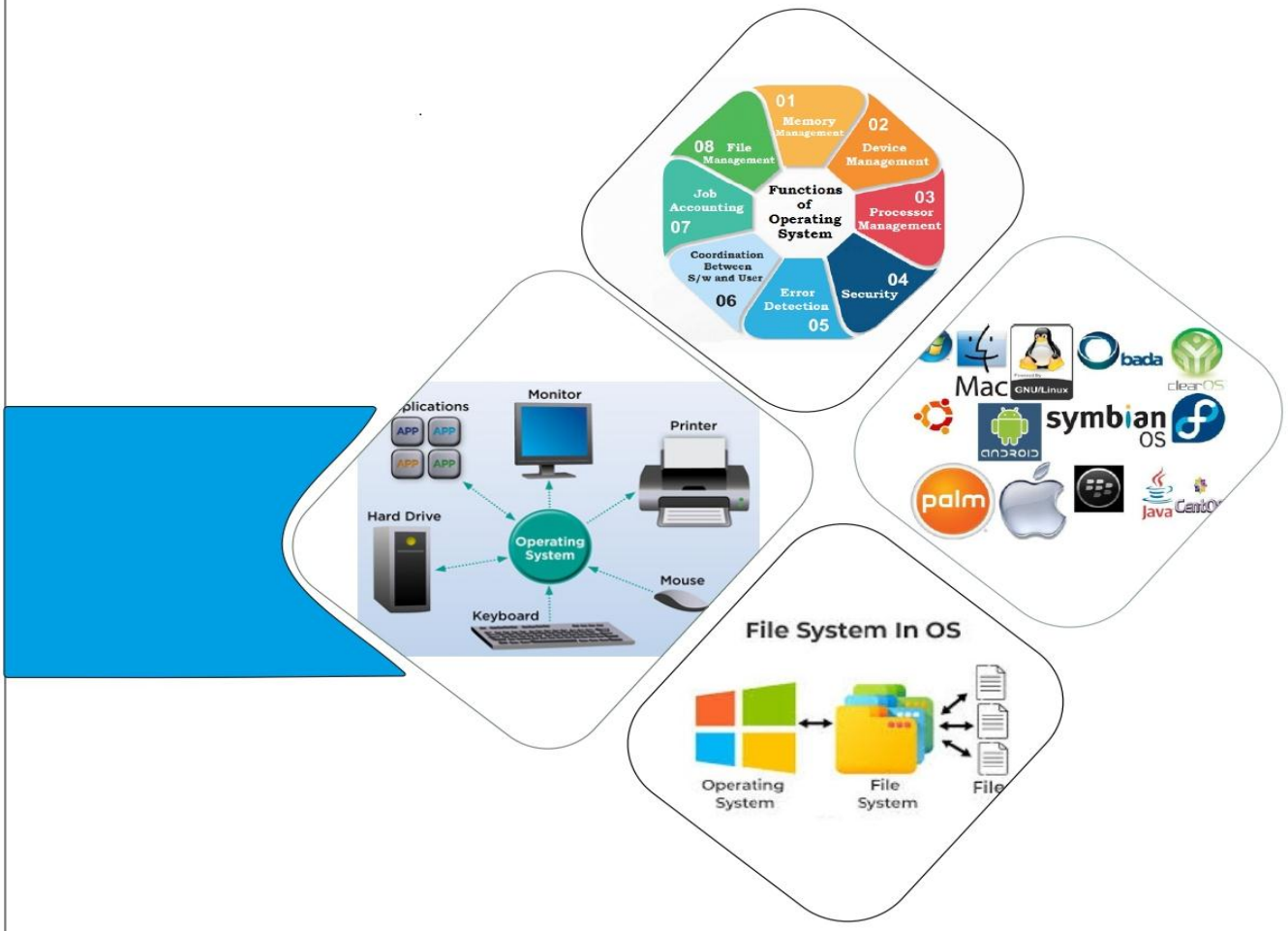
SCHEME :K

Name :_____
Roll No.:_____ Year : 20 ___ 20 __
Exam Seat No. :_____

LABORATORY MANUAL FOR

OPERATING SYSTEM (315319)



COMPUTER ENGINEERING GROUP

MAHARASHTRA STATE BOARD OF TECHNICAL EDUCATION, MUMBAI
(Autonomous)(ISO21001:2018)(ISO/IEC27001:2013)

Vision

To ensure that the Diploma level Technical Education constantly matches the latest requirements of Technology and industry and includes the all-round personal development of students including social concerns and to become globally competitive, technology led organization.

Mission

We, at MSBTE are committed to offer the best in class academic services to the students and institutes to enhance the delight of industry and society. This will be achieved through continual improvement in management practices adopted in the process of curriculum design, development, implementation, evaluation and monitoring system along with adequate faculty development programmes

Core Values

MSBTE believes in the following:

- Skill development in line with industry requirements
- Industry readiness and improved employability of Diploma holders
- Synergistic relationship with industry
- Collective and Cooperative development of all stake holders
- Technological interventions in societal development
- Access to uniform quality technical education

A Practical Manual

for

Operating System

(315319)

Semester-V

AI/AN/ BD/ CM/ CO/ CW/ DS/ HA/ IF/ IH/ SE

Maharashtra State Board of Technical Education, Mumbai

(Autonomous) (ISO 21001:2018)(ISO/IEC 27001:2013)

'K' Scheme Curriculum

MAHARASHTRA STATE BOARD OF TECHNICAL EDUCATION

(Autonomous) (ISO 21001:2018)(ISO/IEC 27001:2013)

Address: 4th floor, Govt. Polytechnic Building, 49,

Kherwadi, Bandra (E), Mumbai- 400 051

Tel: 022 62542100

Email: secretary@msbte.com

Maharashtra State Board of Technical Education

Certificate

This is to certify that Mr./ Ms ……

Roll No.……..,of……Semester of Diploma in……..…….....of **Institute Name** ……..……...…….(**Inst.Code:**……..……...) has completed the term work satisfactorily in Course **Operating System (Course Code:315319)** for the academic year 20…….. to 20…….. as prescribed in the curriculum.

Place ……. Enrollment No……

Date: …..... Exam Seat No. ……....

Course Teacher **Head of the Department** **Principal**

Seal of the Institute

Preface

The primary focus of any engineering laboratory/field work in the technical education system is to develop the much needed industry relevant competencies and skills. With this in view, MSBTE embarked on this innovative 'K' Scheme curricula for engineering Diploma programmes with outcome-based education as the focus and accordingly, relatively large amount of time is allotted for the practical work. This displays the great importance of laboratory work making each teacher, instructor and student to realize that every minute of the laboratory time need to be effectively utilized to develop these outcomes, rather than doing other mundane activities. Therefore, for the successful implementation of this outcome-based curriculum, every practical has been designed to serve as a 'vehicle' to develop this industry identified competency in every student. The practical skills are difficult to develop through 'chalk and duster' activity in the classroom situation. Accordingly, the “K” scheme laboratory manual development team designed the practical's to focus on outcomes, rather than the traditional age old practice of conducting practical's to verify the theory (which may become a by product along the way).

This laboratory manual is designed to help all stakeholders, especially the students, teachers and instructors to develop in the student the pre-determined outcomes. It is expected from each student that at least a day in advance, they have to thoroughly read the concerned practical procedure that they will do the next day and understand minimum theoretical background associated with the practical. Every practical in this manual begins by identifying the competency, industry relevant skills, course outcomes and practical outcomes which serve as a key focal point for doing the practical. Students will then become aware about the skills they will achieve through procedure shown there and necessary precautions to be taken, which will help them to apply in solving real-world problems in their professional life.

This manual also provides guidelines to teachers and instructors to effectively facilitate student-centred lab activities through each practical exercise by arranging and managing necessary resources in order that the students follow the procedures and precautions systematically ensuring the achievement of outcomes in the students.

Operating systems are an essential part of any computer system. Similarly, a course on operating systems is an essential part of any computer group. We hope that students will also find it useful. It provides a clear description of practical performance, execution and working of Operating System.

Although all care has been taken to check for mistakes in this laboratory manual, yet it is impossible to claim perfection especially as this is the first edition. Any such errors and suggestions for improvement can be brought to our notice and are highly welcome.

Program Outcomes (POs) to be achieved through Course:

PO1	Basic and Discipline specific knowledge: Apply knowledge of basic mathematics, science and engineering fundamentals and engineering specialization to solve the engineering problems.
PO2	Problem analysis: Identify and analyses well-defined engineering problems using codified standard methods.
PO3	Design/ development of solutions: Design solutions for well-defined technical problems and assist with the design of systems components or processes to meet specified needs.
PO4	Engineering Tools, Experimentation and Testing: Apply modern engineering tools and appropriate technique to conduct standard tests and measurements.
PO5	Engineering practices for society, sustainability and environment: Apply appropriate technology in context of society, sustainability, environment and ethical practices.
PO6	Project Management: Use engineering management principles individually, as a team member or a leader to manage projects and effectively communicate about well-defined engineering activities.
PO7	Life-long learning: Ability to analyses individual needs and engage in updating in the context of technological changes.

List of Relevant Skills

The following industry relevant skills of the competency “Manage operations of operating System” are expected to be developed in you by performing practical’s of this laboratory manual.

1. Execute system call and process related commands to work with operating System like LINUX.
2. Execute Program on CPU Scheduling algorithms.
3. Execute basic memory management commands.
4. Execute Program on Page replacement algorithms.

Practical Course Outcome Matrix

Course Outcomes (COs)

CO1	Explain the services and components of an Operating System.
CO2	Describe the different aspects of Process Management in an Operating System.
CO3	Implement various CPU Scheduling algorithms and evaluate their effectiveness.
CO4	Analyze the Memory Management techniques used by an Operating System.
CO5	Apply techniques for effective File Management in an Operating System.

Sr. No.	Title of the Experiment	CO1	CO2	CO3	CO4	CO5
1	* System call commands in Linux such as fork(), exec(), getpid, pipe, exit, open, close, stat, uname.	✓				
2	* Process related commands in Linux - top, ps, kill, wait, sleep, nice, renice,bg,fg.		✓			
3	* a. Commands for Sending Messages to Logged-in Users -who, cat, wall, write, mesg. * b. List Processes Attached to a Shared Memory Segment: ipcs.		✓			
4	* Write a C/Python program to calculate average waiting time and Turnaround Time of n processes with First Come First Serve (FCFS) CPU scheduling algorithm.			✓		
5	Write a C/Python program to calculate average waiting time and Turnaround Time of n processes with Shortest Job First (SJF) CPU scheduling algorithm.			✓		
6	Write a C/Python program to calculate average waiting time and Turnaround Time of n processes with Priority CPU scheduling algorithm.			✓		
7	Write a C/Python program to calculate average waiting time and Turnaround Time of n processes with Round Robin (RR) CPU scheduling algorithm.			✓		
8	Write a C/Python program to implement Banker's Algorithm.			✓		

9	Basic memory management commands - df, free, vmstat, /proc/meminfo, htop.				✓	
10	* Write a C/Python program on First In First Out (FIFO) Page Replacement algorithm.				✓	
11	Write a C/Python program on Least Recently Used (LRU) Page Replacement algorithm.				✓	
12	* Write a C/Python program on sequential file allocation method.					✓

Guidelines to Teachers

1. Teachers should align the explanation of the topic to teaching learning outcome (TLOs).
2. Refer to laboratory learning outcome (LLOs) for the execution of the practical to focus on the defined objectives.
3. Promote life-long learning by training the students to equip themselves with essential knowledge, skills and attitudes.
4. If required, provide demonstration for the practical emphasizing on the skills that the student should achieve.
5. Teachers should give opportunity to the students for exhibiting their skills after the demonstration.
6. Provide feedback and/or suggestions and share insights to improve effectiveness.
7. Assess students' skill achievement related to COs of each unit.

