Format No. QSP/7.1/01.F01 (B) **Issue No.05 Rev. No 5 Dated: Jan 1, 2017**

UNIVERSITY OF PETROLEUM & ENERGY STUDIES

College of Engineering Studies

Dehradun

COURSE PLAN

Programme : B. Tech. (CSE+ AIML)

Course : Operating System LAB

Subject Code : CSEG2107

No. of credits : 1

Semester : III

Session : July 20- Dec 20

Batch : 2019-2021

Prepared by : Ms. Juhi Agrawal

Email : jagrawal@ddn.upes.ac.in

Approved By

Faculty

HOD

UPES Campus

"Energy Acres"

P.O. Bidholi, Via Prem Nagar

Tel : +91-135-2770137

Fax : +91 135- 27760904

Website : www.upes.ac.in

UPES

COURSE PLAN

A. PREREQUISITE:

a. Basic Knowledge Mathematics.

b. Basic Knowledge of Data structure

c. Basic Knowledge of Algorithms

B. PROGRAM OUTCOMES (POs) for DCN:

Program Outcomes for B. Tech. CSE After completion of the program the students will be able to:

Engineering Graduates will be able to:

PO1. **Engineering knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.

PO2. **Problem analysis:** Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.

PO3. **Design/development of solutions:** Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.

PO4. **Conduct investigations of complex problems:** Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

PO5. **Modern tool usage:** Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.

PO6. **The engineer and society:** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.

PO7. **Environment and sustainability:** Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

PO8. **Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

PO9. **Individual and team work:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

PO10. **Communication:** Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

PO11. **Project management and finance:** Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

PO12. **Life-long learning:** Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

PROGRAM SPECIFIC OUTCOMES

PSO1. Perform system and application programming using computer system concepts, concepts of Data Structures, algorithm development, problem solving and optimizing techniques,

PSO2. Apply software development and project management methodologies using concepts of front-end and back-end development and emerging technologies and platforms.

----------to be made for each vertical----------

C. **COURSE OUTCOMES FOR Operating Systems: At the end of this course student should be able to**

CO1: Understand basic concepts of Operating System and its Functions
CO2. Understand the process management and CPU scheduling algorithm.
CO3. Design solution for process synchronization, inter process communication and deadlock handling

CO4.Identify how operating system allocates and manages memory along with concepts and needs of virtual memory.
CO5: Evaluate File Systems, Directory Structures and Access Control Mechanism.
CO6: Measure Performance of Disk Scheduling Technique.

.

Table: Correlation of POs v/s COs

PO/CO	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2	PSO3
CO1	2	1	1		1								3	3	
CO2	2	1	2		1	2			1			1	3	3	
CO3	1	2	2		2	2			2			1	3	3	
CO4	2	2	2		3	1			1			1	3	3	
CO5	2		1		1							1	3	3	
CO6	2	1	2		1								3	3	

1. WEAK 2. MODERATE 3. STRONG

PSOs will be based on your different verticals

S.NO.	Lab Exercise	Contents
1.	Lab. Exercise 1	System calls & I/O System calls
2.	Lab. Exercise 2	Scheduling
3.	Lab. Exercise 3	Inter - process communications
4.	Lab. Exercise 4	Semaphore
5.	Lab. Exercise 5	Viva- Voice
6.	Lab. Exercise 6	Memory Management-I
7.	Lab. Exercise 7	Memory management II
8.	Lab. Exercise 8	File manipulation
9.	Lab. Exercise 9	Fork execution
10.	Lab. Exercise 10	Viva Voice

Prerequisite (Detailed):

UNIX Basic:

UNIX is an operating system which was first developed in the 1960s, and has been under constant development ever since. By operating system, we mean the suite of programs, which make the computer work. It is a stable, multi-user, multi-tasking system for servers, desktops and laptops. UNIX systems also have a graphical user interface (GUI) similar to Microsoft Windows, which provides an easy to use environment. There are many different versions of UNIX, although they share common similarities. The most popular varieties of UNIX are Sun Solaris, GNU/Linux, and MacOS X. The UNIX operating system is madeup of three parts; the kernel, the shell and the programs.

