    done
~
~
~
~
~
~
```



```
LENOVO@JARVIS ~
$ ./menu.sh
!-!-!-!-!- MENU -!-!-!-!-!
1. List
2. Process status
3. Date
4. Users in program
5. Quit
Enter your choice :
1
HelloWorld.sh  fact.sh  menu.sh  salary.sh
Enter your choice :
2
911  912  928  930  cpuinfo  cygdrive  devices  filesystems  loadavg  meminfo  misc  mounts  net  partitions  registry  registry32  registry64  self  stat  swaps  sys  sysvipc  uptime  version
Enter your choice :
3
Tue Aug 17 09:11:04 IST 2021
Enter your choice :
4
CYGWIN_NT-10.0 JARVIS 3.2.0(0.340/5/3) 2021-03-29 08:42 x86_64 Cygwin
Enter your choice :
5
```

EXPERIMENT NO. 3

Student Name and Roll Number: Namit Kumar
Semester /Section: V/FS-A-1
Link to Code: https://github.com/NamitKumar16/OS
Date: 17th August 2021
Faculty Signature:
Marks:

Objective:
To write the shell programming code for the following.

Outcome:
Student is able to write code in shell programming

Problem Statement:
a) Write a shell program to find Fibonacci series.

b) Write a shell program to find largest of three numbers.

c) Write a shell program to find average of N numbers

Background Study:

A shell script is a file with a set of commands in it. The shell reads this file and executes the instructions as if they were input directly on the command line.

A shell is a command-line interpreter and operations such as file manipulation, program execution and text printing is performed by shell script. So, we will use vi editor to edit our files.

Question Bank:

1. How to use multi line comments in shell script?
2. What is the difference between soft and hard links?
3. Explain about loops and what are the loops available in LINUX?
4. What are absolute and relative paths.
5. How to debug a shell script.

Student Work Area

Algorithm/Flowchart/Code/Sample Outputs

Q1 Ans - In shell or bash shell, we can comment on multiple lines using << and name of comment.

Q2 Ans - Hard Links cannot be used across file systems whereas soft links can be used across file systems.

Q3 Ans - The while loop

The for loop

The until loop

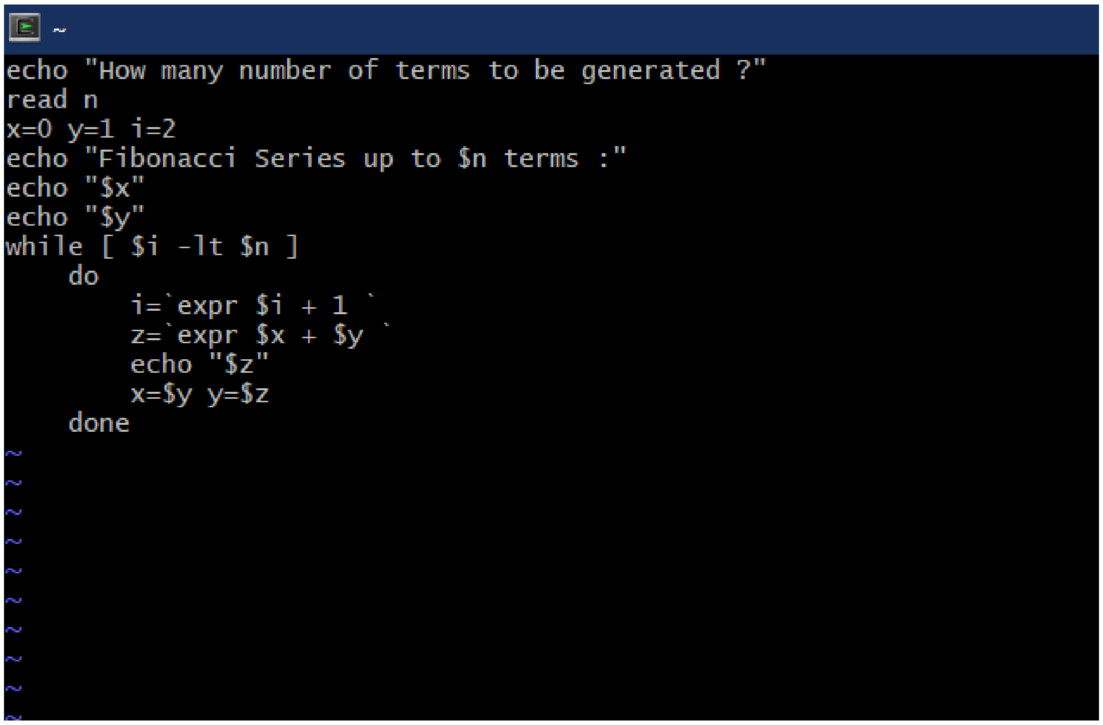
The select loop

Q4 Ans - An absolute path always contains the root element and the complete directory list required to

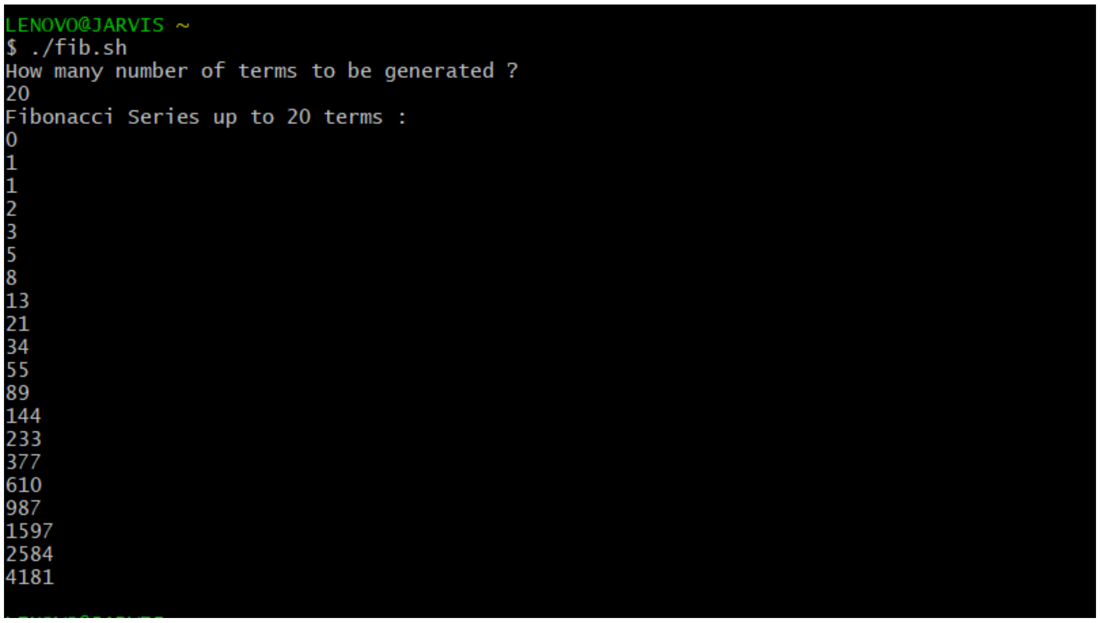
locate the file.

A relative path needs to be combined with another path in order to access a file.

a) Write a shell program to find largest of three numbers

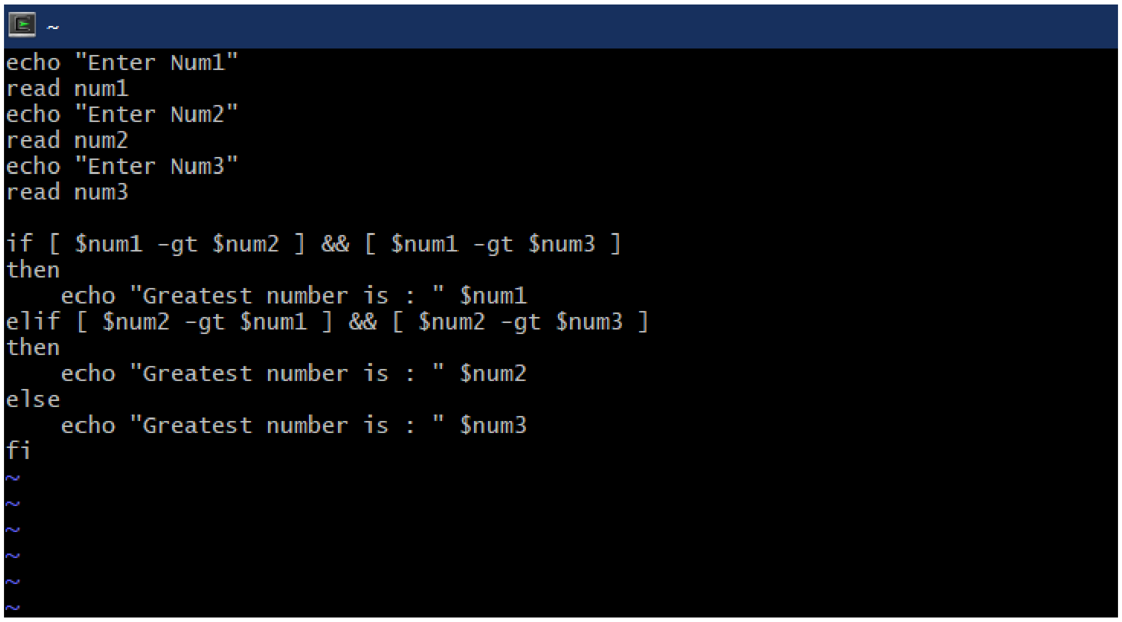


```
echo "How many number of terms to be generated ?"
read n
x=0 y=1 i=2
echo "Fibonacci Series up to $n terms :"
echo "$x"
echo "$y"
while [ $i -lt $n ]
    do
        i=`expr $i + 1 `
        z=`expr $x + $y `
        echo "$z"
        x=$y y=$z
    done
~
~
~
~
~
~
~
~
~
```

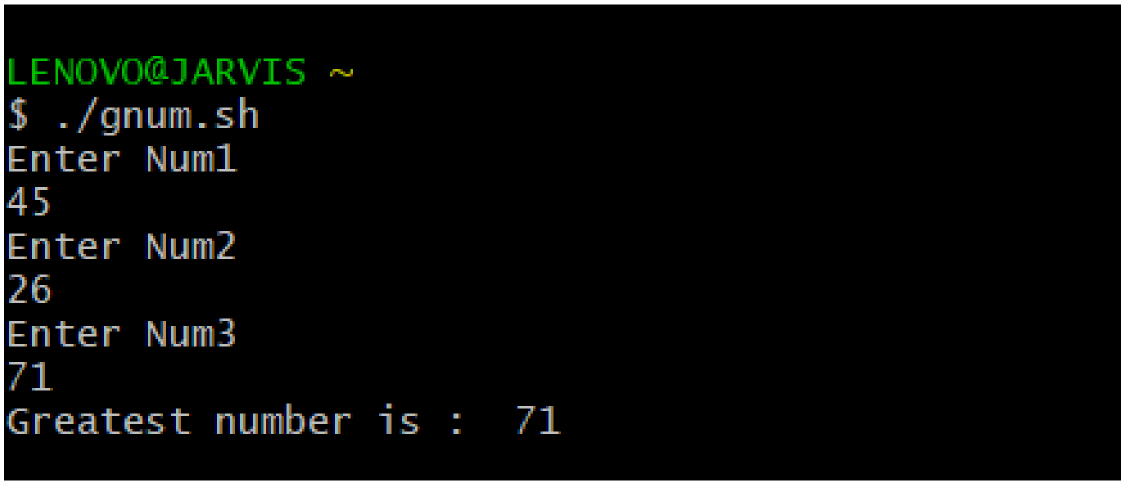


```
LENOVO@JARVIS ~
$ ./fib.sh
How many number of terms to be generated ?
20
Fibonacci Series up to 20 terms :
0
1
1
2
3
5
8
13
21
34
55
89
144
233
377
610
987
1597
2584
4181
```

b) Write a shell program to find average of N numbers

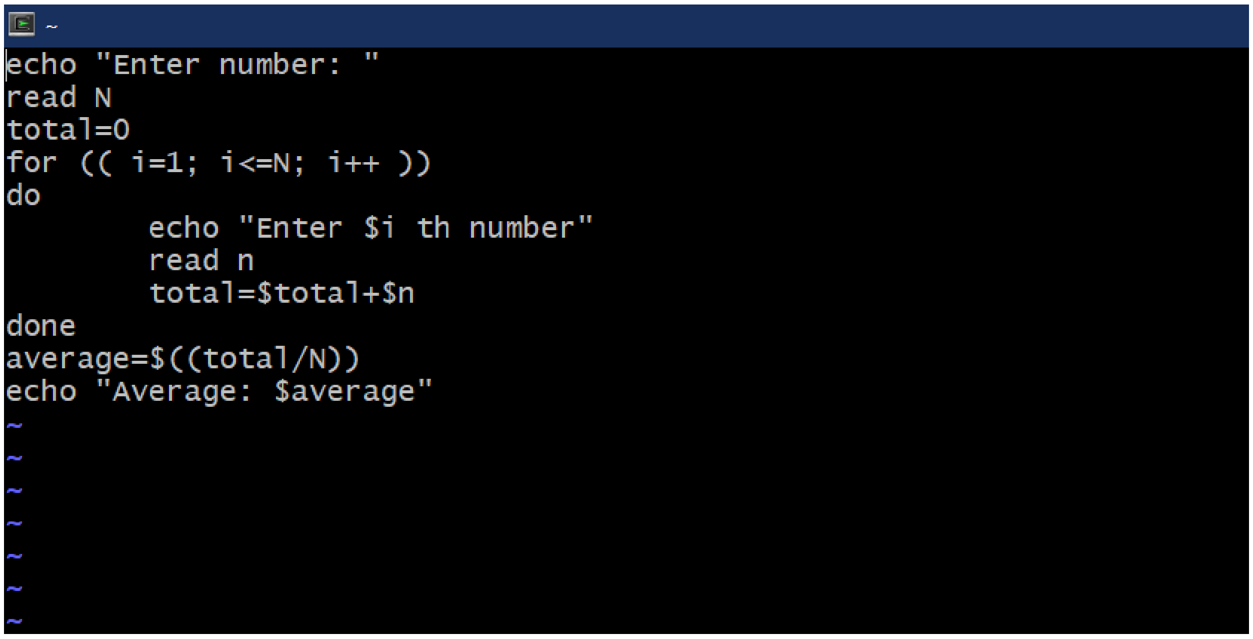