Instructions for Students

1. 100% attendance is compulsory for all practical sessions.
2. Students must adhere to ethical practices.
3. All the students must follow the schedule of practical sessions, complete the assigned work/activity and submit the assignment in stipulated time as instructed by the course teacher.
4. Students shall listen carefully the lecture given by teacher about importance of subject, learning structure, course outcomes.
5. Students shall understand the purpose of experiment and its practical implementation.
6. Students shall write the answers of the questions during practical.
7. Student should feel free to discuss any difficulty faced during the conduct of practical.
8. Student shall attempt to develop related hands-on skills and gain confidence.
9. Students should develop habit to submit the write-ups on the scheduled dates and time.

Content Page

List of Practical and Formative Assessment Sheet

Sr. No	Practical Title	Date of Performance	Date of Submission	Assessment Marks (25)	Teacher's Sign	Remark
1	* System call commands in Linux such as fork (), exec (), getpid, pipe, exit, open, close, stat, uname.					
2	* Process related commands in Linux - top, ps, kill, wait, sleep, nice, renice,bg,fg.					
3	* a. Commands for Sending Messages to Logged-in Users - who, cat, wall, write, mesg. * b. List Processes Attached to a Shared Memory Segment: ipcs.					
4	* Write a C/Python program to calculate average waiting time and Turnaround Time of n processes with First Come First Serve (FCFS) CPU scheduling algorithm.					
5	Write a C/Python program to calculate average waiting time and Turnaround Time of n processes with Shortest Job First (SJF) CPU scheduling algorithm.					
6	Write a C/Python program to calculate average waiting time and Turnaround Time of n processes with Priority CPU scheduling algorithm.					
7	Write a C/Python program to calculate average waiting time and Turnaround Time of n processes with Round Robin (RR) CPU					

	scheduling algorithm.					
8	Write a C/Python program to implement Banker's Algorithm.					
9	Basic memory management commands - df, free, vmstat, /proc/meminfo, htop.					
10	* Write a C/Python program on First In First Out (FIFO) Page Replacement algorithm.					
11	Write a C/Python program on Least Recently Used (LRU) Page Replacement algorithm.					
12	* Write a C/Python program on sequential file allocation method.					
				Total		

***Total marks to be transferred to proforma published by MSBTE**

Note:

- '*' Marked Practical's (LLOs) Are mandatory.
- Minimum 80% of above list of lab experiment are to be performed.
- Judicial mix of LLOs are to be performed to achieve desired outcomes.

Practical No. 1: System call commands in Linux such as fork(), exec(), getpid, pipe, exit, open, close, stat, uname.

I Practical Significance

System calls act as a bridge between user space and kernel space, allowing programs to interact with hardware and core OS features in a controlled manner. System calls are essential for interacting with the Linux kernel. They allow programs to perform operations that require higher privileges or direct interaction with system hardware and resources.

II Industry / Employer Expected Outcome(s)

1. Understanding how user applications interact with the kernel.
2. Ability to create, manages, and terminates processes.
3. Understanding of how the OS controls access to resources
4.

III Course Level Learning Outcomes(s)

CO1 – Explain the services and components of an Operating System.

IV Laboratory Learning Outcome(s)

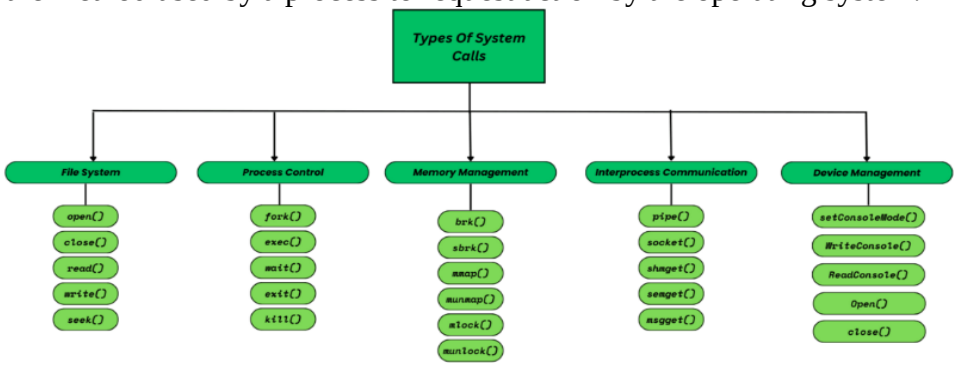
LLO 1.1 Execute the system call commands.

V Relevant Affective Domain related Outcomes

1. Follow precautionary measures.
2. Follow naming conventions
3. Follow ethical practices.

VI Relevant Theoretical Background

System calls provide the means for a user program to ask the operating system to perform tasks reserved for the operating system on the user program's behalf. A system call is invoked in a variety of ways, depending on the functionality provided by the underlying processor. In all forms, it is the method used by a process to request action by the operating system.



1. **fork()**
   A new process is created by the fork() system call. A new process may be created with fork() without a new program being run-the new sub-process simply continues to execute exactly the same program that the first (parent) process was running. It is one of the most widely used system calls under process management.
2. **exec()**
   A new program will start executing after a call to exec().Running a new program does not require that a new process be created first: any process may call exec() at any time.

The currently running program is immediately terminated, and the new program starts executing in the context of the existing process.

3. **Getpid**

   getpid stands for Get the Process ID. The getpid() function shall return the process ID of the calling process. The getpid() function shall always be successful and no return value is reserved to indicate an error.

4. **Pipe**

   The pipe() system call is used to communicate between different Linux processes. It is mainly used for inter-process communication. The pipe() system function is used to open file descriptors.

5. **Exit**

   The exit() system call is used by a program to terminate its execution. The operating system reclaims resources that were used by the process after the exit() system call.

6. **Open**

   It is the system call to open a file. This system call just opens the file, to perform operations such as read and write, we need to execute different system call to perform the operations.

7. **Close**

   This system call closes the opened file.

8. **Stat**

   stat' serves as a versatile utility that fetches and displays data about files and directories. From determining file types to monitoring file system usage, 'stat' helps users gain insights into various file attributes which are crucial for system management and file handling operations.

9. **Uname**

   Let's start with the basics. The term "uname" stands for "Unix Name," and the command itself is designed to provide you with key details about your Linux system. It's like asking your computer, "Hey, who are you, and what are you made of?" The answers you get can help you understand your system's kernel version, operating system, hardware architecture, and more. This command 'uname' displays the information about the system.

   Options and Examples of 'uname' Command in Linux

Options	Description
'-a' **or** '--all'	Displays all available information.
'-s' **or'**--kernel-name'	Shows the kernel name.
'-n' **or** '--nodename'	Displays the network (domain) name of the machine.
'-r' **or'**--kernel-release'	Shows the kernel release.
'-v' **or'**--kernel-version'	Displays the kernel version.
'-m' **or** '--machine'	Shows the machine hardware name.
'-p' **or** '--processor'	Displays the processor type or unknown."
'-i' **or** '--hardware-platform'	Shows the hardware platform or unknown."
'-o' **or** '--operating-system'	Displays the operating system.

VII Recourses Required:

- Computer System with basic configuration. Linux/Ubuntu/ CentOS /any other open source Operating System.
- Software: Turbo C/ vi editor/ Any open source Python IDE such IDLE, Jupyter Notebook, Spyder, PyCharm etc.)

VIII Precautions to be followed:

1. Handle computer system with care.
2. Strictly follow the instructions for writing, compiling, and executing the program.
3. Don't forget to save file before execution.
4. Follow safety Practices

IX Conclusion

……..

……

……

X Practical related questions

Note: Below given are few sample questions for reference. Teachers must design more such questions to ensure the achievement of identified CO.

1. Write use of system calls.
2. Write system calls related to Process management.
3. Write a command which returns a PID.
4. What are various options for uname.
5. Explain about exit command.
6. List any four system calls and write their use

Space for Answer

……..

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……..

……

……

……

……

XII References:

1. Linux system call in Detail | GeeksforGeeks
2. Stat command in Linux with examples | GeeksforGeeks
3. https://www.tutorialspoint.com/what-are-system-calls-in-operating-system

XIII Assessment Scheme (25 Marks)

S. No.	Weightage- Process related (Marks-10)	40%
1.	Logic Formation	20%
2.	Debugging Ability	10%
3.	Follow Ethical Practices:	10%
	Weightage- Product related (Marks-15)	**60%**
4.	Expected Output	25%
5	Timely Submission	25%
6.	Answer to Sample questions	10%
	Total (25 Marks)	**100%**

Marks Obtained			Dated Signature of Teacher
Process related (10)	**Product related (15)**	**Total(25)**	

Practical No. 2: Process related commands in Linux-top, ps, kill, wait, sleep, nice, renice, bg, fg.

I Practical Significance

Process is a program in execution. In Linux, process related commands execute through the command line interface (terminal).

II Industry / Employer Expected Outcome(s)

1. Able to execute process related command top, ps, kill, wait, sleep, nice, renice, bg, fg.

III Course Level Learning Outcomes(s)

CO2 – Describe the different aspects of process Management in an Operating System.