The Kernel

The kernel is the core of the UNIX operating system. Basically,the kernel is a large program that is loaded into memory when the machine is turned on, and it controls the allocation of hardware resources from that point forward. The kernel knows what hardware resources are available (like the processor(s), the on-board memory, the disk drives, network interfaces, etc.), and it has the necessary programs to talk to all the devices connected to it. As an illustration of the way that the shell and the kernel work together, suppose a user types rm myfile (which has the effect of removing the file myfile). The shell searches the file store for the file containing the program rm, and then requests the kernel, through system calls, to execute the program rm on myfile. When the process rm myfile has finished running, the shell then returns the UNIX prompt to the user, indicating that it is waiting for further commands.

The Shell

The shell acts as an interface between the user and the kernel. The shell is a command line interpreter (CLI). It interprets the commands the user types in and arranges for them to be carried out. The commands are themselves programs: when they terminate, the shell gives the

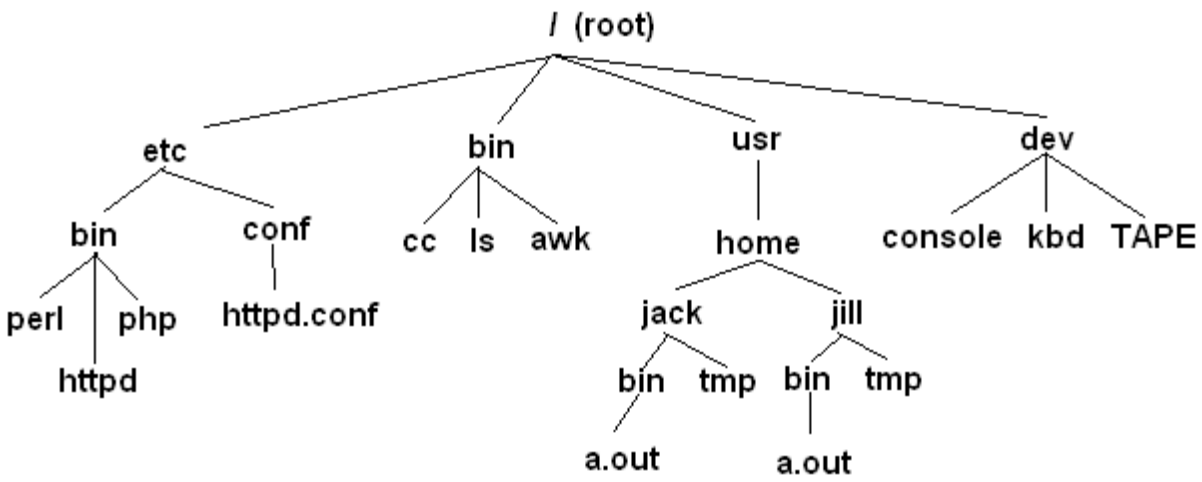
User another prompt .The adept user can customize his /her own shell and users can use different shells on the same machine. UNIX system offers verity of shells like

1) Bourne shell 2) c shell 3) Korn shell 4) Bash shell (very powerful & recommended for use, Linux default shell) History - The shell keeps a list of the commands you have typed in. If you need to repeat a command, use the cursor keys to scroll up and down the list or type history for a list of previous commands.

The UNIX file system

All the stored information on a UNIX computer is kept in a file system. Any time we interact with the UNIX shell, the shell considers us to be located somewhere within a file system. The place in the file system tree where we are located is called the current working directory.

The UNIX file system is hierarchical (resembling a tree structure). The tree is anchored at a place called the root, designated by a slash "/". Every item in the UNIX file system tree is either a file, or a directory. A directory is like a file folder. A directory can contain files, and other directories. A directory contained within another is called the child of the other. A directory in the file system tree may have many children, but it can only have one parent. A file can hold information, but cannot contain other files, or directories.



UNIX/Linux directory structure.

To describe a specific location in the file system hierarchy, you must specify a "path." The path to a location can be defined as an absolute path from the root anchor point, or as a relative path, starting from the current location. When specifying a path, you simply trace a route through the file system tree, listing the sequence of directories you pass through as you go from one point to another. Each directory listed in the sequence is separated by a slash. UNIX provides the shorthand notation of "." to refer to the current location, and ".." to refer to the parent directory.