```
echo "Enter Num1"
read num1
echo "Enter Num2"
read num2
echo "Enter Num3"
read num3

if [ $num1 -gt $num2 ] && [ $num1 -gt $num3 ]
then
    echo "Greatest number is : " $num1
elif [ $num2 -gt $num1 ] && [ $num2 -gt $num3 ]
then
    echo "Greatest number is : " $num2
else
    echo "Greatest number is : " $num3
fi
~
~
~
~
~
~
```

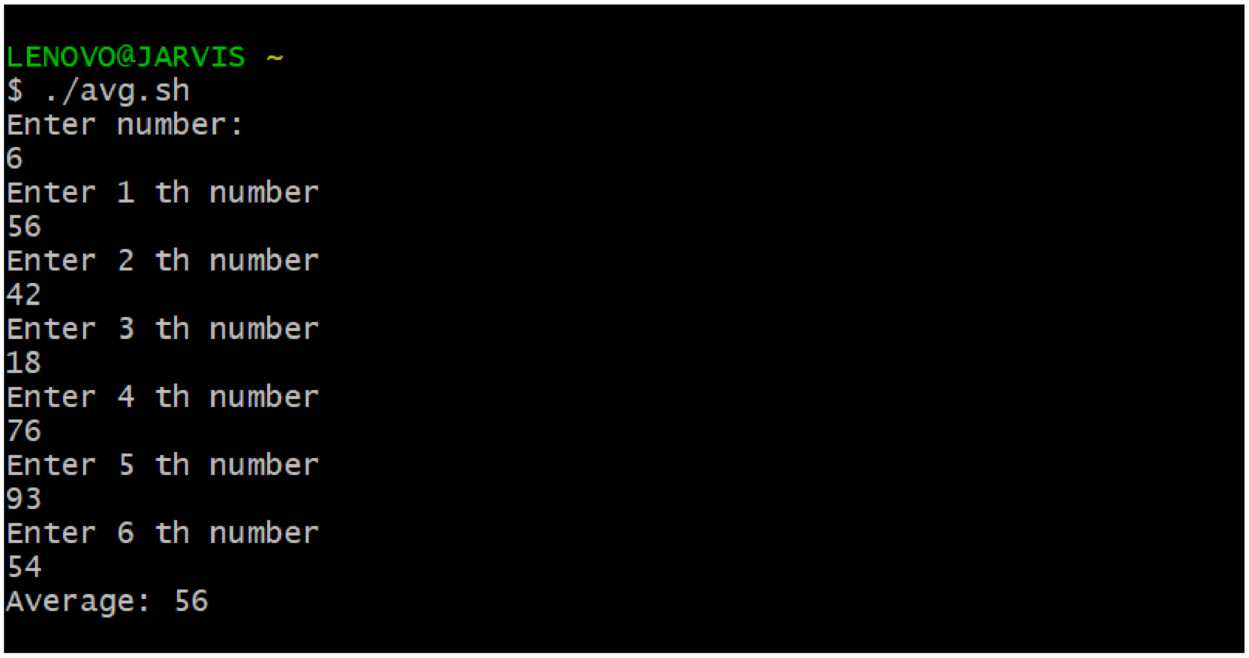


```
LENOVO@JARVIS ~
$ ./gnum.sh
Enter Num1
45
Enter Num2
26
Enter Num3
71
Greatest number is :  71
```

c) Find average of 3 numbers



```
~
echo "Enter number: "
read N
total=0
for (( i=1; i<=N; i++ ))
do
        echo "Enter $i th number"
        read n
        total=$total+$n
done
average=$((total/N))
echo "Average: $average"
~
~
~
~
~
~
~
```



```
LENOVO@JARVIS ~
$ ./avg.sh
Enter number:
6
Enter 1 th number
56
Enter 2 th number
42
Enter 3 th number
18
Enter 4 th number
76
Enter 5 th number
93
Enter 6 th number
54
Average: 56
```

EXPERIMENT NO. 4

Student Name and Roll Number: Namit Kumar
Semester /Section: V/FS-A-1
Link to Code: https://github.com/NamitKumar16/OS
Date: 8th September
Faculty Signature:
Marks:

Objective:
To write the shell programming code for the following.

Outcome:
Student is able to write code in shell programming

Problem Statement:
a) Write a shell program to check whether a number is even or odd

b) Write a shell program to find whether a number is prime or not.

c) Write a shell program to find whether a number is palindrome or not.

d) Write a shell program to type number 1 to 7 and then print its corresponding day of week

Background Study:

A shell script is a file with a set of commands in it. The shell reads this file and executes the instructions as if they were input directly on the command line.

A shell is a command-line interpreter and operations such as file manipulation, program execution and text printing is performed by shell script. So, we will use vi editor to edit our files.

Question Bank:

1. What are Zoombie Process?
2. What are different types of variables used in shell script?
3. What are the different types of modes available in Vi editor?
4. What are the different types of permission at file level in shell?

5. How to use comments in shell script.

Student Work Area

Algorithm/Flowchart/Code/Sample output

Q1 Ans - A zombie process is a process whose execution is completed but it still has an entry in the

process table.

Q2 Ans - System-Defined Variables.

User-Defined Variables.

Q3 Ans - Command Mode , Entry Mode And Last Line Mode

Q4 Ans - Execute Permission

Write Permission

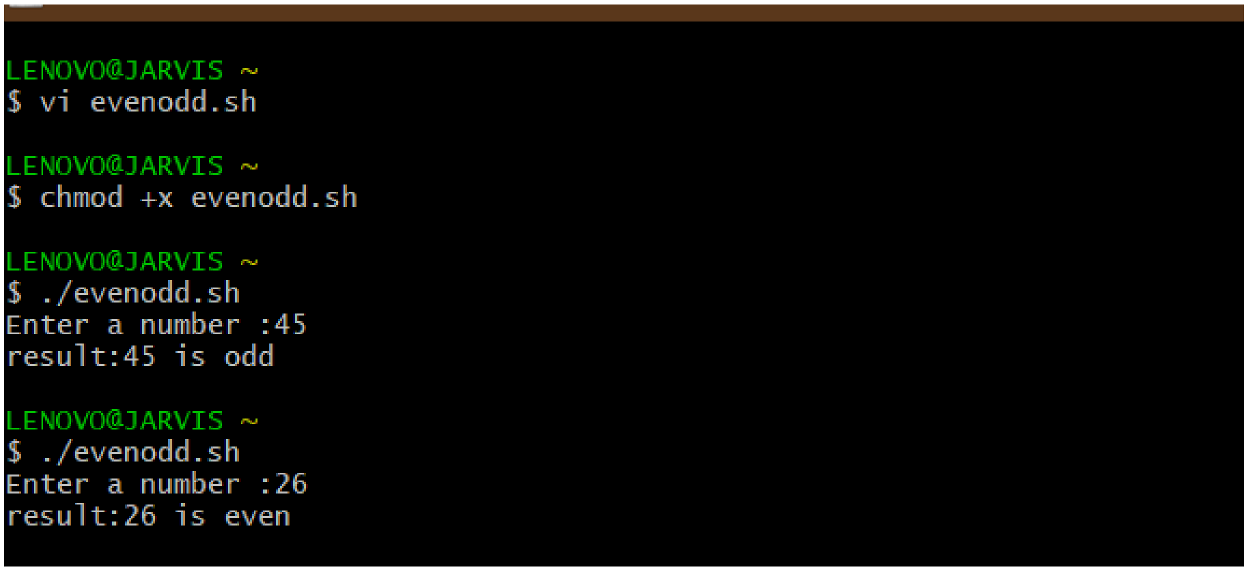
Write and execute permission.

Read Permission

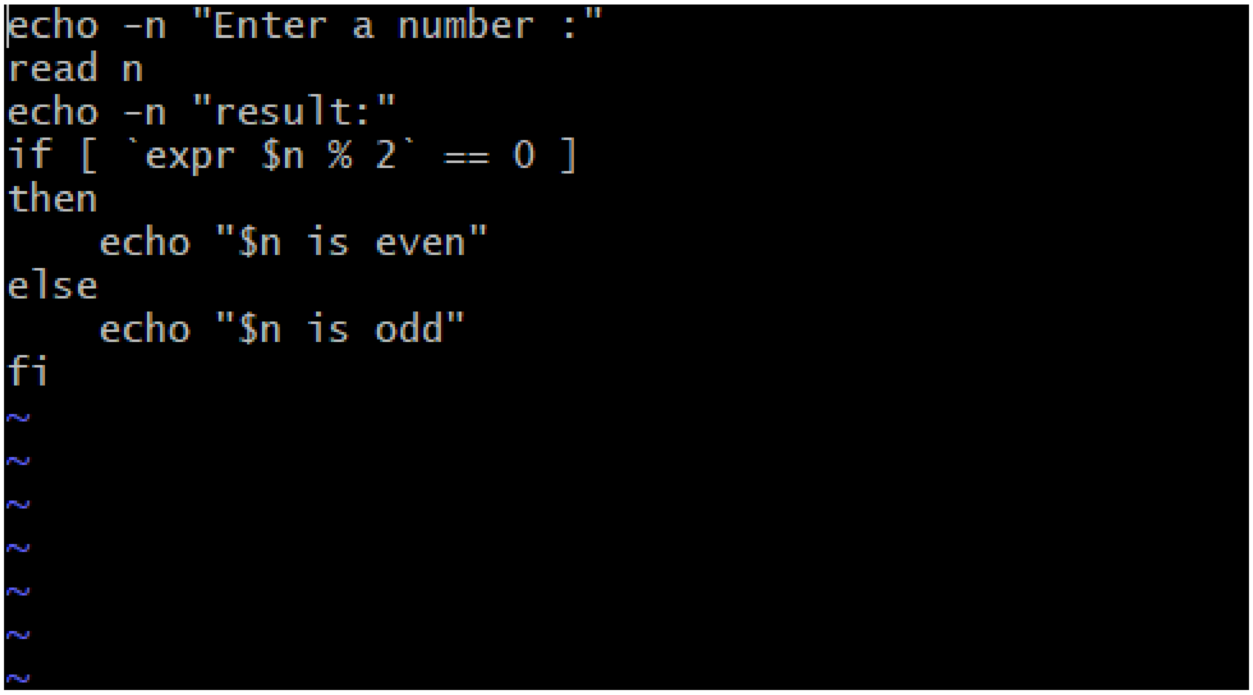
Q5 Ans - By Using #

Screenshots

a) Write a shell program to check whether a number is even or odd

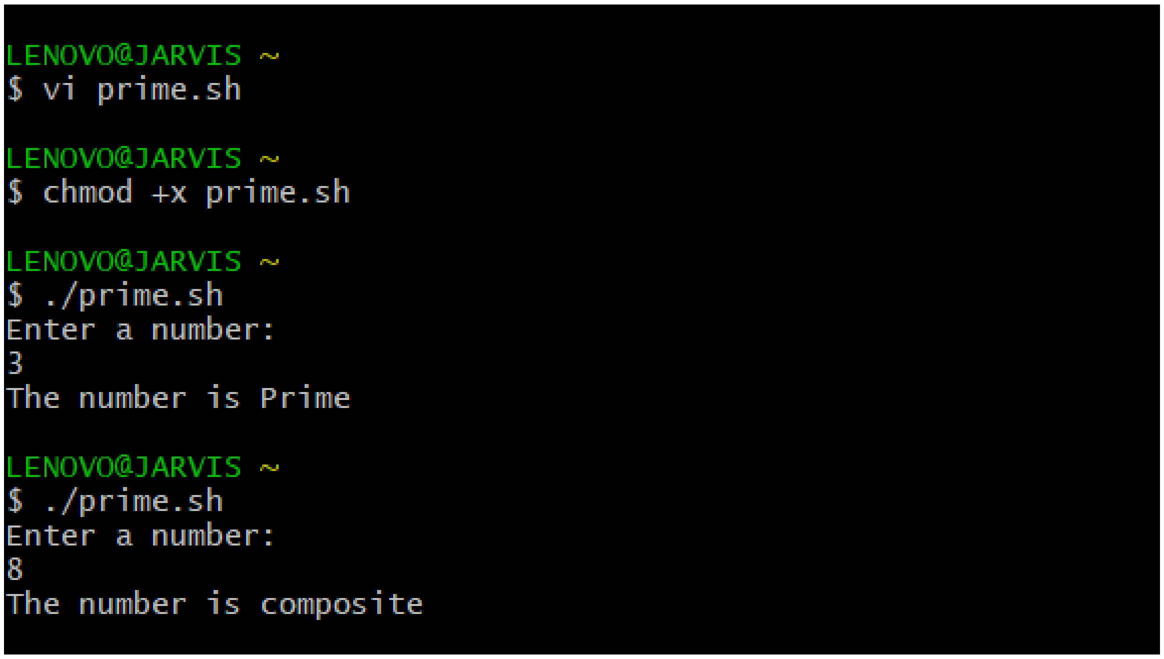