IV Laboratory Learning Outcome(s)

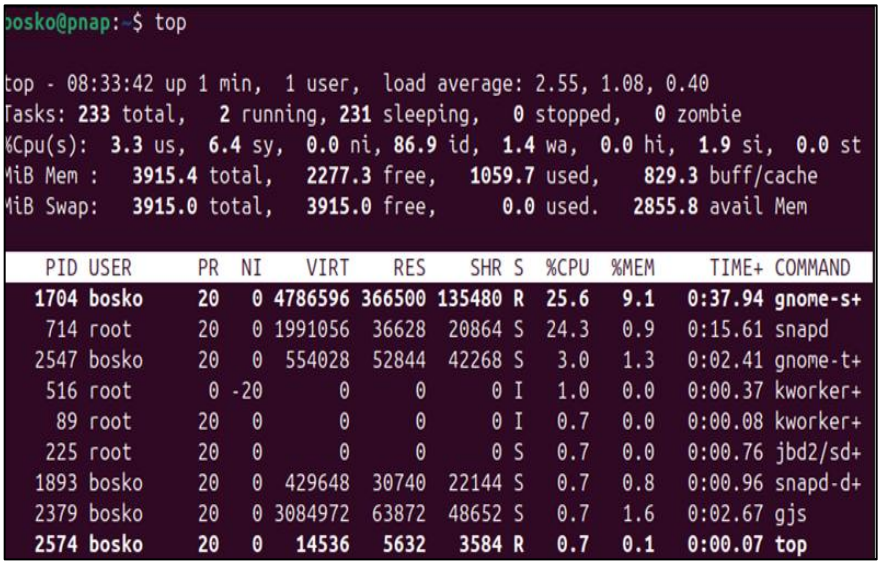
LLO 2.1 Execute process related commands

V Relevant Affective Domain related Outcomes

1. Follow precautionary measures.
2. Follow naming conventions
3. Follow ethical practices.

VI Relevant Theoretical Background

1. top: The top (table of processes) command shows a dynamic, real-time view of running processes and kernel-managed tasks in Linux. The command also provides a system information summary that shows resource utilization, including CPU and memory usage.
Syntax : top [option]



The table below lists the most commonly used top command options:
Option Description

Option	Description
-h	Help
-v	Version
-b	Batch-mode operation
-c	Command-line/Program-name toggle
-H	Thread-mode operation
-i	Idle-process toggle
-s	Secure-mode operation
-S	Cumulative-time toggle
-d	Delay-time
-n	Number of iterations
-u	User-filter-mode
-U	User-id-or-name
-p	Monitor-PIDs mode
-w	Output-width-override

2.ps : **ps (process status)** command is used to display information about the current running processes on the system.

Syntax : ps [options]

Option	Description
$ps -f	Displays a full-format listing of all running processes on your system.
$ps–u username	Displays processes of user 'username'
$ps –a	Processes of all users
$ps –e	Processes including user and system processes.

3. kill : kill command which is used to terminate processes manually.

Syntax : kill [signal]PID

$kill 0	Kills all the processes on the terminal except the login shell by special argument '0'
$kill 120 230 234	Kills three processes with pid 120 230 234
$kill -9 0	Kills all processes including login shell
$kill -9 $$	Kills login shell

4. wait: wait command waits for a process to finish and it returns the termination status.

Syntax: wait [process_id]

Eg : wait 1123

Option	Description
wait -n	Waits for only the following background process to complete and returns an exit status.
wait -f	Terminating a program first waits for the background task to finish before quitting.

5. sleep: Used to execute commands after certain amount of time by sleeping for given seconds

Syntax: sleep NUMBER[SUFFIX]

NUMBER represents the time duration for which the command should sleep.
SUFFIX can be used to specify the unit of time (s for seconds, m for minutes, h for hours, etc.)
Ex. sleep 30

6.nice: This command is used to start a new process with a specific priority, known as the "nice value." A higher nice value lowers the process's priority, while a lower (negative) nice value increases it. Processes with higher priority receive more CPU time.
How Can You Check a Running Process's Nice Value?
To check the nice value of a running process, you can make use of the "ps" command with the "-l" option. Open a terminal and execute the following command:
ps -l <process_id>

The nice value will be displayed under the "NI" column.

How to Set the Priority of a Process?
To set the priority of a process, you can use the nice command in linux along with the desired nice value.
Eg: nice -11 gnome-terminal

How to set the negative priority for a process?
Eg: nice - -11 gnome-terminal

7.renice: Unlike nice, which sets the priority when starting a process, renice modifies the priority of an already running process. This flexibility allows system administrators to manage process priorities based on the current system load dynamically.
The Renice command can be used to alter a process's priority while it is already running. Open a terminal and execute the following command:

Syntax: renice <nice_value> -p <process_id>

Replace <nice_value> with the new priority level you want to assign and <process_id> with the ID of the running process you wish to modify.

For example, to increase the priority of a process with **ID 1234 to 5**, you would run:
renice 5 -p 1234

8.bg command: The bg command is a useful tool that allows you to manage and move processes between the foreground and background. It's especially helpful when you want to multitask in the terminal by placing a process in the background, enabling you to continue using the terminal for other commands while the process runs quietly in the background.

Syntax: bg [job_spec]

where,
job_spec: This is used to identify the job you want to move to the background. It can be specified in several formats:

Options	Description
%n	Refers to job number n.
%str	Refers to a job that was started by a command beginning with str.
%?str	Refers to a job that was started by a command containing str.
%% or %+	Refers to the current job. Both fg and bg commands will operate on this job if no job_spec is provided.
%-	Refers to the previous job

If no job_spec is provided, the most recent job is resumed in the background.

9. fg command: The fg command is used to bring a background job into the foreground. It allows you to resume a suspended job or a background process directly in the terminal window, so you can interact with it.
Syntax:
fg [job_spec]
The **job_spec** is a way to refer to the background jobs that are currently running or suspended. Here are some common ways to specify a job:

%n	Refers to job number n
%str	Refers to a job that was started by a command beginning with str.
%?str	Refers to a job that was started by a command containing str.
%% or %+	• Refers to the current job (this is the default job operated on by fg if no job_spec is provided).
%-	Refers to the previous job.

VII Recourses Required:

- Computer System with basic configuration. Linux/Ubuntu/ CentOS /any other open source Operating System.

VIII Precautions to be followed:

1. Handle computer system with care.
2. Strictly follow the instructions for writing commands
3. Follow safety Practices

IX Conclusion

……

……

……

X Practical related questions

Note: Below given are few sample questions for reference. Teachers must design more such questions to ensure the achievement of identified CO.

1. What is difference between sleep and wait?
2. Write various signal names along with its signal number used in kill command?
3. What is name of your login shell?
4. List the system calls for process management.
5. Explain nice and renice command
6. What are various options of ps command?
7. What is process ID of your login shell?

Space for Answer

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

XI References:

1.https://www.geeksforgeeks.org/processes-in-linuxunix/
2.https://www.digitalocean.com/community/tutorials/linux-commands
3.https://www.redhat.com/en/blog/linux-command-basics-7-commands-process-management

XII Assessment Scheme (25 Marks)

S. No.	Weightage- Process related (Marks-10)	40%
1.	Logic Formation	20%
2.	Debugging Ability	10%
3.	Follow Ethical Practices:	10%
	Weightage- Product related (Marks-15)	**60%**
4.	Expected Output	25%
5	Timely Submission	25%
6.	Answer to Sample questions	10%
	Total (25 Marks)	**100%**

Marks Obtained			Dated Signature of Teacher
Process related (10)	**Product related (15)**	**Total(25)**	

Practical No. 3: a. Commands for Sending Messages to Logged-In Use who,cat,wall,write, mesg.
b. List Processes Attached to a Shared Memory Segment: ipcs.

I Practical Significance

The commands who, cat, wall, write, and mesg are Linux terminal commands that are used for communication between logged-in users on a multi-user system. Their practical significance lies in system administration, collaborative work, and basic communication on shared systems. **T**he **ipcs** command is a utility for displaying information on **inter-process communication** (IPC) mechanisms.

II Industry / Employer Expected Outcome(s)

1. Able to execute commands for sending messages to logged in user.
2. Able to execute ipcs command.

III Course Level Learning Outcomes(s)

CO2 – Describe the different aspects of process Management in an Operating System.

IV Laboratory Learning Outcome(s)

LLO 3.1 Execute Message Passing and Shared Memory commands.

V Relevant Affective Domain related Outcomes

1. Follow precautionary measures.
2. Follow naming conventions
3. Follow ethical practices

VI Relevant Theoretical Background

a. Commands for Sending Messages to Logged-In User

1. **who:** It is used to display who are the users connected to our computer currently
Syntax :who [options] [filename]

Option	Description
-a	Same as -b -d –login -p -r -t -T -u. Displays a more detailed output, including idle time, process IDs, and other information related to the login session.
-b	Shows the time of the last system boot.
-d	Displays dead processes that are no longer running.
-H	Print line of column headings
-l	Print system login processes
-m	Show the host name and user associated with standard input (stdin)
-p	Shows active processes spawned by init
-q	Displays only the logged-in users and their count.
-r	Displays the current system run level.
-s	Shows the default short who command output.
-T	Shows user's message status as + (messages allowed), – (messages not allowed), or ? (cannot find terminal device).

2. cat

It is used to create the file and displaying the contents of file/files.

- **To create new file**
  syntax: cat > newfile_name
  eg: cat>sample.txt
  where cat is a command and sample.txt is argument.

- **To View the Content of a Single File** in Linux
  syntax: cat file_name
  eg: cat sample.txt

- **To View the Content of Multiple Files** in Linux
  User can display contents of more than one file and called as concatenation.
  syntax: cat file_name1 file_name2
  eg: cat sample1.txt sample2.txt

Option	Description
-n	To see the line number
-b	To see the line number(except blank lines)
-s	Remove extra blank lines

3. wall:

wall command is used to write a message to all users. This command displays a message, or the contents of a file, or otherwise its standard input, on the terminals of all currently logged in users.

Syntax:

wall [-n] [-t timeout] [message | file]

where,

- **message:** The string of text you want to broadcast to all users. You can directly type the message or redirect it from a file.
- **file:** A file containing the message to broadcast.
- **OPTION:** Various options that modify the behavior of the wall co
- mmand.

Option	Description
-n	Suppress the default banner, so the message is displayed without the standard header.
-t	This option will abandon the write attempt to the terminals after timeout seconds.
-v	display version information and exit
-h	Show help information about the **wall** command and its options, then exit.

4. **write:** The write command in Linux is used to **send a message to another logged-in user** via the terminal. This command is useful for quick communication between users on the

same system.

Option	Description
write username	Sends a message to the specified user's terminal.
write username tty	Sends a message to the specified user's terminal on a specific terminal device
Ctrl+D	Ends the message input session.
mesg y	Allows messages from other users (enables write access).
mesg n	Disallows messages from other users (disables write access).

5. **mesg:** The mesg command in Linux manages message-sending permissions for a user's terminal.
   Syntax: mesg [y | n]
   - **mesg y:**Allows all users (including other users on the local network) to send messages to your terminal.
   - **mesg n:**Denies all users (except root) the ability to send messages to your terminal.
   - **mesg:** (without any arguments) displays the current message permission setting (either is y or is n).

b. List Processes Attached to a Shared Memory Segment: ipcs.

Inter-Process Communication (IPC) is a fundamental concept in operating systems that allows multiple processes to communicate and synchronize their actions. **Inter-process Communication(IPC) through shared memory** is a concept where two or more processes can access the common memory and communication is done via this shared memory where changes made by one process can be viewed by another process. The Shared memory is a memory segment that multiple processes can access concurrently.

1. **ipcs command:** ipcs shows information on the inter-process communication facilities for which the calling process has read access.