Linux

Linux is not UNIX, as UNIX is a copyrighted piece of software that demands license fees when any part of its source code is used. Linux was written from scratch to avoid license fees entirely, although the operation of the Linux operating system is based entirely on UNIX. It shares UNIX's command set and look-and-feel, so if you know either UNIX or Linux, you know the other, too. Linux is a freely distributable version of UNIX developed primarily by Linus Torvalds at the University of Helsinki in Finland.

Text editors in Linux

Linux is just as well suited for word processing as any other operating system. There are several excellent word processing programs for Linux like AbiWord, KWord, part of the KOffice suite and the OpenOffice.org as well as StarOfficesuite's word processor.

Why use a text editor?

A text editor is just like a word processor without a lot of features. All operating systems come with a basic text editor. Linux comes with several. The main use of a text editor is for writing something in plain text with no formatting so that another program can read it. Based on the information it gets from that file, the program will run one way or another.

(Unix Command)

Aim: To study and execute the commands in UNIX.

(a) SIMPLE COMMANDS :

- Date Command.
- Calendar Command.
- Echo Command.
- Banner Command.
- "Who" Command
- "Who am I" Command.
- "tty" Command.
- "Binary" Calculator Command.
- "CLEAR" Command.
- MAN Command.
- MANIPULATION Command.
- LIST Command.

(b) DIRECTORY RELATED COMMANDS:

- Present Working Directory Command.
- MKDIR Command.
- CD Command.
- RMDIR Command.

(c) FILE RELATED COMMANDS:

- CREATE A FILE.
- DISPLAY A FILE.
- COPYING CONTENTS.
- SORTING A FILE.
- COPYING CONTENTS FROM ONE FILE TO ANOTHER.
- MOVE Command.

- REMOVE Command.
- WORD Command.
- LINE PRINTER.
- PAGE Command.
- FILTERS AND PIPES.

(d) COMMUNICATION THROUGH UNIX COMMANDS:

- MESG.
- WRITE.
- WALL.
- MAIL.
- REPLY.

Successfully executed all the UNIX commands.

(EDITOR COMMANDS)

Aim: To study the various commands operated in vi editor in UNIX.

DESCRIPTION:

The Vi editor is a visual editor used to create and edit text, files, documents and programs. It displays the content of files on the screen and allows a user to add, delete or change part of text. There are three modes available in the Vi editor, they are

1. Command mode
2. Input (or) insert mode.

Starting Vi :

The Vi editor is invoked by giving the following commands in UNIX prompt.

Syntax: $vi <filename> (or)

$vi

This command would open a display screen with 25 lines and with tilt (~) symbol at the start of each line. The first syntax would save the file in the filename mentioned and for the next the filename must be mentioned at the end.

Options:

1. vi +n <filename> - This would point at the nth line (cursor pos).
2. vi –n <filename> - This command is to make the file to read only to change from one mode to another press escape key.

Commands:

(a) INSERTING AND REPLACING COMMANDS.
(b) CURSOR MOVEMENT COMMANDS IN vi.
(c) DELETING THE TEXT FROM Vi.
(d) SAVING AND QUITING FROM vi.: $Esc:wq
(e) Compilation using gcc: $gcc filename.c
(f) For Output: $.\a.out Successfully executed all the vi EDITOR commands

UPES

Experiment:- 1

(System calls & I/O system calls)

1(a) System Calls

Aim: To write programs to perform following operations in UNIX:

a) Process Creation

b) Executing a command

c) Sleep command

d) Sleep command using get pid

e) Signal handling using kill

f) Wait command

Theory:

What do you mean by all the terms mention above? Explain with proper example.

Procedure:

1. Write the Algorithms.
2. Write the programs.
3. Run the programs on the terminal.

Result:

Thus, all the programs were executed and verified successfully.

1(b) (I/O System calls)

Aim: To write programs to perform following operations in UNIX:

a) Reading from a file

b) Writing into a file

c) File Creation

d) Implementation of ls command

e) Implementation of grep command.

Theory:

What do you mean by all the terms mention above? Explain with proper example.

Procedure:

1. Write the Algorithms.
2. Write the programs.
3. Run the programs on the terminal.

Result:

Thus, all the programs were executed and verified successfully.