```
LENOVO@JARVIS ~
$ vi evenodd.sh

LENOVO@JARVIS ~
$ chmod +x evenodd.sh

LENOVO@JARVIS ~
$ ./evenodd.sh
Enter a number :45
result:45 is odd

LENOVO@JARVIS ~
$ ./evenodd.sh
Enter a number :26
result:26 is even
```

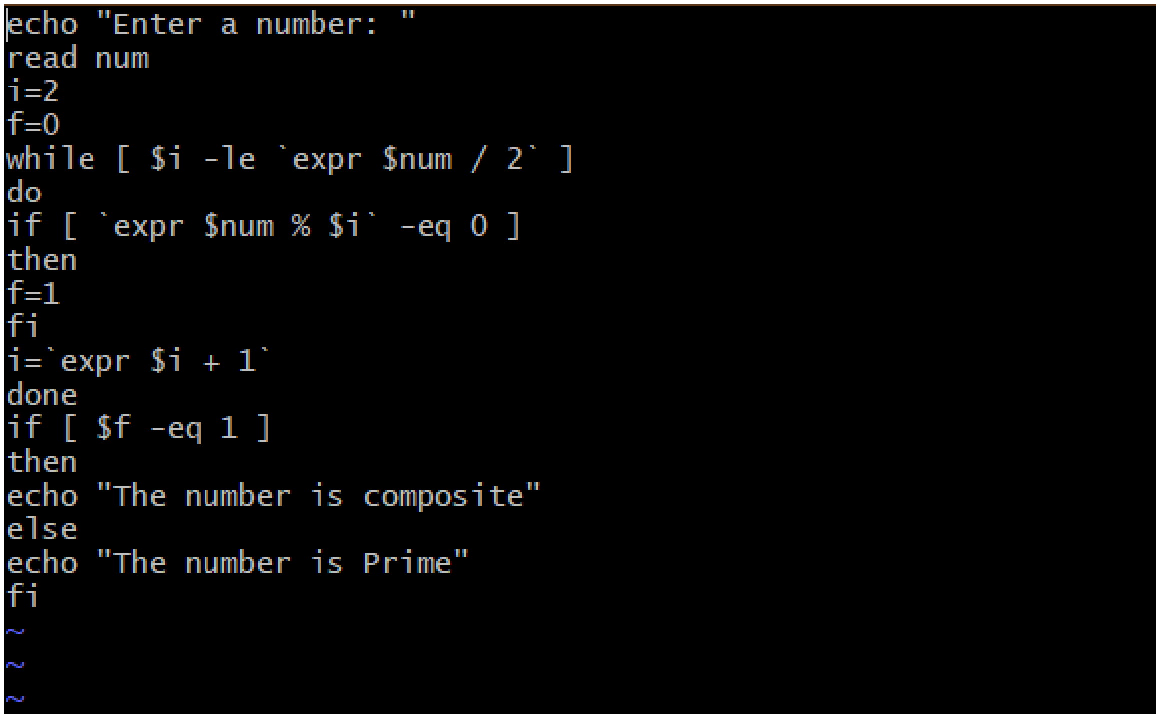


```
echo -n "Enter a number :"
read n
echo -n "result:"
if [ `expr $n % 2` == 0 ]
then
    echo "$n is even"
else
    echo "$n is odd"
fi
~
~
~
~
~
~
~
```

b) Write a shell program to find whether a number is prime or not.

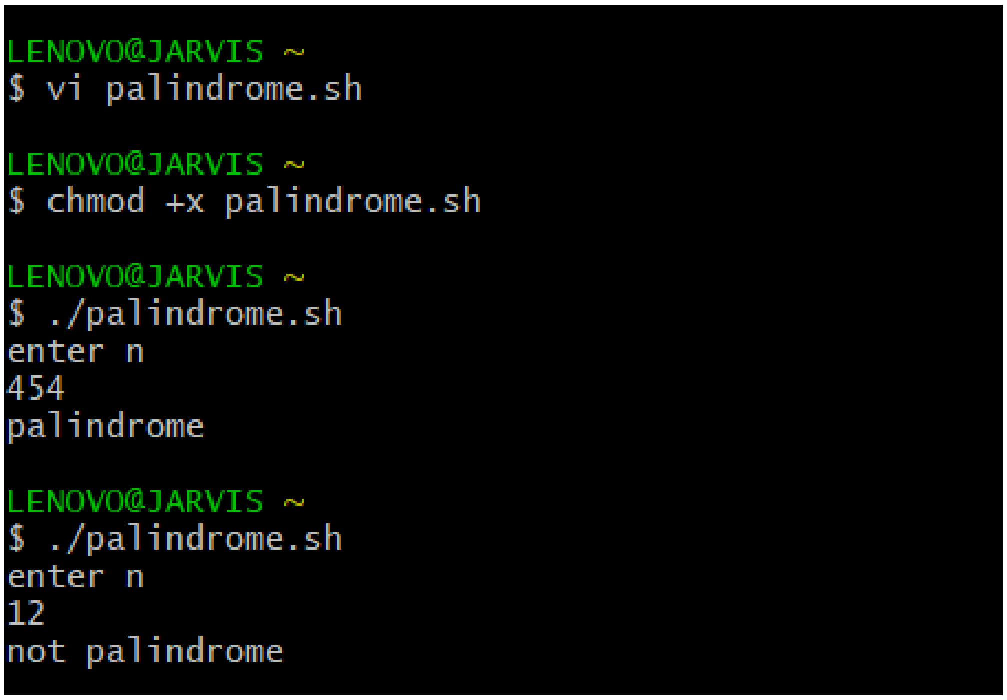


```
LENOVO@JARVIS ~
$ vi prime.sh

LENOVO@JARVIS ~
$ chmod +x prime.sh

LENOVO@JARVIS ~
$ ./prime.sh
Enter a number:
3
The number is Prime

LENOVO@JARVIS ~
$ ./prime.sh
Enter a number:
8
The number is composite
```

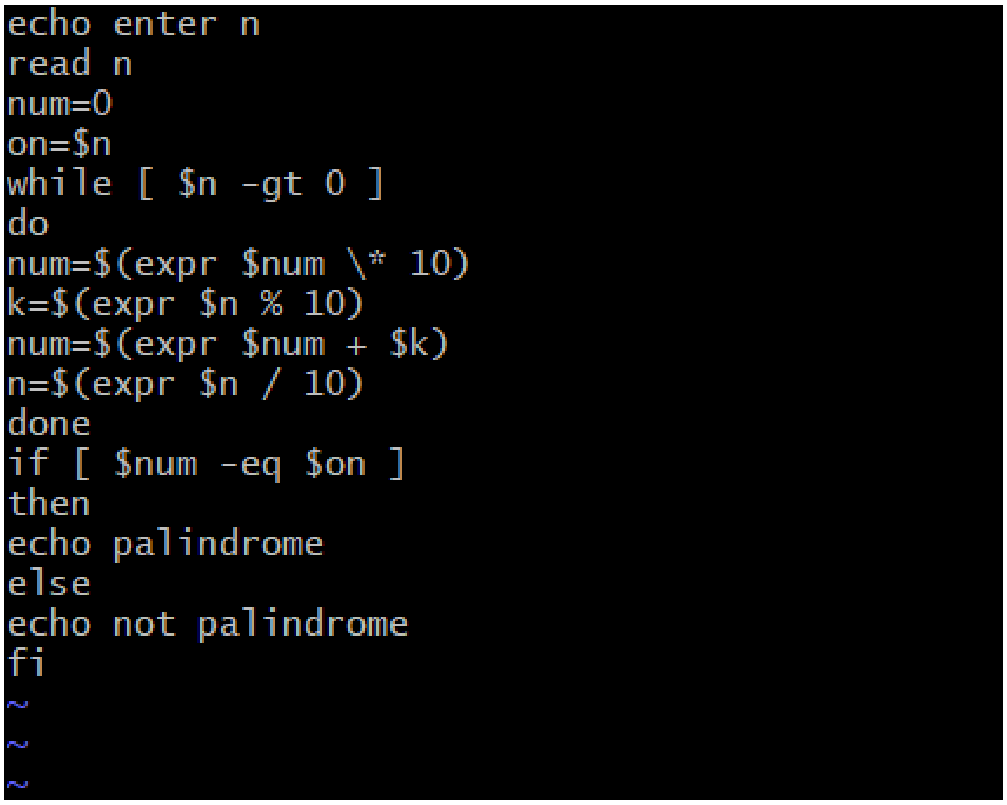


```
echo "Enter a number: "
read num
i=2
f=0
while [ $i -le `expr $num / 2` ]
do
if [ `expr $num % $i` -eq 0 ]
then
f=1
fi
i=`expr $i + 1`
done
if [ $f -eq 1 ]
then
echo "The number is composite"
else
echo "The number is Prime"
fi
~
~
~
```

c) Write a shell program to find whether a number is palindrome or not.

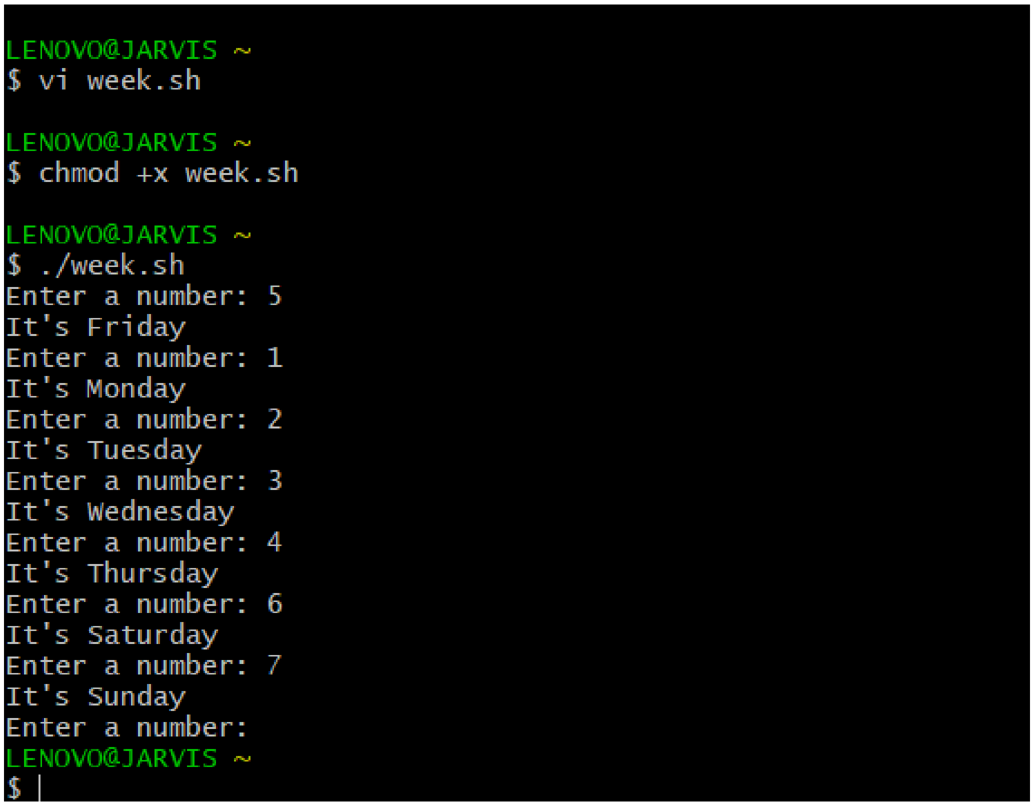


```
LENOVO@JARVIS ~
$ vi palindrome.sh

LENOVO@JARVIS ~
$ chmod +x palindrome.sh

LENOVO@JARVIS ~
$ ./palindrome.sh
enter n
454
palindrome

LENOVO@JARVIS ~
$ ./palindrome.sh
enter n
12
not palindrome
```

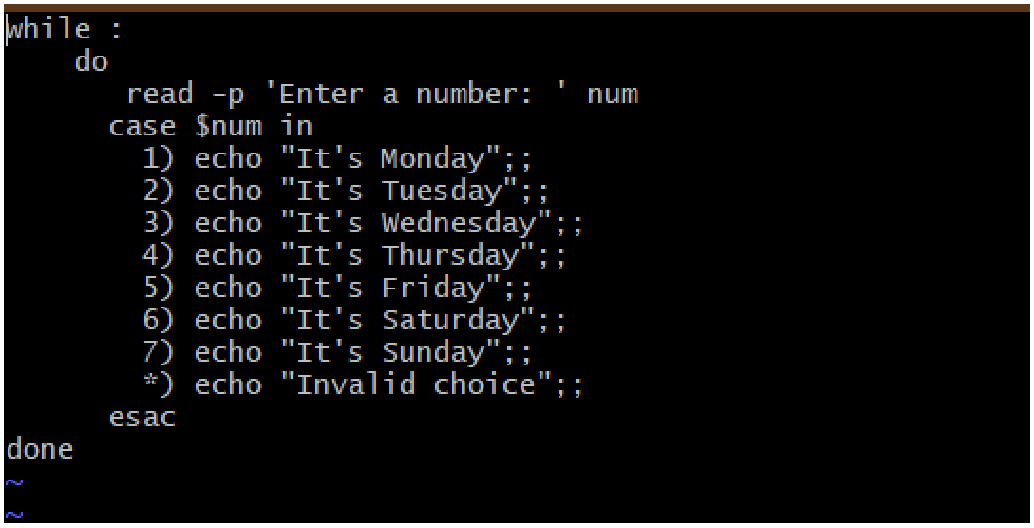


```
echo enter n
read n
num=0
on=$n
while [ $n -gt 0 ]
do
num=$(expr $num \* 10)
k=$(expr $n % 10)
num=$(expr $num + $k)
n=$(expr $n / 10)
done
if [ $num -eq $on ]
then
echo palindrome
else
echo not palindrome
fi
~
~
~
```

d) Write a shell program to type number 1 to 7 and then print its corresponding day of week



```
LENOVO@JARVIS ~
$ vi week.sh

LENOVO@JARVIS ~
$ chmod +x week.sh

LENOVO@JARVIS ~
$ ./week.sh
Enter a number: 5
It's Friday
Enter a number: 1
It's Monday
Enter a number: 2
It's Tuesday
Enter a number: 3
It's Wednesday
Enter a number: 4
It's Thursday
Enter a number: 6
It's Saturday
Enter a number: 7
It's Sunday
Enter a number:
LENOVO@JARVIS ~
$ 
```