Option	Description
-q	Write information about active message queues.
-m	Write information about active shared memory segments.
-s	Write information about active semaphore sets.
-a	Use all print options. (This is a shorthand notation for -b, -c, -o, -p, and -t.)
-b	Write information on maximum allowable size. (Maximum number of bytes in messages on queue for message queues, size of segments for shared memory, and number of semaphores in each set for semaphores.)

-c	Write creator's user name and group name
-o	Write information on outstanding usage. (Number of messages on queue and total number of bytes in messages on queue for message queues, and number of processes attached to shared memory segments.)
-p	Write process number information. (Process ID of the last process to send a message and process ID of the last process to receive a message on message queues, process ID of the creating process, and process ID of the last process to attach or detach on shared memory segments.)
-t	Write time information. (Time of the last control operation that changed the access permissions for all facilities, time of the last msgsnd() and msgrcv() operations on message queues, time of the last shmat() and shmdt() operations on shared memory, and time of the last semop() operation on semaphores.)

VII Recourses Required:

- Computer System with basic configuration. Linux/Ubuntu/ CentOS /any other open source Operating System.

VIII Precautions to be followed:

1. Handle computer system with care.
2. Strictly follow the instructions for writing commands
3. Follow safety Practices

IX Conclusion

…….

…….

…….

X Practical related questions

Note: Below given are few sample questions for reference. Teachers must design more such questions to ensure the achievement of identified CO.

1. What are the prerequisites for using write command?
2. Write a command to display the current run level using who command?
3. Write a command to append the contents of one file to another using cat Command?
4. Write a command to send a message from a file using wall command?
5. Write a command to check the current message permission status of your terminal?
6. What is the purpose of the ipcs command in Linux?

Space for Answer

……

……

……

……

……

……

……

……

XI References:

1. https://www.geeksforgeeks.org/linux-commands/
2. https://phoenixnap.com/kb/write-command-in-linux
3. https://www.geeksforgeeks.org/ipcs-command-linux-examples/
4. https://www.tutorialspoint.com/linux-who-command-with-examples
5. https://phoenixnap.com/kb/linux-who-comm

XII Assessment Scheme (25 Marks)

S. No.	Weightage- Process related (Marks-10)	40%
1.	Logic Formation	20%
2.	Debugging Ability	10%
3.	Follow Ethical Practices:	10%
	Weightage- Product related (Marks-15)	**60%**
4.	Expected Output	25%
5	Timely Submission	25%
6.	Answer to Sample questions	10%
	Total (25 Marks)	**100%**

Marks Obtained			Dated Signature of Teacher
Process related (10)	**Product related (15)**	**Total(25)**	

Practical No. 4: Write C/Python program to calculate average waiting time and turnaround time of n processes with First Come First Serve (FCFS) CPU scheduling algorithm.

I Practical Significance

CPU scheduling is a process used by the operating system to decide which task or process gets to use the CPU at a particular time. The main purpose of a CPU scheduling is to maximize the CPU utilization and minimize the response and waiting time of the process.

II Industry / Employer Expected Outcome(s)

1.Able to find out waiting time and turnaround time by using FCFS algorithm.

III Course Level Learning Outcomes(s)

CO3 – Implement various CPU Scheduling algorithms and evaluate their effectiveness.

IV Laboratory Learning Outcome(s)

LLO 4.1 Implement First come First Serve (FCFS) Scheduling algorithm.

V Relevant Affective Domain related Outcomes

1. Follow precautionary measures.
2. Follow naming conventions
3. Follow ethical practices

VI Relevant Theoretical Background

Proposition 1:
Different Types of CPU Scheduling Algorithms

There are mainly two types of scheduling methods:

- **Preemptive Scheduling:** Preemptive scheduling allows a running process to be interrupted and replaced by another process. Preemptive scheduling is used when a process switches from running state to ready state or from the waiting state to the ready state.
- **Non-Preemptive Scheduling:** Non-Preemptive scheduling is used when a process terminates, or when a process switches from running state to waiting state.
  Non-preemptive algorithms are designed so that once a process enters the running state, it cannot be preempted until it completes its allotted time, whereas the preemptive scheduling is based on priority where a scheduler may preempt a low priority running process anytime when a high priority process enters into a ready state.

Proposition 2:
Turnaround time. The interval from the time of submission of a process to the time of completion is the Turnaround Time. Turnaround time is the sum of the periods spent waiting to get into memory, waiting in the ready queue, executing on the CPU, and doing I/O.
Turnaround Time(TAT) = Completion Time(CT)–Arrival Time(AT)

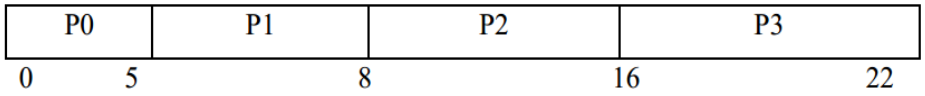
Proposition 3:
Waiting time. The CPU scheduling algorithm does not affect the amount of time during which a process executes or does I/O; it affects only the amount of time that a process spends waiting

in the ready queue. Waiting time is the sum of the periods spent waiting in the ready queue.
Waiting Time(WT) = Turnaround Time(TAT)– Burst Time(BT)

Proposition 4:
First-Come, First-Served (FCFS) Scheduling
FCFS is non-preemptive algorithm. In this scheduling algorithm, the process that requests the CPU first is allocated the CPU first. FCFS easily be implemented with a FIFO queue. When a process enters the ready queue, its PCB is linked onto the tail of the queue. As other jobs come in, they are put onto the end of the queue. When running process is blocked by any reason, the first process from the queue will be executed next. When a blocked process becomes ready, it is put at the end of the queue.
Example of FCFS to calculate average waiting time and turnaround Time.

Process	Arrival Time	CPU burst Time (ms)
P0	0	5
P1	1	3
P2	2	8
P3	3	6

The Gantt Chart is



Completion Time(CT): P0 =5, P1 =8, P2 = 16, P3 =22

Waiting Time and Turnaround Time of each process is as follows:

Process	Arrival Time (AT)	CPU burst Time (ms) (BT)	Turnaround Time(CT-AT)	Waiting Time (TAT-BT)
P0	0	5	5-0=5	5-5=0
P1	1	3	8-1=7	7-3=4
P2	2	8	16-2=14	14-8=6
P3	3	6	22-3=19	19-6=13

Average Turnaround time: (5+7+14+19) / 4 = **11.25 ms**
Average Wait Time: (0+4+6+13) / 4 = **5.75 ms**

VII Recourses Required:

- Computer System with basic configuration. Linux/Ubuntu/ CentOS /any other open source Operating System.
- Software: Turbo C/ vi editor/ Any open source Python IDE such IDLE, Jupyter Notebook, Spyder, PyCharm etc.)

VIII Precautions to be followed:

1. Handle computer system with care.
2. Strictly follow the instructions for writing, compiling, and executing the program.
3. Don't forget to save file before execution.
4. Follow safety Practices

IX **Program Code:** Consider the processes P1,P2, P3,P4 given in the below table, arrives for execution in the same order, with Arrival Time 0,and given Burst Time.
Implement First come first serve Scheduling algorithm to calculate find Average waiting time and Average turnaround time.

Process	Burst Time
P1	21
P2	6
P3	3
P4	2

Note: Attach the code at the end.
(This practical can be performed in any of the compiler like Turbo C/ vi editor etc or Python Interpreter / IDE like any open source IDE such IDLE, Jupyter Notebook, Spyder, PyCharm etc.)

X **Conclusion**

……

……

……

XI **Practical related questions**

Note: Below given are few sample questions for reference. Teachers must design more such questions to ensure the achievement of identified CO.

1. State the conditions for preemptive and non-preemptive scheduling algorithm.
2. State the disadvantages of FCFS.
3. Define Turnaround Time. Write a formula for average Turnaround Time.
4. Define Waiting Time. Write a formula for average Waiting Time.
5. The jobs are scheduled for execution as follows:

Solve the problem by using FCFS .Find Average waiting time and Average turnaround time using Gantt chart.

Process	Arrival Time	CPU burst Time (ms)
P0	0	10
P1	1	4
P2	2	14
P3	3	8

Space Answer

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

XII References:

1. https://www.geeksforgeeks.org/first-come-first-serve-cpu-scheduling- non- preemptive/
2. https://www.tutorialspoint.com/fcfs-scheduling
3. https://takeuforward.org/operating-system/fcfs-scheduling-explained/

XIII Assessment Scheme (25 Marks)

S. No.	Weightage- Process related (Marks-10)	40%
1.	Logic Formation	20%
2.	Debugging Ability	10%
3.	Follow Ethical Practices:	10%
	Weightage- Product related (Marks-15)	**60%**
4.	Expected Output	25%
5	Timely Submission	25%
6.	Answer to Sample questions	10%
	Total (25 Marks)	**100%**

Marks Obtained			Dated Signature of Teacher
Process related (10)	**Product related (15)**	**Total(25)**	

Practical No. 5: Write C/Python program to calculate average waiting time and Turnaround Time of n processes with Shortest Job First(SJF) CPU scheduling algorithm.

I Practical Significance

Shortest Job First (SJF) is a scheduling process that selects the waiting process with the smallest execution time to execute next. This scheduling algorithm significantly reduces the average waiting time for other processes waiting to be executed.

II Industry / Employer Expected Outcome(s)

1. Able to find out waiting time and turnaround time by using SJF CPU Scheduling algorithm.

III Course Level Learning Outcomes(s)

CO3 – Implement various CPU Scheduling algorithms and evaluate their effectiveness

IV Laboratory Learning Outcome(s)

LLO 5.1 Implement Shortest Job First (SJF) Scheduling algorithm.

V Relevant Affective Domain related Outcomes

1. Follow precautionary measures.
2. Follow naming conventions
3. Follow ethical practices

VI Relevant Theoretical Background

This scheduling method may or may not be preemptive.

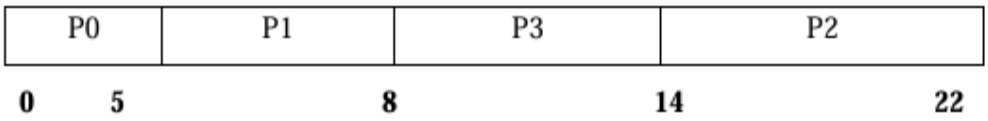
- **SJF non-preemptive scheduling:**
  In the SJF non-preemptive scheduling, once a process is assigned to the CPU, it runs into completion. Here, the short term scheduler is invoked when a process completes its execution or when a new process arrives in an empty ready queue.
- **SJF Preemptive scheduling:**
  This is the preemptive version of SJF scheduling and is also referred as Shortest Remaining Time First(SRTF) scheduling algorithm .If a short process enters ready queue while longer process is executing, process switch occurs by which the executing process is swapped out to the ready queue while the newly arrived shorter process starts to execute. Thus the short term scheduler is invoked either when a new process arrives in the system or an existing process completes its execution.

1. Non preemptive SJF

As an example, consider the following four processes, with the length of the CPU burst given in milliseconds:

Process	Arrival Time	CPU burst Time(ms)
P0	0	5
P1	1	3
P2	2	8
P3	3	6

The Gantt Chart is



Completion Time (CT): P0 =5, P1 =8, P2 = 22, P3 =14

Waiting Time and Turnaround Time of each process is as follows –

Process	Arrival Time (AT)	CPU burst Time(ms) (BT)	Turnaround Time (CT-AT)	Waiting Time (TAT-BT)
P0	0	5	5-0=5	5-5=0
P1	1	3	8-1=7	7-3=4
P2	2	8	22-2=20	20-8=12
P3	3	6	14-3=11	11-6=5

Average Turnaround time: (5+7+20+11) / 4 = 10.75 ms
Average Wait Time: (0 + 4 + 12 + 5)/4 = 21 / 4 = 5.25ms

2. Preemptive SJF or Shortest Remaining Time First

As an example, consider the following four processes, with the length of the CPU burst given in milliseconds:

Process	Arrival Time	CPU burst Time(ms)
P0	0	8
P1	1	4
P2	2	9
P3	3	5

The Gantt Chart is

Completion Time: P0 =17, P1 =5, P2 = 26, P3 =10

Waiting Time and Turnaround Time of each process is as follows –

Process	Arrival Time (AT)	CPU burst Time(ms) (BT)	Turnaround Time (CT-AT)	Waiting Time (TAT-BT)
P0	0	8	17-0=17	17-8=9
P1	1	4	5-1=4	4-4=0
P2	2	9	26-2=24	24-9=15
P3	3	5	10-3=7	7-5=2

Average Turnaround time: (17+4+24+7) / 4 = 13 ms
Average Wait Time: (9 + 0+ 15 + 2)/4 = 26 / 4 = 6.5ms

VII Recourses Required:

- Computer System with basic configuration. Linux/Ubuntu/ CentOS /any other open source Operating System.
- Software: Turbo C/ vi editor/ Any open source Python IDE such IDLE, Jupyter Notebook, Spyder, PyCharm etc.)

VIII Precautions to be followed:

1. Handle computer system with care.
2. Strictly follow the instructions for writing, compiling, and executing the program.
3. Don't forget to save file before execution.
4. Follow safety Practices

IX Program Code:

Implement the **Shortest Job First (SJF) CPU Scheduling Algorithm to calculate** average waiting time and average turnaround time

Note: Attach the code at the end.

(This practical can be performed in any of the compiler like Turbo C/ vi editor etc or Python Interpreter / IDE like any open source IDE such IDLE, Jupyter Notebook, Spyder, PyCharm etc.)

X Conclusion

…….

…….

…….

XI Practical related questions

Note: Below given are few sample questions for reference. Teachers must design more such questions to ensure the achievement of identified CO.

1. Explain the difference between preemptive and non-preemptive SJF.
2. What is the main advantage of SJF scheduling?
3. State disadvantages of SJF scheduling.
4. Consider the processes P1,P2 ,P3,P4 given in the below table. Find average waiting time and average turnaround time by using SJF(non-preemptive and preemptive) Scheduling algorithm.

Process	Burst Time	Arrival Time
P1	6	0
P2	3	1
P3	4	2
P4	10	3

Space for Answer

……

……

……

……

……

……

……

……

……

……

XII References:

1. https://www.tutorialspoint.com/operating_system/os_shortest_job_first_scheduling.htm
2. https://www.geeksforgeeks.org/program-for-shortest-job-first-or-sjf-cpu-scheduling-set-1-non-preemptive/
3. https://www.vbspu.ac.in/e-content/Scheduling-Algorithm.pdf

XII Assessment Scheme (25 Marks)

S. No.	Weightage- Process related (Marks-10)	40%
1.	Logic Formation	20%
2.	Debugging Ability	10%
3.	Follow Ethical Practices:	10%
	Weightage- Product related (Marks-15)	**60%**
4.	Expected Output	25%
5	Timely Submission	25%
6.	Answer to Sample questions	10%
	Total (25 Marks)	**100%**

Marks Obtained			Dated Signature of Teacher
Process related (10)	**Product related (15)**	**Total(25)**	

Practical No. 6: Write C/Python program to calculate average waiting time and Turnaround Time of n processes with Priority CPU scheduling algorithm.

I Practical Significance

Priority Scheduling is a fundamental scheduling algorithm used in operating systems to manage the execution of processes. Its **significance** lies in how it helps the OS make decisions about which process to run next, especially when resources are limited and multiple processes are competing for CPU time.

II Industry / Employer Expected Outcome(s)

1. Able to find out waiting time and turnaround time by using Priority CPU Scheduling algorithm.

III Course Level Learning Outcomes(s)

CO3 – Implement various CPU Scheduling algorithms and evaluate their effectiveness.

IV Laboratory Learning Outcome(s)

LLO 6.1 Implement Priority Scheduling algorithm.

V Relevant Affective Domain related Outcomes

1. Follow precautionary measures.
2. Follow naming conventions
3. Follow ethical practices

VI Relevant Theoretical Background

Priority scheduling is one of the most common scheduling algorithms used by the operating system to schedule processes based on their priority. Each process is assigned a priority value based on criteria such as memory requirements, time requirements, other resource needs, or the ratio of average I/O to average CPU burst time.

The process with the highest priority is selected for execution first. If there are multiple processes sharing the same priority, they are scheduled in the order they arrived, following a First-Come, First-Served approach. Some system represents low numbers to represent low priority while others use low numbers to represent high priority. The chosen process is then executed, either until completion or until it is preempted, depending on whether the scheduling is preemptive or non-preemptive.

Priority Scheduling can be implemented in two ways:

- Non-Preemptive Priority Scheduling
- Preemptive Priority Scheduling

Non-Preemptive Priority Scheduling

In Non-Preemptive Priority Scheduling, the CPU is not taken away from the running process. Even if a higher-priority process arrives, the currently running process will complete first.

Ex: A high-priority process must wait until the currently running process finishes.

Preemptive Priority Scheduling

In **Preemptive Priority Scheduling**, the CPU can be taken away from the currently running process if a new process with a higher priority arrives.

Ex: A low-priority process is running, and a high-priority process arrives; the CPU

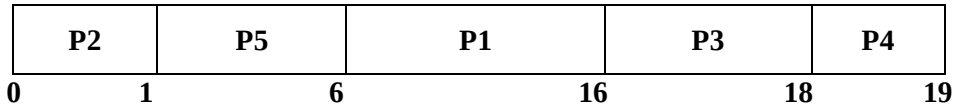
immediately switches to the high-priority process.

1. Non-Preemptive Priority Scheduling

As an example, consider the following set of processes, assumed to have arrived at time 0, in the order P1, P2,p3,p4 and P5, with the length of the CPU burst given in milliseconds:

Process	CPU Burst Time	Priority
P1	10	3
P2	1	1
P3	2	4
P4	1	5
P5	5	2

The Gantt Chart is



Completion Time: P1 =16, P2 =1, P3 = 18, P4 =19,P5= 6
Waiting Time and Turnaround Time of each process is as follows –

Process	Priority	CPU burst Time(ms) (BT)	Turnaround Time (CT-AT)	Waiting Time (TAT-BT)
P1	3	10	16-0=16	16-10=6
P2	1	1	1-0=1	1-1=0
P3	4	2	18-0=18	18-2=16
P4	5	1	19-0=19	19-1=18
P5	2	5	6-0=6	6-5=1

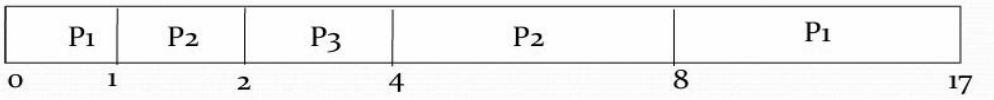
Average Turnaround time: (16+1+18+19+6) / 5 = 60/5=12 ms
Average Wait Time: (6+ 0+ 16 + 18+1)/5 = 41 / 5 = 8.2ms

2. Preemptive Priority Scheduling

As an example, consider the following set of processes in the order P1, P2,P3 with the length of the CPU burst given in milliseconds:

Process	CPU Burst Time	Priority	Arrival Time
P1	10	3	0
P2	5	2	1
P3	2	1	2

The Gantt Chart is



Completion Time: P1 =17, P2 =8, P3 = 4

Process	CPU Burst Time (BT)	Priority	Arrival Time (AT)	Turnaround Time (CT-AT)	Waiting Time (TAT-BT)
P1	10	3	0	17-0=17	17-10=7
P2	5	2	1	8-1=7	7-5=2
P3	2	1	2	4-2=2	2-2=0

Average Turnaround time: (17+7+2) / 3 =26/3= 8.6 ms
Average Wait Time: (7 + 2+ 0)/3 = 9 / 3 = 3ms

VII Recourses Required:

- Computer System with basic configuration. Linux/Ubuntu/ CentOS /any other open source Operating System.
- Software: Turbo C/ vi editor/ Any open source Python IDE such IDLE, Jupyter Notebook, Spyder, PyCharm etc.)

VIII Precautions to be followed:

1. Handle computer system with care.
2. Strictly follow the instructions for writing, compiling, and executing the program.
3. Don't forget to save file before execution.
4. Follow safety Practices

IX Program Code:

Implement the Priority CPU Scheduling Algorithm to calculate average waiting time and average turnaround time.

Note: Attach the code at the end.
(This practical can be performed in any of the compiler like Turbo C/ vi editor etc or Python Interpreter / IDE like any open source IDE such IDLE, Jupyter Notebook, Spyder, PyCharm etc.)

X Conclusion

…….

…….

…….

XI Practical related questions

Note: Below given are few sample questions for reference. Teachers must design more such questions to ensure the achievement of identified CO.

1. What is priority CPU scheduling?
2. How is the priority of a process determined in a priority scheduling algorithm?
3. What problem can arise in priority scheduling and how can it be resolved?
4. Consider the processes P1,P2 ,P3 given in the below table. Find average waiting time and average turnaround time by using priority (non-preemptive and preemptive) Scheduling algorithm**.**

Process	Burst Time	Arrival Time	Priority
P1	6	0	2
P2	4	1	3
P3	5	2	1

Space for Answer

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

XII References:

1. https://www.geeksforgeeks.org/priority-scheduling-in-operating-system/
2. https://www.tutorialspoint.com/operating_system/os_priority_scheduling_algorithm.htm
3. https://www.vbspu.ac.in/e-content/Scheduling-Algorithm.pdf

XIII Assessment Scheme (25 Marks)

S. No.	Weightage- Process related (Marks-10)	40%
1.	Logic Formation	20%
2.	Debugging Ability	10%
3.	Follow Ethical Practices:	10%
	Weightage- Product related (Marks-15)	**60%**
4.	Expected Output	25%
5	Timely Submission	25%
6.	Answer to Sample questions	10%
	Total (25 Marks)	**100%**

Marks Obtained			Dated Signature of Teacher
Process related (10)	**Product related (15)**	**Total(25)**	

Practical No. 7: Write C/Python program to calculate average waiting time and Turnaround Time of n processes with Round Robin (RR) CPU scheduling algorithm.