```
while :
    do
       read -p 'Enter a number: ' num
      case $num in
        1) echo "It's Monday";;
        2) echo "It's Tuesday";;
        3) echo "It's Wednesday";;
        4) echo "It's Thursday";;
        5) echo "It's Friday";;
        6) echo "It's Saturday";;
        7) echo "It's Sunday";;
        *) echo "Invalid choice";;
      esac
done
~
~
```

Experiment No: 5

Student Name and Roll Number: Namit Kumar
Semester /Section: V/FS-A1
Link to Code: https://github.com/NamitKumar16/OS
Date: 8th September
Faculty Signature:
Marks:

Objective:
Write a program to implement CPU scheduling for first come first serve approach.

Outcome:
The students will understand the First-cum-first-serve algorithm

Problem Statement:
Implement the following CPU scheduling Algorithms.

i) FCFS with Arrival time
ii) FCFS without Arrival time

Background Study:
FCFS

- The simplest CPU-scheduling algorithm is the first-come, first-served (FCFS) scheduling algorithm. With this algorithm, processes are assigned the CPU in the order they request it.
- There is a single queue of ready processes.
- The implementation of the FCFS policy is easily managed with a FIFO queue. When a process enters the ready queue, its PCB is linked onto the tail of the queue.
- The average waiting time under the FCFS policy, however, is often quite long.

Question Bank:

1. Which module gives control of the CPU to the process selected by the short-term scheduler?
a) **dispatche**r
b) interrupt
c) scheduler
d) none of the mentioned

2. The processes that are residing in main memory and are ready and waiting to execute are kept on a list called

a) job queue
b) ready queue
c) execution queue
d) process queue

3. The interval from the time of submission of a process to the time of completion is termed as
a) waiting time
b) turnaround time
c) response time
d) throughput

4. Which scheduling algorithm allocates the CPU first to the process that requests the CPU first?
a) first-come, first-served scheduling
b) shortest job scheduling
c) priority scheduling
d) none of the mentioned

5. In priority scheduling algorithm

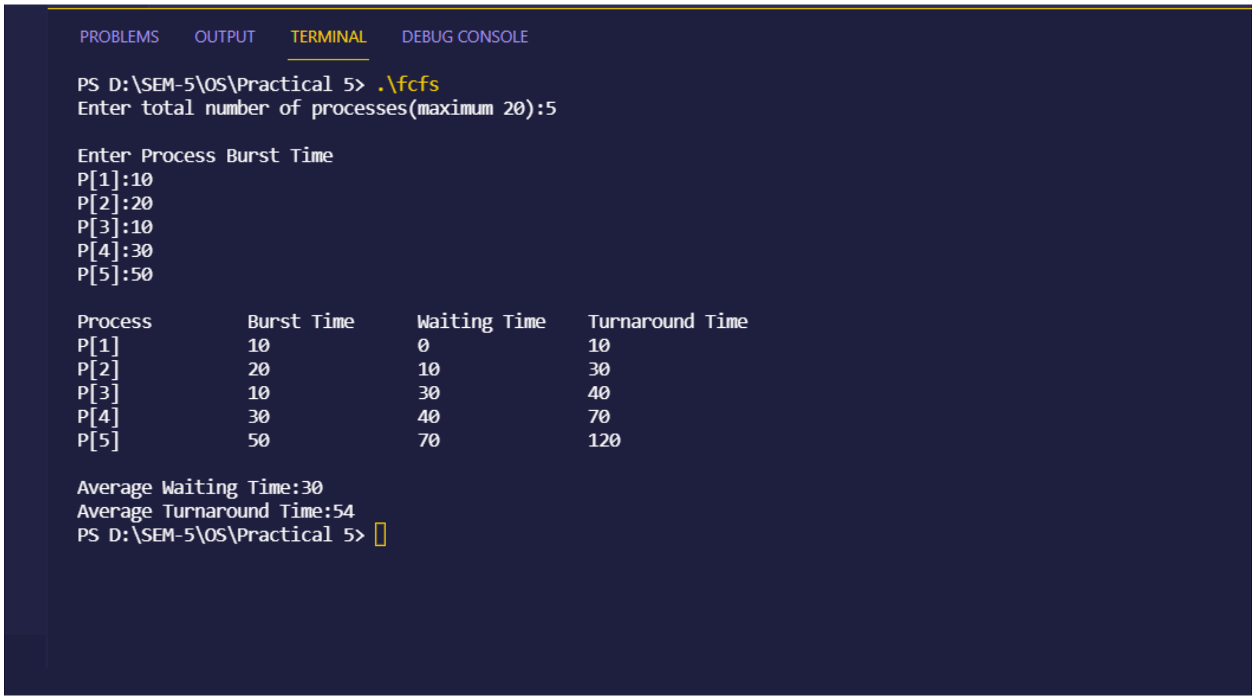
a) CPU is allocated to the process with highest priority
b) CPU is allocated to the process with lowest priority
c) equal priority processes cannot be scheduled
d) none of the mentioned

Student Work Area

Algorithm/Flowchart/Code/Sample Outputs

```
#include<stdio.h>


int main()
{
   int n,bt[20],wt[20],tat[20],avwt=0,avtat=0,i,j;
   printf("Enter total number of processes(maximum 20):");
   scanf("%d",&n);

   printf("\nEnter Process Burst Time\n");
   for(i=0;i<n;i++)
   {
      printf("P[%d]:",i+1);
      scanf("%d",&bt[i]);
   }

   wt[0]=0;

   for(i=1;i<n;i++)
   {
      wt[i]=0;
```

```
        for(j=0;j<i;j++)
            wt[i]+=bt[j];
    }

    printf("\nProcess\t\tBurst Time\tWaiting Time\tTurnaround Time");

    //calculating turnaround time
    for(i=0;i<n;i++)
    {
        tat[i]=bt[i]+wt[i];
        avwt+=wt[i];
        avtat+=tat[i];
        printf("\nP[%d]\t\t%d\t\t%d\t\t%d",i+1,bt[i],wt[i],tat[i]);
    }

    avwt/=i;
    avtat/=i;
    printf("\n\nAverage Waiting Time:%d",avwt);
    printf("\nAverage Turnaround Time:%d",avtat);

    return 0;
}
```



```
PS D:\SEM-5\OS\Practical 5> .\fcfs
Enter total number of processes(maximum 20):5

Enter Process Burst Time
P[1]:10
P[2]:20
P[3]:10
P[4]:30
P[5]:50

Process         Burst Time      Waiting Time    Turnaround Time
P[1]            10              0               10
P[2]            20              10              30
P[3]            10              30              40
P[4]            30              40              70
P[5]            50              70              120

Average Waiting Time:30
Average Turnaround Time:54
PS D:\SEM-5\OS\Practical 5>
```

Experiment No: 6

Student Name and Roll Number: Namit Kumar 19CSU185	
Semester /Section: V/FSA1	
Link to Code: https://github.com/NamitKumar16/OS	
Date: 8th September 2021	
Faculty Signature:	
Marks:	

Objective:
Write a program to implement CPU scheduling for shortest job first (Preemptive and Non-Preemptive)

Outcome:
The students will understand the Shortest Job First scheduling mechanism

Problem Statement:
Implement the following CPU scheduling Algorithms.

- SJF (Non-Preemptive)
- SJTF (shortest remaining time first -Preemptive SJF)

Background Study:

- Shortest Job first is having the advantage of a minimum average waiting time .
- This algorithm associates with each process the length of the process next burst time.When CPU is available it assigned to the process that has the smallest next CPU burst time.if CPU burst time of two process is same then it follows FCFS.
- It may cause starvation if shorter processes keep coming. This problem can be solved using the concept of ageing.
- It is practically infeasible as Operating System may not know burst time and therefore may not sort them.

Question Bank:

1. scheduling algorithm In multilevel feedback
   A. processes are not classified into groups
   B. a process can move to a different classified ready queue…
   C. classification of the ready queue is permanent
   D. none of the mentioned
2. Select one which algorithms tend to minimize the process flow time?
   A. First come First served
   B. Earliest Deadline First
   C. Shortest Job First
   D. Longest Job First

3. The process can be classified into many groups in
   A. shortest job scheduling algorithm
   B. multilevel queue scheduling algorithm
   C. round-robin scheduling algorithm
   D. priority scheduling algorithm
4. The turnaround time for short jobs during multiprogramming is usually Shortened and that for long jobs is slightly ____________
   A. Shortened
   B. Unchanged
   C. Lengthened
   D. Shortened
5. Time quantum can be said
   A. multilevel queue scheduling algorithm
   B. round-robin scheduling algorithm
   C. shortest job scheduling algorithm
   D. priority scheduling algorithm

Student Work Area

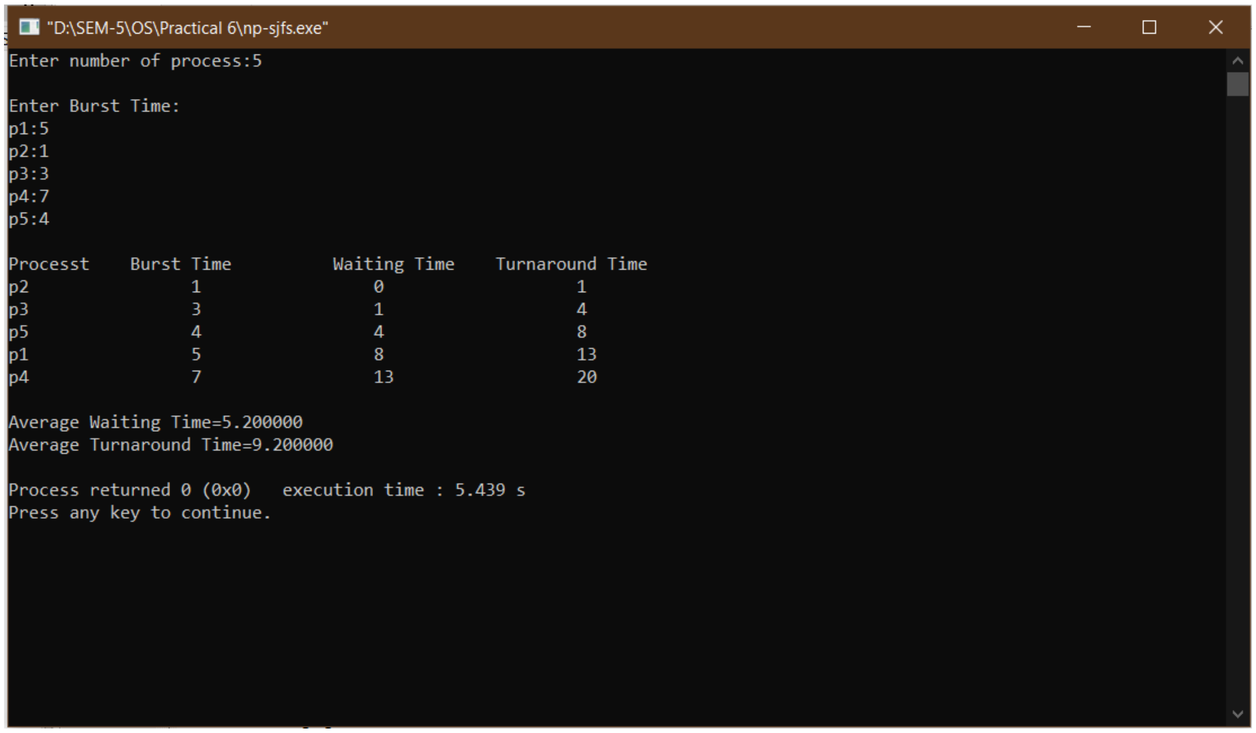
Algorithm/Flowchart/Code/Sample Outputs

SJF Non - Preemptive

```
#include<stdio.h>

 int main()

{

    int bt[20],p[20],wt[20],tat[20],i,j,n,total=0,pos,temp;

    float avg_wt,avg_tat;

    printf("Enter number of process:");

    scanf("%d",&n);


    printf("\nEnter Burst Time:\n");

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

    {

        printf("p%d:",i+1);

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

        p[i]=i+1;

    }


   //sorting of burst times

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

    {
```

```
    pos=i;

    for(j=i+1;j<n;j++)

    {

        if(bt[j]<bt[pos])

            pos=j;

    }


    temp=bt[i];

    bt[i]=bt[pos];

    bt[pos]=temp;


    temp=p[i];

    p[i]=p[pos];

    p[pos]=temp;

}


wt[0]=0;



for(i=1;i<n;i++)