I Practical Significance

The Round Robin (RR) Scheduling Algorithm is a preemptive scheduling algorithm designed to provide fairness and responsiveness in time-sharing systems. Round Robin scheduling is preemptive, which means that a running process can be interrupted by another process and sent to the ready queue even when it has not completed its entire execution in CPU. It is a preemptive version of First Come First Serve (FCFS) scheduling algorithm.

II Industry / Employer Expected Outcome(s)

1. Able to find out waiting time and turnaround time by using Round Robin (RR) CPU Scheduling algorithm.

III Course Level Learning Outcomes(s)

CO3 – Implement various CPU Scheduling algorithms and evaluate their effectiveness.

IV Laboratory Learning Outcome(s)

LLO 7.1 Implement Round Robin (RR) Scheduling algorithm.

V Relevant Affective Domain related Outcomes

1. Follow precautionary measures.
2. Follow naming conventions
3. Follow ethical practices

VI Relevant Theoretical Background

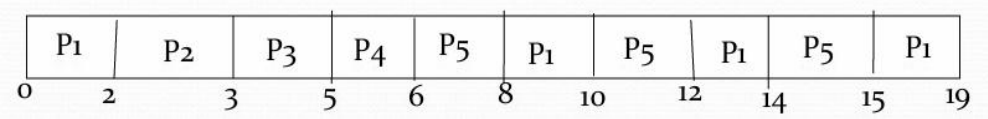
The Round-Robin (RR) scheduling algorithm is designed especially for time sharing systems. It is similar to FCFS scheduling, but preemption is added to switch between processes. A small unit of time, called a time quantum or time slice, is defined. The ready queue is treated as a circular queue. The CPU scheduler goes around the ready queue, allocating the CPU to each process for a time interval of up to 1 time quantum.

To implement RR scheduling, keep the ready queue as a FIFO queue of processes. New processes are added to the tail of the ready queue. The CPU scheduler picks the first process from the ready queue, sets a timer to interrupt after 1 time quantum, and dispatches the process. One of two things will then happen. The process may have a CPU burst of less than 1 time quantum. In this case, the process itself will release the CPU voluntarily. The scheduler will then proceed to the next process in the ready queue. Otherwise, if the CPU burst of the currently running process is longer than 1 time quantum, the timer will go off and will cause an interrupt to the operating system. A context switch will be executed, and the process will be put at the tail of the ready queue. The CPU scheduler will then select the next process in the ready queue.

As an example, consider the following set of processes, assumed to have arrived at time 0, in the order P1, P2,p3,p4 and P5, with the length of the CPU burst given in milliseconds. Time quantum = 2ms.

Process	CPU Burst Time
P1	10
P2	1
P3	2
P4	1
P5	5

Gantt chart is



Completion Time(CT): P1 =19, P2 =3, P3 = 5, P4 =6, P5= 15

Process	CPU burst Time(ms) (BT)	Turnaround Time (CT-AT)	Waiting Time (TAT-BT)
P1	10	19-0=19	19-10=9
P2	1	3-0=3	3-1=2
P3	2	5-0=5	5-2=3
P4	1	6-0=6	6-1=5
P5	5	15-0=15	15-5=10

Average Turnaround Time(ATAT) = (19+3+5+6+15) / 5 = 9.6 ms
Average Waiting Time(AWT) = (9+2+3+5+10) / 5 = 5.8 ms

VII Recourses Required:

- Computer System with basic configuration. Linux/Ubuntu/ CentOS /any other open source Operating System.
- Software: Turbo C/ vi editor/ Any open source Python IDE such IDLE, Jupyter Notebook, Spyder, PyCharm etc.)

VIII Precautions to be followed:

1. Handle computer system with care.
2. Strictly follow the instructions for writing, compiling, and executing the program.
3. Don't forget to save file before execution.
4. Follow safety Practices

IX Program Code:

Implement the Round Robin (RR) CPU Scheduling Algorithm to calculate average waiting time and average turnaround time.

Note: Attach the code at the end.

(This practical can be performed in any of the compiler like Turbo C/ vi editor etc or Python Interpreter / IDE like any open source IDE such IDLE, Jupyter Notebook, Spyder, PyCharm etc.)

X Conclusion

……

……

……

XI Practical related questions

Note: Below given are few sample questions for reference. Teachers must design more such questions to ensure the achievement of identified CO.

1. What is time quantum?
2. State advantages and disadvantages of round robin scheduling.
3. How context switching occurs in Round Robin scheduling and its impact on system performance?
4. Consider the processes P1,P2 ,P3,P4,p5 given in the below table. Find average waiting time and average turnaround time by using Round Robin Scheduling algorithm. Time quantum=3ms

Process	Burst Time	Arrival Time
P1	8	0
P2	7	1
P3	2	5
P4	3	6
P5	5	8

Space for Answer

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

XII References:

1. https://www.geeksforgeeks.org/round-robin-scheduling-in-operating-system/
2. https://www.tutorialspoint.com/operating_system/os_round_robin_scheduling_algorithm.htm
3. https://www.vbspu.ac.in/e-content/Scheduling-Algorithm.pdf

XIII Assessment Scheme (25 Marks)

S. No.	Weightage- Process related (Marks-10)	40%
1.	Logic Formation	20%
2.	Debugging Ability	10%
3.	Follow Ethical Practices:	10%
	Weightage- Product related (Marks-15)	**60%**
4.	Expected Output	25%
5	Timely Submission	25%
6.	Answer to Sample questions	10%
	Total (25 Marks)	**100%**

Marks Obtained			Dated Signature of Teacher
Process related (10)	**Product related (15)**	**Total(25)**	

Practical No. 8: Write a C/Python program to implement Banker's Algorithm.

I Practical Significance

When a new process enters a system, it must declare the maximum number of instances of each resource type it needed. This number may exceed the total number of resources in the system. When the user request a set of resources, the system must determine whether the allocation of each resources will leave the system in safe state. If it will the resources are allocation; otherwise the process must wait until some other process release the resources.

II Industry / Employer Expected Outcome(s)

1. Able to understand resource allocation and deadlock avoidance in operating system.

III Course Level Learning Outcomes(s)

CO3 – Implement various CPU Scheduling algorithms and evaluate their effectiveness.

IV Laboratory Learning Outcome(s)

LLO 8.1 Implement Banker's algorithm for deadlock avoidance.

V Relevant Affective Domain related Outcomes

1. Follow precautionary measures.
2. Follow naming conventions
3. Follow ethical practices

VI Relevant Theoretical Background

Deadlock is a situation where in two or more competing actions are waiting f or the other to finish, and thus neither ever does. When a new process enters a system, it must declare the maximum number of instances of each resource type it needed. This number may exceed the total number of resources in the system. When the user request a set of resources, the system must determine whether the allocation of each resources will leave the system in safe state. If it will the resources are allocation; otherwise the process must wait until some other process release the resources.

Data structures

n- Number of process, m-number of resource types.
Available: Available[j]=k, k – instance of resource type Rj is available.
Max: If max[i, j]=k, Pi may request at most k instances resource Rj.
Allocation: If Allocation [i, j]=k, Pi allocated to k instances of resource Rj Need: If Need[I, j]=k, Pi may need k more instances of resource type Rj, Need[I, j]=Max[I, j]-Allocation[I, j];

Safety Algorithm

1. Work and Finish be the vector of length m and n respectively, Work=Available and Finish[i] =False.
2. Find an i such that both

Finish[i]
=False
Need<=Work If no such I

exists go to step 4.
3. work= work + Allocation, Finish[i] =True;
4. if Finish[1]=True for all I, then the system is in safe state.

Resource request algorithm

Let Request i be request vector for the process Pi, If request i=[j]=k, then process Pi wants k instances of resource type Rj.

1. if Request<=Need I go to step 2. Otherwise raise an error condition.
2. if Request<=Available go to step 3. Otherwise Pi must since the resources are available.
3. Have the system pretend to have allocated the requested resources to process Pi by modifying the state as follows;

Available=Available-Request I;
Allocation I=Allocation +Request I;
Need i=Need i- Request I;

If the resulting resource allocation state is safe, the transaction is completed and process Pi is allocated its resources. However if the state is unsafe, the Pi must wait for Request i and the old resource-allocation state is restored.

ALGORITHM:

1. Start the program.
2. Get the values of resources and processes.
3. Get the avail value.
4. After allocation find the need value.
5. Check whether its possible to allocate.
6. If it is possible then the system is in safe state.
7. Else system is not in safety state.
8. If the new request comes then check that the system is in safety.
9. Or not if we allow the request.
10. Stop the program.
11. End

VII Recourses Required:

- Computer System with basic configuration. Linux/Ubuntu/ CentOS /any other open source Operating System.
- Software: Turbo C/ vi editor/ Any open source Python IDE such IDLE, Jupyter Notebook, Spyder, PyCharm etc.)

VIII Precautions to be followed:

1. Handle computer system with care.
2. Strictly follow the instructions for writing, compiling, and executing the program.
3. Don't forget to save file before execution.
4. Follow safety Practices

IX Program Code: Write a C/Python program to implement Banker's Algorithm for deadlock avoidance.

Note: Attach the code at the end.

(This practical can be performed in any of the compiler like Turbo C/ vi editor etc or Python Interpreter / IDE like any open source IDE such IDLE, Jupyter Notebook, Spyder, PyCharm etc.)

X Conclusion

……

……

……

XI Practical related questions

Note: Below given are few sample questions for reference. Teachers must design more such questions to ensure the achievement of identified CO.

1. What is meant by deadlock
2. What is safe state in banker's algorithms?
3. What is banker's algorithm?
4. What happens if a resource request leads to an unsafe state?
5. What are the necessary conditions where deadlock occurs?

Space for Answer

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

XI References:

1. Banker's Algorithm in Operating System - GeeksforGeeks
2. https://www.tutorialspoint.com/banker-s-algorithm-in-operating-system
3. Bankers Algorithm in OS - Scaler Topics

XII Assessment Scheme (25 Marks)

S. No.	Weightage- Process related (Marks-10)	40%
1.	Logic Formation	20%
2.	Debugging Ability	10%
3.	Follow Ethical Practices:	10%
	Weightage- Product related (Marks-15)	**60%**
4.	Expected Output	25%
5	Timely Submission	25%
6.	Answer to Sample questions	10%
	Total (25 Marks)	**100%**

Marks Obtained			Dated Signature of Teacher
Process related (10)	**Product related (15)**	**Total(25)**	

Practical No. 9: Basic memory management commands - df, free, vmstat, /proc/meminfo, htop

I Practical Significance

Basic memory management commands in an operating system (OS) allow users to monitor and manage how memory is used. These commands provide insights into memory usage, including physical RAM and swap space.

II Industry / Employer Expected Outcome(s)

1. Able to execute basic memory management command df, free, vmstat, /proc/meminfo, htop.

III Course Level Learning Outcomes(s)

CO4 – Analyze the Memory Management techniques used by an Operating System.

IV Laboratory Learning Outcome(s)

LLO 9.1 Execute memory management commands.

V Relevant Affective Domain related Outcomes

1. Follow precautionary measures.
2. Follow naming conventions
3. Follow ethical practices

VI Relevant Theoretical Background

1. df

disk free also known as `df`, which is a powerful utility that provides valuable information on disk space utilization. The df command displays information about file system disk space usage on the mounted file system. This command retrieves the information from `/proc/mounts` or `/etc/mtab`. By default, df command shows disk space in Kilobytes (KB) and uses the SI unit suffixes (e.g, M for megabytes, G for gigabytes) for clarity.

Syntax:

df [options] [filesystems]

```
jayesh@jayesh-VirtualBox:~$ df
Filesystem     1K-blocks     Used Available Use% Mounted on
tmpfs             401716    41836    359880  11% /run
/dev/sda4       11265616 11249232         0 100% /
tmpfs            2008576        0   2008576   0% /dev/shm
tmpfs               5120        4      5116   1% /run/lock
/dev/sda2         524252     6216    518036   2% /boot/efi
tmpfs             401712      176    401536   1% /run/user/1000
jayesh@jayesh-VirtualBox:~$
```

2. Free

free Displays the amount of memory that is currently available and used by the system (both physical and swapped). free command gathers this data by parsing proc/meminfo. By default, the amount of memory is displayed in kilobytes.

Syntax :free

```
root@jayesh-VirtualBox:~# free
               total        used        free      shared  buff/ca
che   available
Mem:         3495708     1267844      113496       10808     2114
368     2036412
Swap:        8007676      497228     7510448
root@jayesh-VirtualBox:~#
```

3. Vmstat

vmstat command is used to display virtual memory statistics of the system. This command reports data about the memory, paging, disk, and CPU activities, etc. The first use of this command returns the data averages since the last reboot. Further users return the data based on sampling periods of long delays.

Syntax : Vmstat

```
root@jayesh-VirtualBox:~# vmstat
procs -----------memory---------- ---swap-- -----io---- -system-- ------cpu-----
 r  b   swpd   free   buff  cache   si   so    bi    bo   in   cs us sy id wa st
 0  0 497228 125916 126740 1988936    0    1    17    30    2    5  0  0 100  0  0
root@jayesh-VirtualBox:~#
```

To view disk statics

Vmstat –d

```
root@jayesh-VirtualBox:~# vmstat -d
disk- ------------reads------------ ------------writes----------- -----IO------
       total merged sectors      ms  total merged sectors      ms    cur    sec
loop0     16      0      50       1      0      0       0       0      0      0
loop1    668      0   18548     147      0      0       0       0      0      0
loop2    152      0    6674      59      0      0       0       0      0      0
loop3   2109      0   49134    3288      0      0       0       0      0      3
loop4   4034      0   51640     326      0      0       0       0      0      1
loop5  28668      0  149856    1696      0      0       0       0      0      3
loop6   1263      0   72292     232      0      0       0       0      0      2
loop7     60      0     816      14      0      0       0       0      0      0
sr0        0      0       0       0      0      0       0       0      0      0
sda   421722  82154 33648266  279598 1051093 866849 59948043 1057558      0
```

Displays the amount of memory used and available.

4./proc/meminfo

/proc/meminfo This file contains all the data about the memory usage. It provides the current memory usage details rather than old, stored values.

Syntax:

cat /proc/meminfo

```
root@jayesh-VirtualBox:~# cat /proc/meminfo
MemTotal:        3495708 kB
MemFree:           88428 kB
MemAvailable:    1998904 kB
Buffers:          126308 kB
Cached:          1784284 kB
SwapCached:        26388 kB
Active:          1593224 kB
Inactive:        1119596 kB
Active(anon):     282616 kB
Inactive(anon):   527932 kB
Active(file):    1310608 kB
```

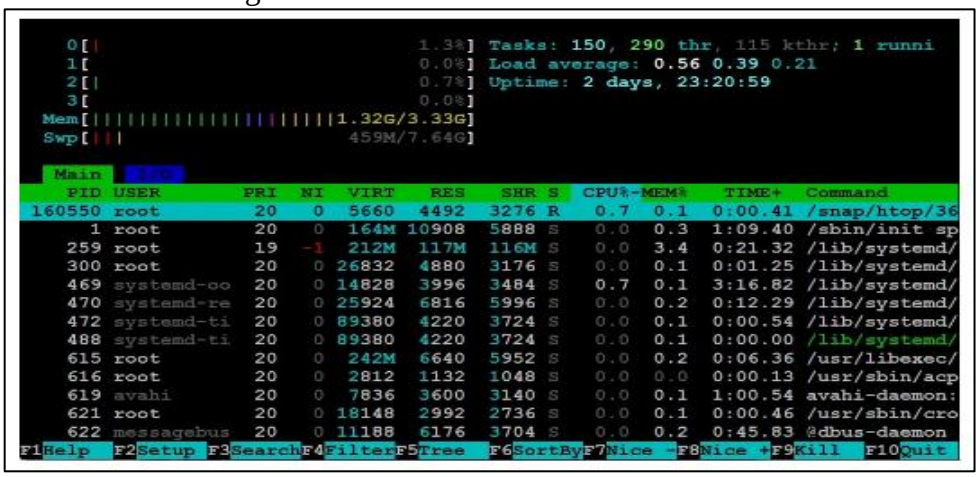
5. Htop

htop is an interactive process viewer. This command is similar to top command except that it allows to scroll vertically and horizontally to allows users to view all processes running on the system, along with their full command line as well as viewing them as a process tree, selecting multiple processes and acting on them all at once.

To use htop we need to install it in our system.

sudo apt-get install htop

Now we can use the following command.



VII Recourses Required:

- Computer System with basic configuration. Linux/Ubuntu/ CentOS /any other open source Operating System**.**

VIII Precautions to be followed:

1. Handle computer system with care.
2. Strictly follow the instructions for writing commands
3. Follow safety Practices

IX Conclusion

……

……

……

X Practical related questions

Note: Below given are few sample questions for reference. Teachers must design more such questions to ensure the achievement of identified CO.

1. What is memory management in an operating system?
2. How can you check the total amount of RAM on a Linux system using a command?
3. What does the vmstat command show, and what are its key fields?
4. Which commands provides the current memory usage details?
5. Write the work of htop command.
6. How do you monitor memory usage trends over time?

Space for Answer

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

XI References:

1. https://www.geeksforgeeks.org/linux-unix/df-command-linux-examples/
2. https://www.geeksforgeeks.org/tracing-memory-usage-linux/

XII Assessment Scheme (25 Marks)

S. No.	Weightage- Process related (Marks-10)	40%
1.	Logic Formation	20%
2.	Debugging Ability	10%
3.	Follow Ethical Practices:	10%
	Weightage- Product related (Marks-15)	**60%**
4.	Expected Output	25%
5	Timely Submission	25%
6.	Answer to Sample questions	10%
	Total (25 Marks)	**100%**

Marks Obtained			Dated Signature of Teacher
Process related (10)	**Product related (15)**	**Total(25)**	

Practical No. 10: Write a C/Python program on First In First Out (FIFO) Page Replacement algorithm.

I Practical Significance

When there is a page fault, the referenced page must be loaded. If there is no available frame in memory, then one page is selected for replacement. If the selected page has been modified, it must be copied back to disk (swapped out). A page replacement algorithm is needed to decide which page needs to be replaced when new page comes in.

II Industry / Employer Expected Outcome(s)

1. Able to write a program to solve page fault problem and page hit.
2. Able to develop a program to implement page replacement algorithm.

III Course Level Learning Outcomes(s)

CO4 – Analyze the Memory Management techniques used by an Operating System

IV Laboratory Learning Outcome(s)

LLO 10.1 Implement First In First Out (FIFO) Page Replacement algorithm .

V Relevant Affective Domain related Outcomes

1. Follow precautionary measures.
2. Follow naming conventions
3. Follow ethical practices

VI Relevant Theoretical Background

This is the simplest page replacement algorithm. In this algorithm, the operating system keeps track of all pages in the memory in a queue, the oldest page is in the front of the queue. When a page needs to be replaced page in the front of the queue is selected for removal.
A page fault is an exception that occurs when a program tries to access a memory page not currently present in physical memory (RAM). Conversely, a page hit occurs when the required page is found in RAM

Example :
Consider page reference string 7,0,1,2,0,3,0,4,2,3,0,3,2,1,2,0,1,7,0,1 with 3-page frames. Find the number of page faults using FIFO Page Replacement Algorithm

F1	7	7	7	2	2	2	2	4	4	4	0	0	0	0	0	0	0	7	7	7
F2		0	0	0	0	3	3	3	2	2	2	2	2	1	1	1	1	1	0	0
F3			1	1	1	1	0	0	0	3	3	3	3	3	2	2	2	2	2	1
	F	F	F	F	H	F	F	F	F	F	F	H	H	F	F	H	H	F	F	F

Total page Faults = 15 **Total page hit = 5**

Initially all slots are empty, so when 7, 0, 1 came they are allocated to the empty slots → 3 Page Faults.
Then 2 comes, it is not available in memory so it replaces the oldest page slot i.e 7 →1 Page Fault.

When 0 comes, it is already in memory so → 1 Page hits.
Belady's anomaly proves that it is possible to have more page faults when increasing the number of page frames while using the First in First Out (FIFO) page replacement algorithm.

VII Recourses Required:

- Computer System with basic configuration. Linux/Ubuntu/ CentOS /any other open source Operating System.
- Software: Turbo C/ vi editor/ Any open source Python IDE such IDLE, Jupyter Notebook, Spyder, PyCharm etc.)

VIII Precautions to be followed:

1. Handle computer system with care.
2. Strictly follow the instructions for writing commands
3. Follow safety Practices

IX Program Code :

Consider the string: 0, 1, 2, 3, 0, 1, 2, 3, 0, 1, 2, 3, 4, 5, 6, 7 with frame size 3, calculate page fault in both the cases using FIFO algorithm.

Note: Attached the code at the end.
This practical can be performed in any of the compiler like Turbo C/ vi editor etc or Python Interpreter / IDE like any open source IDE such IDLE, Jupyter Notebook, Spyder, PyCharm etc.)

X Conclusion

…….

…….

…….

X Practical related questions

Note: Below given are few sample questions for reference. Teachers must design more such questions to ensure the achievement of identified CO.