{

    wt[i]=0;
```

```
        for(j=0;j<i;j++)
            wt[i]+=bt[j];

        total+=wt[i];
    }

    avg_wt=(float)total/n;
    total=0;

    printf("\nProcesst    Burst Time    \tWaiting Time\tTurnaround Time");
    for(i=0;i<n;i++)
    {
        tat[i]=bt[i]+wt[i];
        total+=tat[i];
        printf("\np%d\t\t  %d\t\t    %d\t\t\t%d",p[i],bt[i],wt[i],tat[i]);
    }

    avg_tat=(float)total/n;
    printf("\n\nAverage Waiting Time=%f",avg_wt);
    printf("\nAverage Turnaround Time=%f\n",avg_tat);
}
```



"D:\SEM-5\OS\Practical 6\np-sjfs.exe"

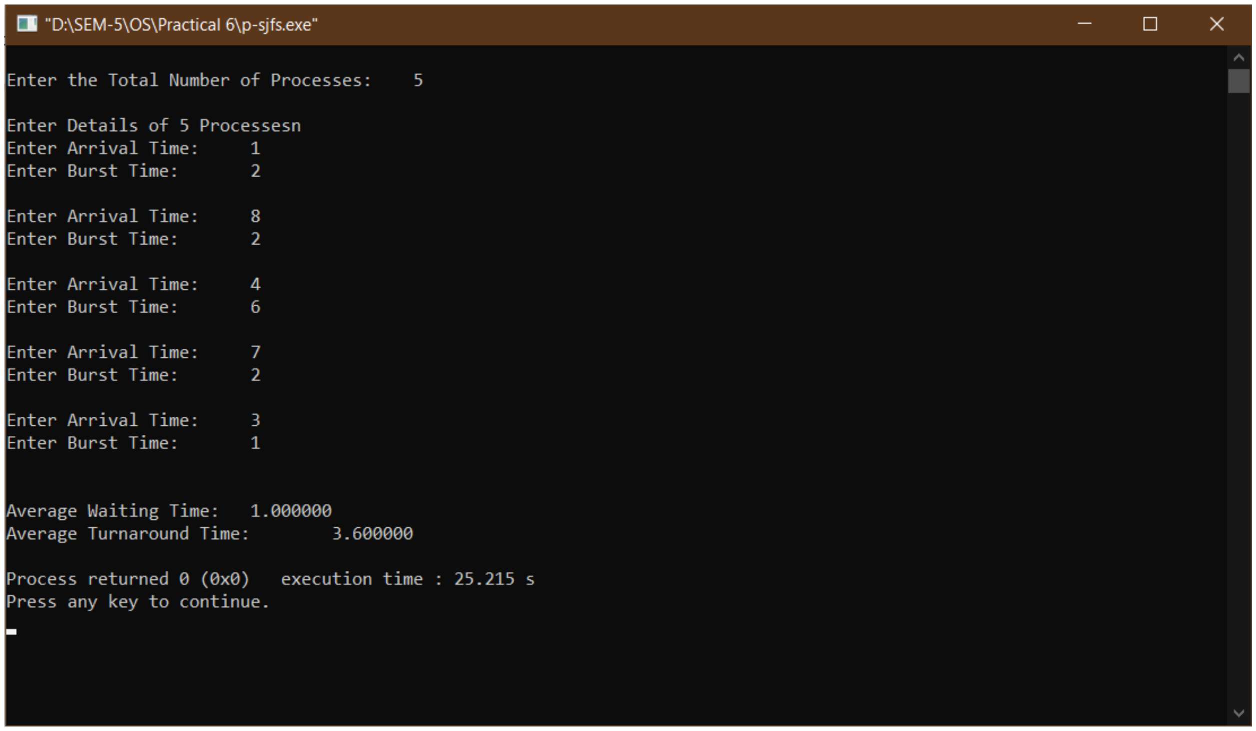
```
Enter number of process:5

Enter Burst Time:
p1:5
p2:1
p3:3
p4:7
p5:4

Processt    Burst Time          Waiting Time    Turnaround Time
p2              1                   0                   1
p3              3                   1                   4
p5              4                   4                   8
p1              5                   8                   13
p4              7                   13                  20

Average Waiting Time=5.200000
Average Turnaround Time=9.200000

Process returned 0 (0x0)   execution time : 5.439 s
Press any key to continue.
```

SJF Preemptive

```
#include <stdio.h>


int main()

{

    int arrival_time[10], burst_time[10], temp[10];

    int i, smallest, count = 0, time, limit;

    double wait_time = 0, turnaround_time = 0, end;

    float average_waiting_time, average_turnaround_time;

    printf("\nEnter the Total Number of Processes:\t");

    scanf("%d", &limit);
```

```
printf("\nEnter Details of %d Processesn", limit);

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

{

    printf("\nEnter Arrival Time:\t");

    scanf("%d", &arrival_time[i]);

    printf("Enter Burst Time:\t");

    scanf("%d", &burst_time[i]);

    temp[i] = burst_time[i];

}

burst_time[9] = 9999;

for(time = 0; count != limit; time++)

{

    smallest = 9;

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

    {

        if(arrival_time[i] <= time && burst_time[i] < burst_time[smallest] && burst_time[i] > 0)

        {

            smallest = i;

        }

    }

    burst_time[smallest]--;

    if(burst_time[smallest] == 0)
```

```
        {
            count++;
            end = time + 1;
            wait_time = wait_time + end - arrival_time[smallest] - temp[smallest];
            turnaround_time = turnaround_time + end - arrival_time[smallest];
        }
    }
    average_waiting_time = wait_time / limit;
    average_turnaround_time = turnaround_time / limit;
    printf("\n\nAverage Waiting Time:\t%lf\n", average_waiting_time);
    printf("Average Turnaround Time:\t%lf\n", average_turnaround_time);
    return 0;
}
```



"D:\SEM-5\OS\Practical 6\p-sjfs.exe"

```
Enter the Total Number of Processes:    5

Enter Details of 5 Processesn
Enter Arrival Time:     1
Enter Burst Time:       2

Enter Arrival Time:     8
Enter Burst Time:       2

Enter Arrival Time:     4
Enter Burst Time:       6

Enter Arrival Time:     7
Enter Burst Time:       2

Enter Arrival Time:     3
Enter Burst Time:       1


Average Waiting Time:   1.000000
Average Turnaround Time:        3.600000

Process returned 0 (0x0)   execution time : 25.215 s
Press any key to continue.
```

Experiment No: 7

Student Name and Roll Number: Namit Kumar
Semester /Section: V/FSA-1
Link to Code: https://github.com/NamitKumar16/OS
Date: 6th October 2021
Faculty Signature:
Marks:

Objective:
Write a program to perform priority scheduling among a set of processes.

Outcome:
Student will understand the working of priority scheduling among a set of processes.

Problem Statement:
Implement the priority scheduling.

Background Study:
Priority scheduling is a non-preemptive algorithm and used in batch systems. Each process is assigned a priority. Process with highest priority is to be executed first and so on.

Question Bank:

1. What are advantages of Priority scheduling?
2. What are disadvantages of priority scheduling?
3. At the ready queue when a process arrives In priority scheduling algorithm, the priority of this process is compared with the priority of?
   A. currently running process
   B. parent process
   C. all process
   D. init process
4. Differentiate between pre-emptive and non pre-emptive scheduling?
5. What is total no. of queue required to perform Priority Scheduling ?

Student Work Area

Algorithm/Flowchart/Code/Sample Outputs

Q1

- Easy to use scheduling method
- Processes are executed on the basis of priority so high priority does not need to wait for long which saves time
- This method provides a good mechanism where the relative important of each process may be precisely defined.
- Suitable for applications with fluctuating time and resource requirements.

Q2

- If the system eventually crashes, all low priority processes get lost.
- If high priority processes take lots of CPU time, then the lower priority processes may starve and will be postponed for an indefinite time.
- This scheduling algorithm may leave some low priority processes waiting indefinitely.
- A process will be blocked when it is ready to run but has to wait for the CPU because some other process is running currently.
- If a new higher priority process keeps on coming in the ready queue, then the process which is in the waiting state may need to wait for a long duration of time.

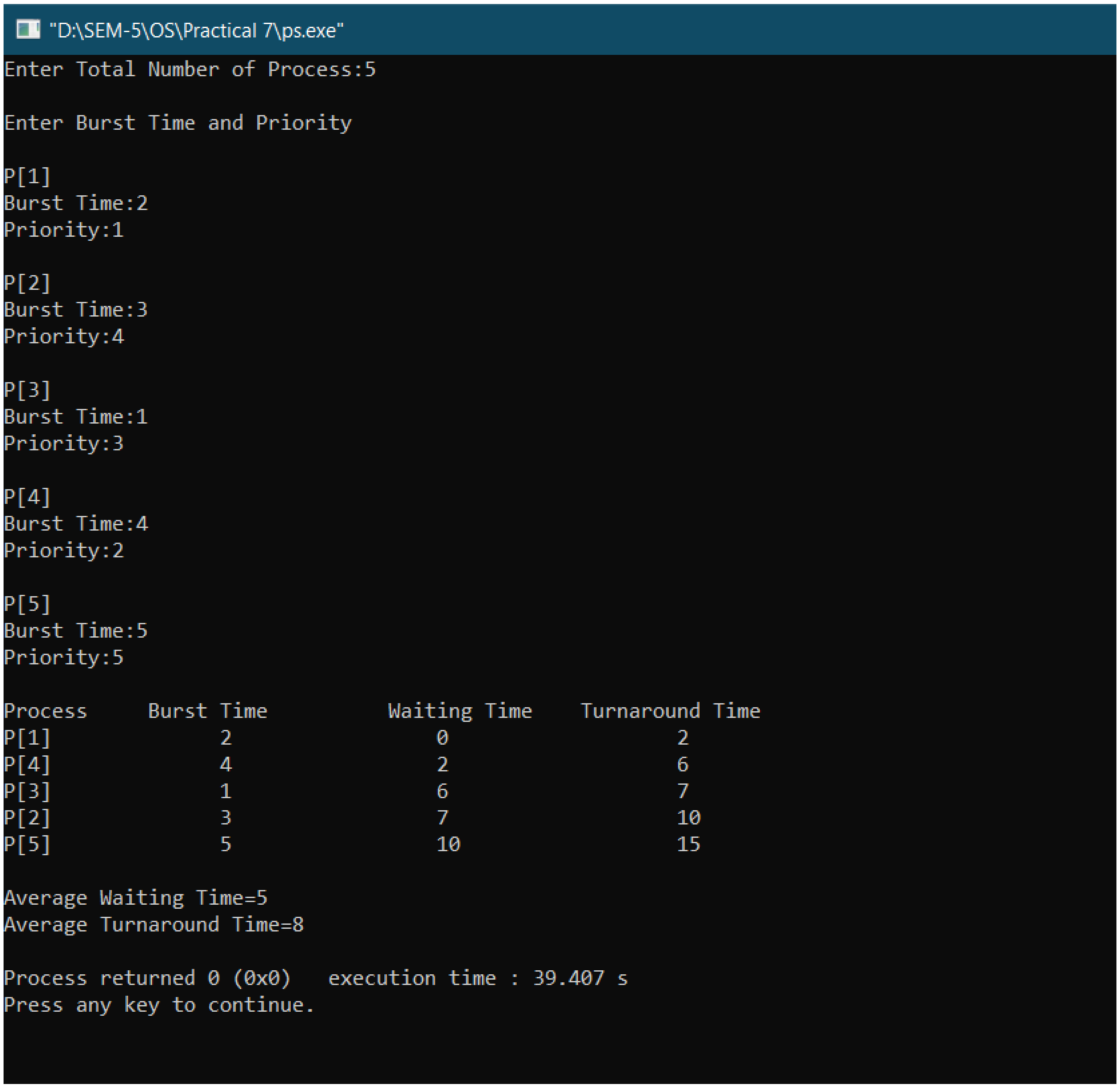
Q3 A – Currently Running Process

Q4 – The basic difference between preemptive and non-preemptive scheduling is that in preemptive scheduling the CPU is allocated to the processes for the limited time. While in Non-preemptive scheduling, the CPU is allocated to the process till it terminates or switches to waiting state.

Algorithm/Flowchart/Code/Sample Outputs

```
#include<stdio.h>


int main()
```

```
{
    int bt[20],p[20],wt[20],tat[20],pr[20],i,j,n,total=0,pos,temp,avg_wt,avg_tat;
    printf("Enter Total Number of Process:");
    scanf("%d",&n);

    printf("\nEnter Burst Time and Priority\n");
    for(i=0;i<n;i++)
    {
        printf("\nP[%d]\n",i+1);
        printf("Burst Time:");
        scanf("%d",&bt[i]);
        printf("Priority:");
        scanf("%d",&pr[i]);
        p[i]=i+1;
    }
    for(i=0;i<n;i++)
    {
        pos=i;
        for(j=i+1;j<n;j++)
        {
            if(pr[j]<pr[pos] || (pr[j]==pr[pos] && bt[j]>bt[pos]))
                pos=j;
        }

        temp=pr[i];
        pr[i]=pr[pos];
        pr[pos]=temp;

        temp=bt[i];
        bt[i]=bt[pos];
        bt[pos]=temp;

        temp=p[i];
        p[i]=p[pos];
        p[pos]=temp;
    }

    wt[0]=0;
    for(i=1;i<n;i++)
    {
        wt[i]=0;
        for(j=0;j<i;j++)
            wt[i]+=bt[j];
        total+=wt[i];
    }

    avg_wt=total/n;
    total=0;

    printf("\nProcess\t    Burst Time    \tWaiting Time\tTurnaround Time");
    for(i=0;i<n;i++)
    {
        tat[i]=bt[i]+wt[i];
        total+=tat[i];
        printf("\nP[%d]\t\t  %d\t\t    %d\t\t\t%d",p[i],bt[i],wt[i],tat[i]);
    }

    avg_tat=total/n;
    printf("\n\nAverage Waiting Time=%d",avg_wt);
    printf("\nAverage Turnaround Time=%d\n",avg_tat);

return 0;
}
```



```
Enter Total Number of Process:5

Enter Burst Time and Priority

P[1]
Burst Time:2
Priority:1

P[2]
Burst Time:3
Priority:4

P[3]
Burst Time:1
Priority:3

P[4]
Burst Time:4
Priority:2

P[5]
Burst Time:5
Priority:5

Process     Burst Time          Waiting Time    Turnaround Time
P[1]              2                 0                   2
P[4]              4                 2                   6
P[3]              1                 6                   7
P[2]              3                 7                   10
P[5]              5                 10                  15