1. What is page hit ?
2. What is page fault?
3. What is FIFO page replacement? How does it work?
4. State the advantages and disadvantages of FIFO.
5. Count number of page hits in above mention example given in section IX.
6. Given the following page reference string:1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5
   How many page faults will occur using FIFO with 3 frames?

Space for Answer

…….

…….

…….

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

XI References:

1. Page Replacement Algorithms in Operating Systems - GeeksforGeeks
2. Program for Page Replacement Algorithms | Set 2 (FIFO) - GeeksforGeeks
3. FIFO Page Replacement Algorithm - Scaler Topics

XII Assessment Scheme (25 Marks)

S. No.	Weightage- Process related (Marks-10)	40%
1.	Logic Formation	20%
2.	Debugging Ability	10%
3.	Follow Ethical Practices:	10%
	Weightage- Product related (Marks-15)	**60%**
4.	Expected Output	25%
5	Timely Submission	25%
6.	Answer to Sample questions	10%
	Total (25 Marks)	**100%**

Marks Obtained			Dated Signature of Teacher
Process related (10)	**Product related (15)**	**Total(25)**	

Practical No. 11: Write a C/Python program on Least Recently Used (LRU) Page Replacement algorithm.

I Practical Significance

Page replacement algorithm are needed to decide which page needed to be replaced when new page comes in. Whenever a new page is referred and not present in memory, page fault occurs and Operating System replaces one of the existing pages with newly needed page. In Least Recently Used (LRU) algorithm is a Greedy algorithm where the page to be replaced is least recently used.

II Industry / Employer Expected Outcome(s)

1. Able to write a program to solve page fault problem and page hit.
2. Able to develop a program to implement LRU page replacement algorithm.

III Course Level Learning Outcomes(s)

CO4 – Analyze the Memory Management techniques used by an Operating System.

IV Laboratory Learning Outcome(s)

LLO 11.1 Implement Least Recently Used (LRU) Page Replacement algorithm.

V Relevant Affective Domain related Outcomes

1. Follow precautionary measures.
2. Follow naming conventions
3. Follow ethical practices

VI Relevant Theoretical Background

The Least Recently used (LRU) algorithm replaces the page that has not been used for the longest period of time. It is based on the observation that pages that have not been used for long time will probably remain unused for the longest time and are to be replaced.

Example :
Consider page reference string 7,0,1,2,0,3,0,4,2,3,0,3,2,1,2,0,1,7,0,1 with 3-page frames. Find the number of page faults using LRU Page Replacement Algorithm

F1	7	7	7	2	2	2	2	4	4	4	0	0	0	1	1	1	1	1	1	1
F2		0	0	0	0	0	0	0	0	3	3	3	3	3	3	0	0	0	0	0
F3			1	1	1	3	3	3	2	2	2	2	2	2	2	2	2	7	7	7
	F	F	F	F	H	F	H	F	F	F	F	H	H	F	H	F	H	F	H	H

Total page Faults = 12 **Total page hit = 8**

Initially all slots are empty, so when 7 0 1 2 are allocated to the empty slots → 4 Page faults.
0 is already there so → 0 Page fault(Hit).
when 3 came it will take the place of 7 because it is least recently used→1 Page fault
0 is already in memory so → 0 Page fault.
4 will takes place of 1 →1 Page Fault.
Now for the further page reference string → 0 Page fault because they are already available in the memory and so on.

VII Recourses Required:

- Computer System with basic configuration. Linux/Ubuntu/ CentOS /any other open source Operating System.
- Software: Turbo C/ vi editor/ Any open source Python IDE such IDLE, Jupyter Notebook, Spyder, PyCharm etc.)

VIII Precautions to be followed:

1. Handle computer system with care.
2. Strictly follow the instructions for writing, compiling, and executing the program.
3. Don't forget to save file before execution.
4. Follow safety Practices

IX Program Code : Consider the string: 0, 1, 2, 3, 0, 1, 2, 3, 0, 1, 2, 3, 4, 5, 6, 7 with frame size 3, calculate page fault using LRU algorithm.

Note: Attached the code at the end.
This practical can be performed in any of the compiler like Turbo C/ vi editor etc or Python Interpreter / IDE like any open source IDE such IDLE, Jupyter Notebook, Spyder, PyCharm etc.)

X Conclusion

…….

…….

……

XI Practical related questions

Note: Below given are few sample questions for reference. Teachers must design more such questions to ensure the achievement of identified CO.

1. Compare FIFO and LRU.
2. State the advantages and disadvantages of LRU.
3. Count number of page hits and page faults in above mention example given in section IX.
4. You are given the following page reference string: 1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5 Using 3 page frames, how many page faults occur with LRU?

Space for Answer

…….

…….

……………………………………………………………………………………………………
……………………………………………………………………………………………………
……………………………………………………………………………………………………
……………………………………………………………………………………………………
……………………………………………………………………………………………………
……………………………………………………………………………………………………
……………………………………………………………………………………………………
……………………………………………………………………………………………………
……………………………………………………………………………………………………
……………………………………………………………………………………………………
……………………………………………………………………………………………………
……………………………………………………………………………………………………
……………………………………………………………………………………………………
……………………………………………………………………………………………………
……………………………………………………………………………………………………
……………………………………………………………………………………………………
……………………………………………………………………………………………………
……………………………………………………………………………………………………
……………………………………………………………………………………………………
…………………………………………………………………………………………………
…………………………………………………………………………………………………
……………………………………………………………………………………………………
……………………………………………………………………………………………………
……………………………………………………………………………………………………
……………………………………………………………………………………………………
……………………………………………………………………………………………………
……………………………………………………………………………………………………
……………………………………………………………………………………………………
……………………………………………………………………………………………………
…………………………………………………………………………………………………..
……………………………………………………………………………………………………
……………………………………………………………………………………………….

……

……

……

……

……

……

……

……

……

……

……

XII References:

1. Page Replacement Algorithms in Operating Systems - GeeksforGeeks
2. Program for Least Recently Used (LRU) Page Replacement algorithm - GeeksforGeeks
3. LRU Page Replacement Algorithm - Scaler Topics

XIII Assessment Scheme (25 Marks)

S. No.	Weightage- Process related (Marks-10)	40%
1.	Logic Formation	20%
2.	Debugging Ability	10%
3.	Follow Ethical Practices:	10%
	Weightage- Product related (Marks-15)	**60%**
4.	Expected Output	25%
5	Timely Submission	25%
6.	Answer to Sample questions	10%
	Total (25 Marks)	**100%**

Marks Obtained			Dated Signature of Teacher
Process related (10)	**Product related (15)**	**Total(25)**	

Practical No. 12: Write a C/Python program on sequential file allocation method.

I Practical Significance

The most common form of file structure is the sequential file in this type of file, a fixed format is used for records. All records (of the system) have the same length, consisting of the same number of fixed length fields in a particular order because the length and position of each field are known, only the values of fields need to be stored, the field name and length for each field are attributes of the file structure.

II Industry / Employer Expected Outcome(s)

1. Able to develop a program on Sequential file allocation method.
2. Able to understand how files are stored and accessed on a disk, including how to implement sequential access methods.

III Course Level Learning Outcomes(s)

CO5 – Apply techniques for effective File Management in an Operating System.

IV Laboratory Learning Outcome(s)

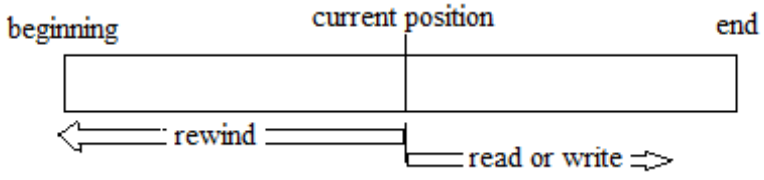
LLO 12.1 Implement sequential file allocation method.

V Relevant Affective Domain related Outcomes

1. Follow precautionary measures.
2. Follow naming conventions
3. Follow ethical practices

VI Relevant Theoretical Background

The simplest access method is sequential access. Information in the file is processed in order, one record after the other. This mode of access is the most common; for example, editors and compilers usually access files in this fashion.
Reads and writes make up the bulk of the operations on a file. A read operation — read next — reads the next portion of the file and automatically advances a file pointer, which tracks the I/O location. Similarly, the write operation — write next — appends to the end of the file and advances to the end of the newly written material (the new end of file). Such a file can be reset to the beginning.



Sequential access, which is depicted in Figure, is based on a tape model of a file and works as well on sequential-access devices as it does on random-access ones.

Algorithm :
Step 1: Start the program.
Step 2: Get the number of files.

Step 3: Get the memory requirement of each file.
Step 4: Allocate the required locations to each in sequential order a).
Randomly select a location from available location s1= random(100);
a) Check whether the required locations are free from the selected location.

```
if(b[s1].flag==0){
for
(j=s1;j<s1+p[i];j++){
if((b[j].flag)==0)count++;
}
if(count==p[i]) break;
}
```

b) Allocate and set flag=1 to the allocated locations.

```
for(s=s1;s<(s1+p[i]);s++)
{
k[i][j]=s; j=j+1; b[s].bno=s;
b[s].flag=1;
}
```

Step 5: Print the results file no, length, Blocks allocated. Step
Step 6: Stop the program

VII Recourses Required:

- Computer System with basic configuration. Linux/Ubuntu/ CentOS /any other open source Operating System.
- Software: Turbo C/ vi editor/ Any open source Python IDE such IDLE, Jupyter Notebook, Spyder, PyCharm etc.)

VIII Precautions to be followed:

1. Handle computer system with care.
2. Strictly follow the instructions for writing, compiling, and executing the program.
3. Don't forget to save file before execution.
4. Follow safety Practices

IX Program Code : Write a program on sequential file allocation method..
Note:
Attached the code at the end.

This practical can be performed in any of the compiler like Turbo C/ vi editor etc or Python Interpreter / IDE like any open source IDE such IDLE, Jupyter Notebook, Spyder, PyCharm etc.)

X Conclusion

……

……

……

XI Practical related questions

Note: Below given are few sample questions for reference. Teachers must design more such questions to ensure the achievement of identified CO.

1. What are the various file allocation strategies?
2. What is Sequential file Allocation Method?
3. What are the disadvantages of sequential allocation methods?
4. Compare sequential allocation with linked and indexed file allocation methods.
5. Compare sequential allocation with linked and indexed file allocation methods.

Space for Answer

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

XII References:

1. https://www.geeksforgeeks.org/file-allocation-methods/
2. https://www.tutorialspoint.com/file-allocation-methods

XII Assessment Scheme (25 Marks)

S. No.	Weightage- Process related (Marks-10)	40%
1.	Logic Formation	20%
2.	Debugging Ability	10%
3.	Follow Ethical Practices:	10%
	Weightage- Product related (Marks-15)	**60%**
4.	Expected Output	25%
5	Timely Submission	25%
6.	Answer to Sample questions	10%
	Total (25 Marks)	**100%**

Marks Obtained			Dated Signature of Teacher
Process related (10)	**Product related (15)**	**Total(25)**	