Average Waiting Time=5
Average Turnaround Time=8

Process returned 0 (0x0)   execution time : 39.407 s
Press any key to continue.
```

Experiment No: 8

Student Name and Roll Number: Namit Kumar 19CSU185
Semester /Section: V/FS-A-1
Link to Code: https://github.com/NamitKumar16/OS
Date: 13th October 2021
Faculty Signature:
Marks:

Objective: Objective

To familiarize the students about CPU scheduling Algorithms

Program Outcome

The students will understand the Round Robin Algorithm.

Problem Statement:
Implement the Round Robin Algorithm.

Background Study:

- In Round Robin each process is assigned a fixed time slot in a cyclic way and this is preemptive. It has a disadvantage of context switch and have quantum time

Question Bank:

1. What is Preemptive and Non- Preemptive CPU scheduling? Explain with examples.
2. Explain the difference between short term, long term and medium term scheduling.
3. Explain the function of Dispatcher and Context Switch mechanism.
4. What are the advantages and disadvantages of Round robin?
5. Give the application are of Robin Robin.

Student Work Area

Algorithm/Flowchart/Code/Sample Outputs

Q1 – Preemptive scheduling is used when a process switches from running state to ready state or from the waiting state to ready state. The resources (mainly CPU cycles) are allocated to the process for a limited amount of time and then taken away, and the process is again placed back in the ready queue if that process still has CPU burst time remaining. That process stays in the ready queue till it gets its next chance to execute. Examples of preemptive scheduling are Round Robin and Shortest Remaining Time First.

Q2 –

Long Term	**Short Term**	**Medium Term**
It is a job scheduler.	It is a CPU scheduler.	It is swapping.
Speed is less than short term scheduler.	Speed is very fast.	Speed is in between both
It controls the degree of multiprogramming	Less control over the degree of multiprogramming.	Reduce the degree of multiprogramming.
It selects processes from the pool and load them into memory for execution.	It selects from among the processes that are ready to execute.	Process can be reintroduced into the meat and its execution can be continued.

Q3 – Schedulers are special system software which handle process scheduling in various ways. Their main task is to select the jobs to be submitted into the system and to decide which process to run.

A dispatcher is a special program which comes into play after the scheduler. When the scheduler completes its job of selecting a process, it is the dispatcher which takes that process to the desired state/queue.

Q4 – Advantages -

- Each process is served by the CPU for a fixed time quantum, so all processes are given the same priority.
- Starvation doesn't occur because, for each round robin cycle, every process is given a fixed time to execute. No process is left behind.

Disadvantages -

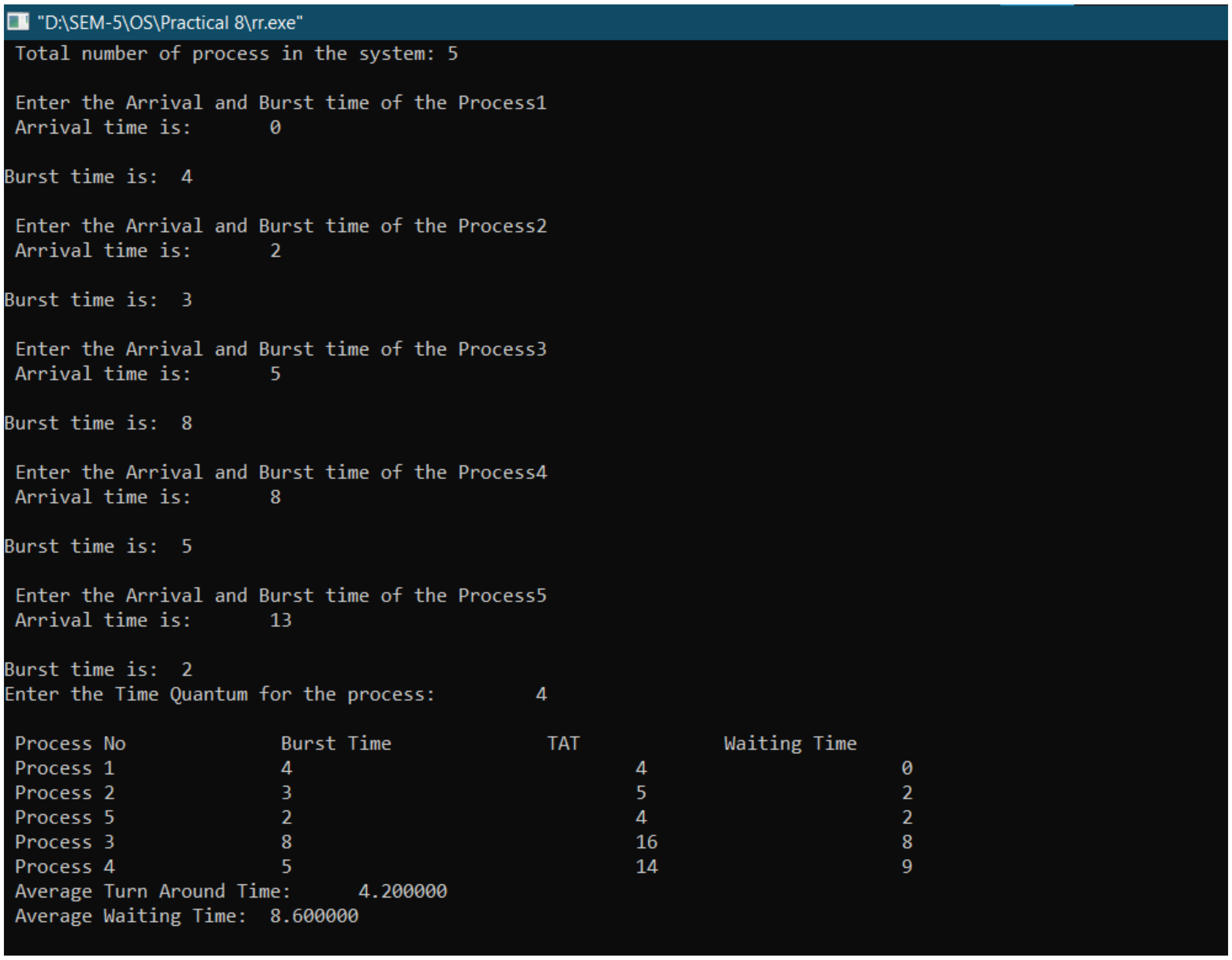
- The throughput in RR largely depends on the choice of the length of the time quantum. If time quantum is too large it behaves as FCFS. If time quantum is too short much of the time is spent in process switching and hence low throughput.
- Also here one cannot assign priority to any process which can be a drawback.

Q5 - Round-robin scheduling can be applied to other scheduling problems, such as data packet scheduling in computer networks.

```
#include<stdio.h>

#include<conio.h>


void main()

{

   int i, NOP, sum=0,count=0, y, quant, wt=0, tat=0, at[10], bt[10], temp[10];

   float avg_wt, avg_tat;

   printf(" Total number of process in the system: ");

   scanf("%d", &NOP);

   y = NOP;


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

{

printf("\n Enter the Arrival and Burst time of the Process%d\n", i+1);

printf(" Arrival time is: \t");

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

```
printf(" \nBurst time is: \t");

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

temp[i] = bt[i];

}

printf("Enter the Time Quantum for the process: \t");

scanf("%d", &quant);

printf("\n Process No \t\t Burst Time \t\t TAT \t\t Waiting Time ");

for(sum=0, i = 0; y!=0; )

{

if(temp[i] <= quant && temp[i] > 0)

{

   sum = sum + temp[i];

   temp[i] = 0;

   count=1;

   }

   else if(temp[i] > 0)

   {

      temp[i] = temp[i] - quant;

      sum = sum + quant;

   }

   if(temp[i]==0 && count==1)

   {
        y--;

        printf("\n Process %d \t\t %d\t\t\t\t %d\t\t\t %d", i+1, bt[i], sum-at[i], sum-at[i]-bt[i]);

        wt = wt+sum-at[i]-bt[i];

        tat = tat+sum-at[i];

        count =0;

    }

    if(i==NOP-1)

    {

        i=0;

    }

    else if(at[i+1]<=sum)

    {

        i++;

    }

    else

    {

        i=0;

    }

}


avg_wt = wt * 1.0/NOP;

avg_tat = tat * 1.0/NOP;
```

```
printf("\n Average Turn Around Time: \t%f", avg_wt);

printf("\n Average Waiting Time: \t%f", avg_tat);

getch();

}
```



"D:\SEM-5\OS\Practical 8\rr.exe"

```
 Total number of process in the system: 5

 Enter the Arrival and Burst time of the Process1
 Arrival time is:       0

Burst time is:  4

 Enter the Arrival and Burst time of the Process2
 Arrival time is:       2

Burst time is:  3

 Enter the Arrival and Burst time of the Process3
 Arrival time is:       5

Burst time is:  8

 Enter the Arrival and Burst time of the Process4
 Arrival time is:       8

Burst time is:  5

 Enter the Arrival and Burst time of the Process5
 Arrival time is:       13

Burst time is:  2
Enter the Time Quantum for the process:         4

 Process No              Burst Time              TAT             Waiting Time
 Process 1               4                               4                       0
 Process 2               3                               5                       2
 Process 5               2                               4                       2
 Process 3               8                               16                      8
 Process 4               5                               14                      9
 Average Turn Around Time:      4.200000
 Average Waiting Time:  8.600000
```

Experiment No: 9

Student Name and Roll Number: Namit Kumar
Semester /Section: V/FS-A-1
Link to Code: https://github.com/NamitKumar16/OS
Date: 20th October 2021
Faculty Signature:
Marks:

Objective: Write a program to implement reader/writer problem using semaphore

Program Outcome

The students will understand the reader/writer problem using semaphore

Problem Statement:
Write a program to implement reader/writer problem using semaphore

Background Study: There is a shared resource which should be accessed by multiple processes. There are two types of processes in this context. They are reader and writer. Any number of readers can read from the shared resource simultaneously, but only one writer can write to the shared resource. When a writer is writing data to the resource, no other process can access the resource. A writer cannot write to the resource if there are non-zero number of readers accessing the resource at that time.

Question Bank:

1. An un-interruptible unit is known as _____________
   a) single
   b) atomic
   c) static
   d) none of the mentioned
2. TestAndSet instruction is executed _____________
   a) after a particular process
   b) periodically
   c) atomically
   d) none of the mentioned

3. Semaphore is a/an ________ to solve the critical section problem.
a) hardware for a system
b) special program for a system
c) integer variable
d) none of the mentioned

4. What are the two atomic operations permissible on semaphores?
a) wait
b) stop
c) hold
d) none of the mentioned

5. When several processes access the same data concurrently and the outcome of the execution depends on the particular order in which the access takes place is called ________
a) dynamic condition
b) race condition
c) essential condition
d) critical condition

Student Work Area

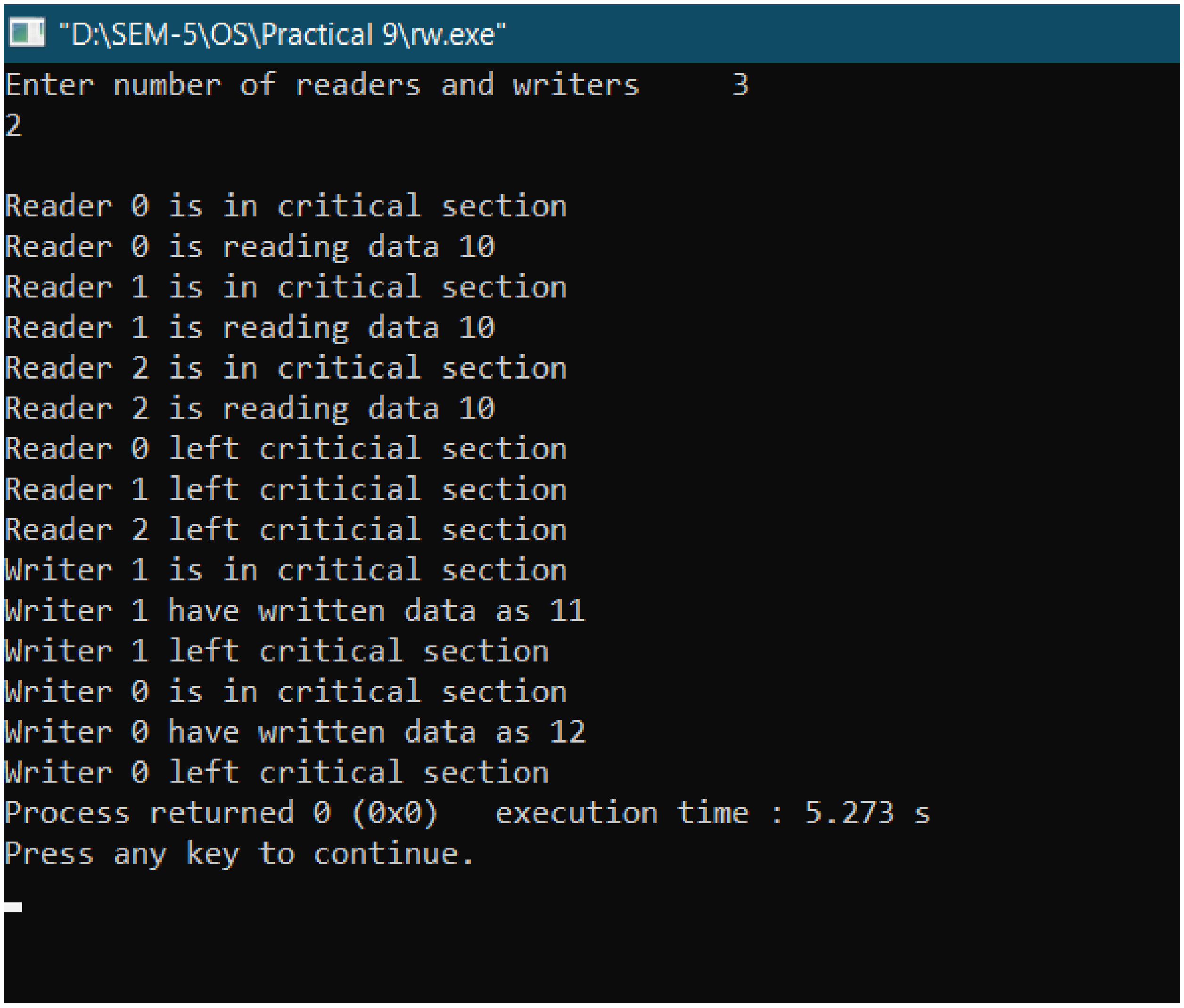
Algorithm/Flowchart/Code/Sample Outputs

```
#include<stdio.h>

#include<pthread.h>

#include<unistd.h>

#include<stdlib.h>


pthread_mutex_t wr,mutex;

int a = 10,readcount=0;


void * reader(void *arg)

{

   long int num;

   num=(long int) arg;

   pthread_mutex_lock(&mutex);

   readcount++;

   pthread_mutex_unlock(&mutex);

   if(readcount==1)

   {

      pthread_mutex_lock(&wr);

   }

   printf("\nReader %ld is in critical section",num);
```

```
    printf("\nReader %ld is reading data %d",num,a);
    sleep(1);

    pthread_mutex_lock(&mutex);
    readcount--;
    pthread_mutex_unlock(&mutex);
    if(readcount==0)
    {
        pthread_mutex_unlock(&wr);
    }
    printf("\nReader %ld left criticial section",num);
}
void * writer(void *arg)
{
    long int num;
    num=(long int) arg;
    pthread_mutex_lock(&wr);
    printf("\nWriter %ld is in critical section",num);
    printf("\nWriter %ld have written data as %d",num,++a);
    sleep(1);
    pthread_mutex_unlock(&wr);
    printf("\nWriter %ld left critical section",num);
```

```
}



int main()

{

   pthread_t r[10],w[10];

   long int i,j;

   int nor,now;

   pthread_mutex_init(&wr,NULL);

   pthread_mutex_init(&mutex,NULL);

   printf("Enter number of readers and writers\t");

   scanf("%d %d",&nor,&now);


   for(i=0;i<nor;i++)

   {

       pthread_create(&r[i],NULL,reader,(void *)i);

   }

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

   {

       pthread_create(&w[j],NULL,writer,(void *)j);

   }

    for(i=0;i<nor;i++)
```

```
  {
     pthread_join(r[i],NULL);
  }
   for(j=0;j<now;j++)
  {
     pthread_join(w[j],NULL);
  }
return 0;
}
```



"D:\SEM-5\OS\Practical 9\rw.exe"

```
Enter number of readers and writers     3
2

Reader 0 is in critical section
Reader 0 is reading data 10
Reader 1 is in critical section
Reader 1 is reading data 10
Reader 2 is in critical section
Reader 2 is reading data 10
Reader 0 left criticial section
Reader 1 left criticial section
Reader 2 left criticial section
Writer 1 is in critical section
Writer 1 have written data as 11
Writer 1 left critical section
Writer 0 is in critical section
Writer 0 have written data as 12
Writer 0 left critical section
Process returned 0 (0x0)   execution time : 5.273 s
Press any key to continue.
```

Experiment No: 10

Student Name and Roll Number: Namit Kumar
Semester /Section: V/FSA1
Link to Code: https://github.com/NamitKumar16/OS
Date: 10th November 2021
Faculty Signature:
Marks:

Objective:

Write a program to implement Dining Philosopher problem using semaphore

Outcome:
The students will understand the problem of synchronization among processes and its solution through Dining Philosopher problem using semaphore

Problem Statement:

Write a program to implement Dining Philosopher problem using semaphore

Background Study:
Five philosophers, spend their time thinking and eating spaghetti. They eat at a round table with five individual seats. For eating each philosopher needs two forks (the resources). There are five forks on the table, one left and one right of each seat. When a philosopher cannot grab both forks it sits and waits. Eating takes random time, then the philosopher puts the forks down and leaves the dining room. After spending some random time thinking he again becomes hungry, and the circle repeats itself.

Question Bank:

1. Which one of the following is a synchronization tool?
   a) thread
   b) pipe
   c) semaphore
   d) socket
2. A semaphore is a shared integer variable ___________
   a) that can not drop below zero
   b) that can not be more than zero

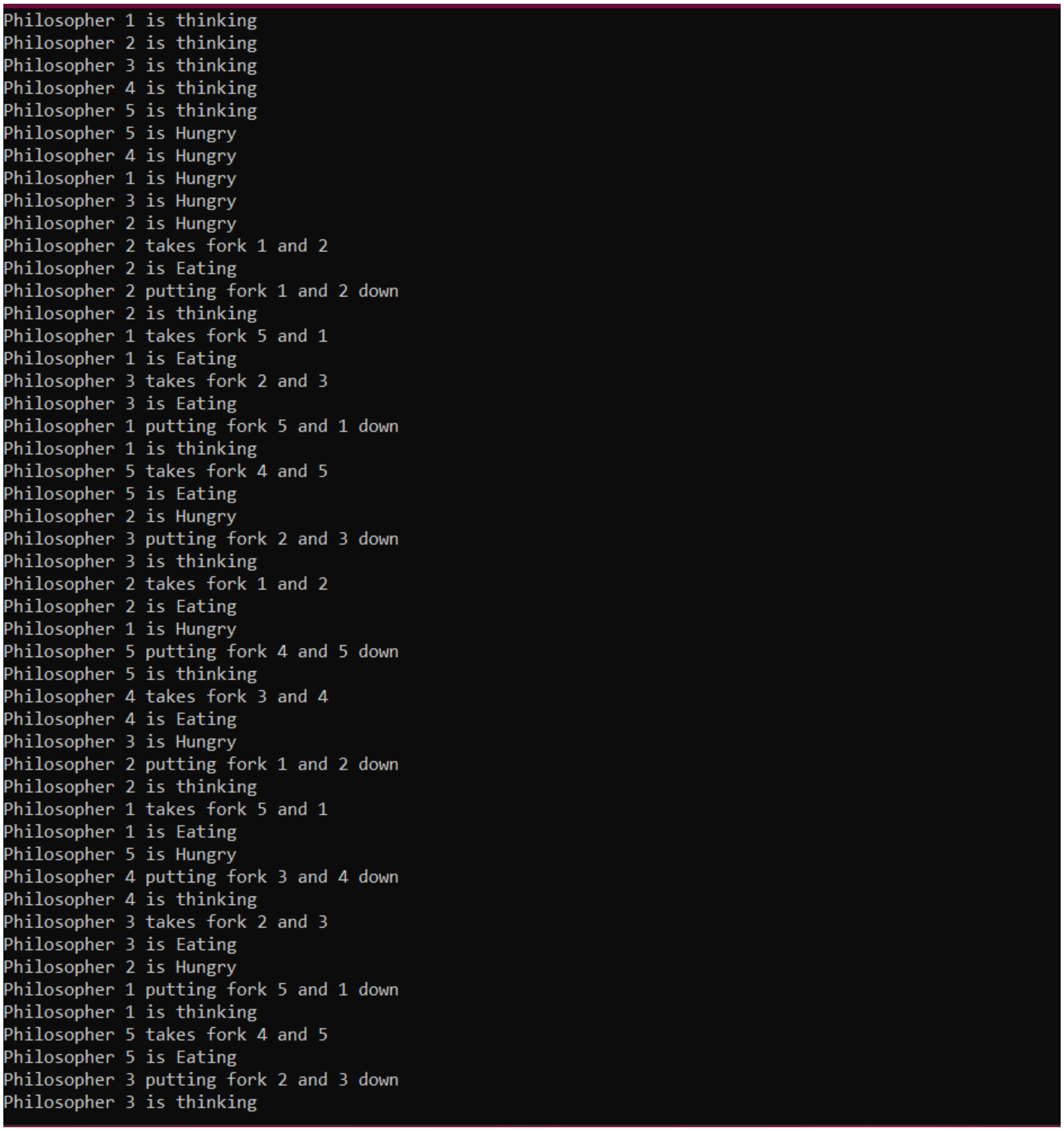
c) that can not drop below one
d) that can not be more than one

3. The bounded buffer problem is also known as ____________
   a) Readers – Writers problem
   b) Dining – Philosophers problem
   c) Producer – Consumer problem
   d) None of the mentioned
4. In the bounded buffer problem ____________
   a) there is only one buffer
   b) there are n buffers (n being greater than one but finite)
   c) there are infinite buffers
   d) the buffer size is bounded
5. To ensure difficulties do not arise in the readers – writers problem _______ are given exclusive access to the shared object.
   a) readers
   b) writers
   c) readers and writers
   d) none of the mentioned

Student Work Area

Algorithm/Flowchart/Code/Sample Outputs

```
#include <pthread.h>

#include <semaphore.h>

#include <stdio.h>

#define N 5

#define THINKING 2

#define HUNGRY 1

#define EATING 0

#define LEFT (phnum + 4) % N

#define RIGHT (phnum + 1) % N


int state[N];

int phil[N] = { 0, 1, 2, 3, 4 };


sem_t mutex;

sem_t S[N];


void test(int phnum)

{

        if (state[phnum] == HUNGRY

                && state[LEFT] != EATING
                && state[RIGHT] != EATING) {
                // state that eating
                state[phnum] = EATING;

                sleep(2);

                printf("Philosopher %d takes fork %d and %d\n",
                              phnum + 1, LEFT + 1, phnum + 1);

                printf("Philosopher %d is Eating\n", phnum + 1);

                // sem_post(&S[phnum]) has no effect
                // during takefork
                // used to wake up hungry philosophers
                // during putfork
                sem_post(&S[phnum]);
        }
}

// take up chopsticks
void take_fork(int phnum)
{
        sem_wait(&mutex);

        // state that hungry
        state[phnum] = HUNGRY;

        printf("Philosopher %d is Hungry\n", phnum + 1);

        // eat if neighbours are not eating
        test(phnum);

        sem_post(&mutex);

        // if unable to eat wait to be signalled
        sem_wait(&S[phnum]);

        sleep(1);
}

// put down chopsticks
void put_fork(int phnum)
{
        sem_wait(&mutex);

        // state that thinking
        state[phnum] = THINKING;

        printf("Philosopher %d putting fork %d and %d down\n",
                phnum + 1, LEFT + 1, phnum + 1);
        printf("Philosopher %d is thinking\n", phnum + 1);

        test(LEFT);
        test(RIGHT);

        sem_post(&mutex);
}

void* philospher(void* num)
{

        while (1) {

                int* i = num;
                sleep(1);

                take_fork(*i);

                sleep(0);

                put_fork(*i);
        }
}

int main()
{

        int i;
        pthread_t thread_id[N];

        // initialize the semaphores
        sem_init(&mutex, 0, 1);

        for (i = 0; i < N; i++)
        sem_init(&S[i], 0, 0);

    for (i = 0; i < N; i++) {

        // create philosopher processes
        pthread_create(&thread_id[i], NULL,
                       philospher, &phil[i]);

        printf("Philosopher %d is thinking\n", i + 1);
    }

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

        pthread_join(thread_id[i], NULL);
}
```



```
Philosopher 1 is thinking
Philosopher 2 is thinking
Philosopher 3 is thinking
Philosopher 4 is thinking
Philosopher 5 is thinking
Philosopher 5 is Hungry
Philosopher 4 is Hungry
Philosopher 1 is Hungry
Philosopher 3 is Hungry
Philosopher 2 is Hungry
Philosopher 2 takes fork 1 and 2
Philosopher 2 is Eating
Philosopher 2 putting fork 1 and 2 down
Philosopher 2 is thinking
Philosopher 1 takes fork 5 and 1
Philosopher 1 is Eating
Philosopher 3 takes fork 2 and 3
Philosopher 3 is Eating
Philosopher 1 putting fork 5 and 1 down
Philosopher 1 is thinking
Philosopher 5 takes fork 4 and 5
Philosopher 5 is Eating
Philosopher 2 is Hungry
Philosopher 3 putting fork 2 and 3 down
Philosopher 3 is thinking
Philosopher 2 takes fork 1 and 2
Philosopher 2 is Eating
Philosopher 1 is Hungry
Philosopher 5 putting fork 4 and 5 down
Philosopher 5 is thinking
Philosopher 4 takes fork 3 and 4
Philosopher 4 is Eating
Philosopher 3 is Hungry
Philosopher 2 putting fork 1 and 2 down
Philosopher 2 is thinking
Philosopher 1 takes fork 5 and 1
Philosopher 1 is Eating
Philosopher 5 is Hungry
Philosopher 4 putting fork 3 and 4 down
Philosopher 4 is thinking
Philosopher 3 takes fork 2 and 3
Philosopher 3 is Eating
Philosopher 2 is Hungry
Philosopher 1 putting fork 5 and 1 down
Philosopher 1 is thinking
Philosopher 5 takes fork 4 and 5
Philosopher 5 is Eating
Philosopher 3 putting fork 2 and 3 down
Philosopher 3 is thinking
```

Experiment No:11

Student Name and Roll Number: Namit Kumar 19CSU185
Semester /Section: V/FSA1
Link to Code: https://github.com/NamitKumar16/OS
Date: 17th November, 2021
Faculty Signature:
Marks:

Objective:
Write a program to implement Banker's algorithm for deadlock avoidance.

Outcome:
The students will understand how system handles deadlock using Banker's algorithm for deadlock avoidance

Problem Statement:
Write a program to implement Banker's algorithm for deadlock avoidance.

Background Study:
The banker's algorithm is a resource allocation and deadlock avoidance algorithm that tests for safety by simulating the allocation for predetermined maximum possible amounts of all resources, then makes an "s-state" check to test for possible activities, before deciding whether allocation should be allowed to continue. Banker's algorithm is named so because it is used in banking system to check whether loan can be sanctioned to a person or not. Suppose there are n number of account holders in a bank and the total sum of their money is S. If a person applies for a loan then the bank first subtracts the loan amount from the total money that bank has and if the remaining amount is greater than S then only the loan is sanctioned. It is done because if all the account holders comes to withdraw their money then the bank can easily do it.

Question Bank:

1. Each request requires that the system consider the _____________ to decide whether the current request can be satisfied or must wait to avoid a future possible deadlock.
   a) resources currently available
   b) processes that have previously been in the system
   c) resources currently allocated to each process
   d) future requests and releases of each process
2. Given a priori information about the ________ number of resources of each type that maybe requested for each process, it is possible to construct an algorithm that ensures that the system will never enter a deadlock state.
   a) minimum
   b) average

c) maximum
d) approximate

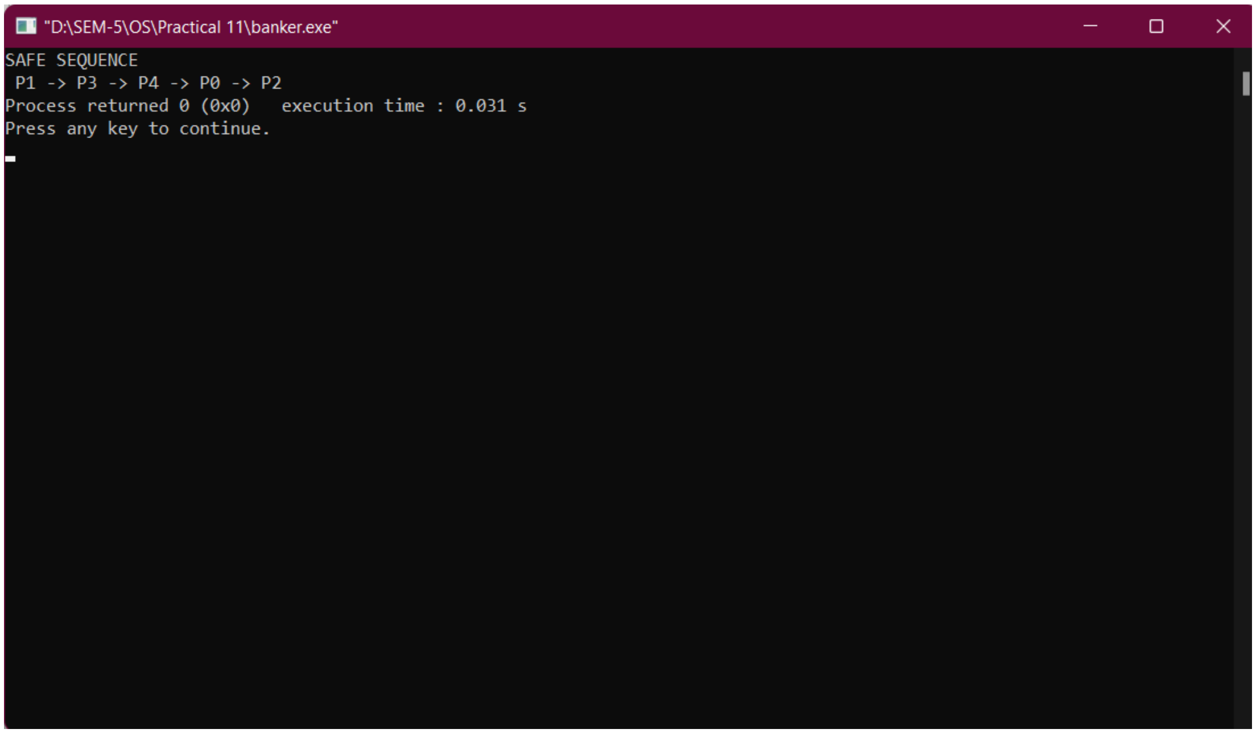
3. A deadlock avoidance algorithm dynamically examines the ___________ to ensure that a circular wait condition can never exist.
   a) resource allocation state
   b) system storage state
   c) operating system
   d) resources
4. A state is safe, if _____________
   a) the system does not crash due to deadlock occurrence
   b) the system can allocate resources to each process in some order and still avoid a deadlock
   c) the state keeps the system protected and safe
   d) all of the mentioned
5. The two ways of aborting processes and eliminating deadlocks are _____________
   a) Abort all deadlocked processes
   b) Abort all processes
   c) Abort one process at a time until the deadlock cycle is eliminated
   d) All of the mentioned

Student Work Area

Algorithm/Flowchart/Code/Sample Outputs

```
// Banker's Algorithm

#include <stdio.h>

int main()

{


    int n, m, i, j, k;

    n = 5;

    m = 3;

    int alloc[5][3] = { { 0, 1, 0 },

                        { 2, 0, 0 },

                        { 3, 0, 2 },

                        { 2, 1, 1 },

                        { 0, 0, 2 } };


    int max[5][3] = { { 7, 5, 3 },

                      { 3, 2, 2 },

                      { 9, 0, 2 },

                      { 2, 2, 2 },

                      { 4, 3, 3 } };
```

```
int avail[3] = { 3, 3, 2 };


int f[n], ans[n], ind = 0;

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

    f[k] = 0;

}

int need[n][m];

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

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

        need[i][j] = max[i][j] - alloc[i][j];

}

int y = 0;

for (k = 0; k < 5; k++) {

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

        if (f[i] == 0) {


            int flag = 0;

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

                if (need[i][j] > avail[j]){

                    flag = 1;

                     break;

                }
                }

                if (flag == 0) {
                    ans[ind++] = i;
                    for (y = 0; y < m; y++)
                        avail[y] += alloc[i][y];
                    f[i] = 1;
                }
            }
        }
    }

    printf("SAFE SEQUENCE\n");
    for (i = 0; i < n - 1; i++)
        printf(" P%d ->", ans[i]);
    printf(" P%d", ans[n - 1]);

    return (0);

}
```



"D:\SEM-5\OS\Practical 11\banker.exe"

```
SAFE SEQUENCE
 P1 -> P3 -> P4 -> P0 -> P2
Process returned 0 (0x0)   execution time : 0.031 s
Press any key to continue.
```

CASE STUDY

WINDOWS -

Windows Operating System was introduced into the market in the year 1985, and as a robust and comprehensive kind of software, has almost 90% market share over and above other operating systems. With its great and dominant presence in commercial buildings, industrial facilities, as well as its obvious presence as home computers. Although this assertion is believed not to be so again as a result of overwhelming people's interest in open-source operating systems. The Microsoft Operating System as a family of Microsoft Windows was created as a graphical layer over that of old MS dos with its root from MS-DOS Command line and this it retains till date with DOS Box command prompt that is cmd.exe. Original Windows NT core happens to be the first to take shape in an operating system upon which modern versions are dependent. 32 and 64-bit AMD and Intel systems accommodate Windows OS, DEC Alpha, PowerPC architectures Windows OS, and MIPS is also comfortable with recent versions, likewise low and mid-range servers. Database and web servers also allow Windows operating system to use them. In recent years, Microsoft has proofed significant with marketing and with its finance to establish that Windows interoperability is not in doubt and that it has all that it takes as a platform to run any enterprise application.

Merits of Windows OS

- Technical/Maintenance support: Support is made available either online or offline because of its general acceptability by so many users.
- Compatibility: Windows accommodates almost every application, game works, and different types of drivers.
- Enormous quantity of functions: Getting used to Windows, one would realize that there are many functions one can do almost anything quite easily with when call up.

Demerits of Windows OS

- Viruses: Need to purchase antivirus programs that need to be activated frequently, and this can be done on Auto or Manual mode, although free antivirus exists but with limitations.
- Slow: Windows operating system, particularly Vista and Windows 7 needs a lot of system resources like registers, cache, main memory, processor, disk space, and this makes the system runs slower.

- Price: The cost of purchasing Windows operating system is high and very few users can afford it and this necessitates cracking and makes pirated software versions available.

LINUX -

The GNU’s answer to MAC and also Windows is Linux. Yes, what this means is that Linux is a FREE OS, one can perform downloads, modification, and also redistribution without any cost. Linux is relatively new in the operating system realm. It was written in the year 1991, and also enhanced for current usage. Linux and Windows can be compared to an entity whose floor and roof are either replaceable or not. However, with Linux, as an entity, both floor and roof can be moved in any manner as one want, but Windows floor and roof are very rigid that it remains immovable. One cannot go beyond what Microsoft has designed Linux, designed by Linus Torvalds in the year 1991, heads a group of fresh school open-source Unix’s that came to be in the year 1990, it also includes FreeBSD, NetBSD, OpenBSD, and Darvin. All these are a representation of a design direction that the whole group agreed upon. Linux code is different compared to the original UNIX source tree code, however, it uses UNIX standards to behave like a UNIX. Developers in Linux open-source community have a desire to acquire a substantial share of end-user desktops making Linux's intended users increase in number than the users of the old-school Unix’s, who have fear share desire in the server and workstation market. The aspiration to reach end-users made Linux developers much more concerned with ease of installation and in resolving software distribution issues as it was more difficult with UNIX as proprietary systems, applications in Linux are forced to display a high degree of ruggedness than their colleagues with proprietary UNIX status.

Merits of Linux OS

- Price: Linux is FREE. It can be downloaded, installed, used, modified without incurring any cost.
- Variety: Linux is nowhere a complete OS but a kernel. The fact that it is a kernel, requires additional ad-ins in form of software. Many of these kinds of distributions or distros exist.
- Virus: The fact that it is open-sourced, it is less vulnerable compared with Mac, does not mean that it’s free from virus attacks.

Demerits of Linux OS

- Complicated: A good deal of Computer skills are required to use Linux distros even when some of them are quite easy to use.
- Compatibility: Although Linux has a few percent of the market share like Mac, however, it does not have many programs and games like that of Windows.
- Vendors: Linux has very few vendors selling Linux computers, if one needs a Linux computer, then it might be that one will need to purchase a Windows computer, reformat the hard drive, and then install Linux on it.

MAC OS

Mac OS is much older than Windows OS. It was released one year earlier than its Microsoft counterpart, and it happens to be the first among other OS, ever successful graphical-inclined OS. Mac OS has undergone, two important design transitions, and is on its third stage. The first transition was from supporting only a single application at one time to the ability to cooperatively multitask multiple applications (MultiFinder); the second was the transition from 68000 to PowerPC processors, the third was the coming together of Mac OS design ideas with a Unix-derived infrastructure in Mac OS X. Mac OS has very high unifying idea significantly different from that of Unix's, this is the Mac Interface Guidelines. These explain a great detail of what an Apps Graphical User Interface is supposed to depict with its expected behavior. One major idea of the Mac Interface Guidelines is that everything should stay where they are kept. Mac operating system apps are termed not huge monoliths. The system's graphic user interface (GUI) supports program instructions or codes, which are partially implemented in a ROM conveyed with the hardware and partly implemented in shared libraries, communicates easily with Mac OS software programs through a quite stable event interface. Hence, the operating system design encourages a distinct and clean separation between the GUI interface and the application engine. Leading-edge Unix's like Linux OS is beginning to borrow ideas like file attributes from Mac OS.

Merits of Mac OS

- Viruses: Apple Macs get almost no viruses. This is because Windows has a very large and superior market share over other OS.
- Reliability: Apple computers offer themselves for Macs to run only on it, and hence less prone to the crashing of hardware and software.
- Looks: often time, Mac seems to look better than its counterpart, windows OS.

Demerits of Mac OS

- Expensive: The cost of purchase of Mac is more than that of Windows.
- Only available on Apple computers: Already having a computer system that is not an Apple, one will not be able to install MAC in such a system. Otherwise, one will need to purchase a new computer system.
- Compatibility: Very few programs can only run on MAC OS, likewise computer games.

DIFFERENCES

- **Cost/Price:** A significant factor to consider is pricing, and as such, it kicks off this list. The macOS is strictly for Apple's hardware, the Macintosh PCs. And depending on the model, the price of a new Macintosh could run into thousands. Windows, on the other hand, can run on external hardware of several manufacturers. Also, Windows and its associated hardware are relatively less expensive. However, you'd still exchange a few hundred bucks for a decent Windows PC. As part of GNU's software licensing, Linux is entirely free to download, alter, and redistribute!
- **Gaming:** There's a vast collection of games available for Windows, majorly due to its wide usage. If you're a gamer, that's good news. Windows-based PCs also benefit from a wide range of graphics cards and gaming hardware upgrades. There are games for Mac, but it doesn't compare to that of Windows. As for Linux, the available games are relatively few.
- **Ease of use:** Setting up and getting started with Windows is pretty much straightforward. It's one of the reasons why you'll find Windows PCs in several homes and businesses. Mac is easier to use compared to Linux, especially for first-time users.
- **Security/Virus Proof:** Asides from the regularly provided security patches and updates from Apple, macOS is less prone to security attacks due to its closed source software licensing. Due to its open-source licensing, Linux is more vulnerable than a Mac but has less malware developed for it as it's less popular. However, being open-source, Linux has a large community base to help anytime a security breach occurs. Of the three systems, Windows is the most prone to virus attacks due to its popularity among users. And as you'd expect, there are large numbers of malware developed for it. Notwithstanding, several free and paid anti-viruses offer security for Windows users.
- **Percentage of Usage/Popularity:** In a published report by Statista, Windows has 70.92% of the market share for laptop, tablet, and console operating systems making it

the most popular OS worldwide. macOS follows with approximately 16% market share and lastly Linux with about 2%.

- **Hardware Support:** macOS shines in this category as it is only available on Apple hardware. As a result, it offers a reliable hardware-software integration that delivers the best results. Linux and Windows, however, can be used on a wide range of computer hardware. Although, stability issues may arise as a result of differences in hardware configurations.
- **User Target Group:** Windows is relatively cheaper than the Mac, so it targets people of all ages and socio-economic classes, making it suitable for many homes and businesses. Mac is more prevalent among creatives – video editors, graphic designers, and animators. For people looking to spend more on something classy and different, Mac is a great option. Linux generally finds popularity among techies – developers and programmers. And that's because Linux is open source, doesn't track its users' digital footprint, and provides a graphical user interface (GUI) or a command-line interface (CLI) for simple and advanced customization.
- **Where they are used:** Desktops, laptops, smartphones, and other devices run on the Windows operating system. Computers, servers, and several embedded systems use Linux. Mac is primarily used on desktops and not recommended for servers due to its high cost.