Experiment 2

(Scheduling)

2(a) (FIRST COME FIRST SERVE)

Aim: To write a C program to implement the CPU scheduling algorithm for FIRST COME FIRST SERVE.

PROBLEM DESCRIPTION:

CPU scheduler will decide which process should be given the CPU for its execution. For this, it uses different algorithm to choose among the process. One among that algorithm is FCFS algorithm. In this algorithm, the process, which arrives first, is given the CPU after finishing its request only it will allow CPU to execute other process.

Procedure:

1. Write the Algorithm.
2. Write the program.
3. Run the programs on the terminal.

Result:

Thus the C program to implement CPU scheduling algorithm for first come first serve was executed successfully.

2(b) (SHORTEST JOB FIRST)

Aim: To write a C program to implement the CPU scheduling algorithm for shortest job first.

PROBLEM DESCRIPTION:

CPU scheduler will decide which process should be given the CPU for its execution. For this it use different algorithm to choose among the process. One among that algorithm is SJF algorithm.

In this algorithm the process which has less service time given the CPU after finishing its request only it will allow CPU to execute next other process.

Procedure:

1. Write the Algorithm.
2. Write the program.
3. Run the programs on the terminal.

Result:

Thus the C program to implement the CPU scheduling algorithm for shortest job first was executed successfully.

2(c) (ROUND ROBIN)

Aim: To write a C program to simulate the CPU scheduling algorithm for round robin.

PROBLEM DESCRIPTION:

CPU scheduler will decide which process should be given the CPU for its execution .For this it use different algorithm to choose among the process .one among that algorithm is Round robin algorithm.

In this algorithm we are assigning some time slice .The process is allocated according to the time slice ,if the process service time is less than the time slice then process itself will release the CPU voluntarily .The scheduler will then proceed to the next process in the ready queue. If the CPU burst of the currently running process is longer than 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.

Procedure:

1. Write the Algorithm.
2. Write the program.
3. Run the programs on the terminal.

Result:

Thus the C program to simulate CPU scheduling algorithm for round robin was executed successfully.

2(d) (priority scheduling.)

Aim: To write a C program to implement CPU scheduling algorithm for priority scheduling.

PROBLEM DESCRIPTION:

CPU scheduler will decide which process should be given the CPU for its execution. For this it use different algorithm to choose among the process. One among that algorithm is FCFS algorithm.

In this algorithm the process which arrives first is given the CPU after finishing its request only it will allow CPU to execute other process.

Procedure:

1. Write the Algorithm.
2. Write the program.
3. Run the programs on the terminal.

Result:

Thus C program to implement CPU scheduling algorithm for priority scheduling was executed successfully.

Experiment 3

(Inter process communication)

Aim: Write a program that creates a child process. Parent process writes data to pipe and child process reads the data from pipe and prints it on the screen.

Theory:

One of the mechanisms that allow related-processes to communicate is the pipe. A pipe is a one-way mechanism that allows two related processes (i.e. one is an ancestor of the other) to send a byte stream from one of them to the other one. The system assures us of one thing: The order in which data is written to the pipe, is the same order as that in which data is read from the pipe. The system also assures that data won't get lost in the middle, unless one of the processes (the sender or the receiver) exits prematurely.

Procedure:

1. Write the Algorithm.
2. Write the program.
3. Run the programs on the terminal.

Sample O/P:

Result:

Thus the Pipe Processing program was executed and verified successfully.

Experiment 4

(Semaphore)

4(a)

Aim: Write a program that demonstrates how two processes can share a variable using semaphore.

Theory:

Semaphore: Semaphore is a protected variable that can be accessed and manipulated by means of only two operations.

1) Wait() :

If (s > 0) then decrements the value of s. otherwise the process is blocked on s.

2) Signal()

Increments the value of semaphore. If semaphore queue is not empty , it moves one of the blocked processes to the ready queue.

- Binary Semaphore : can take on only two values 0 and 1
- Counting Semaphore : Can take any non-negative integer value.

Procedure:

1. Write the Algorithm.
2. Write the program.
3. Run the programs on the terminal.

Result:

Concept and use of semaphores have been studied & implemented

4(b) (Producer & consumer)

Aim: To write a C program to implement the Producer & consumer Problem (Semaphore)

Theory:

The producer-consumer problem (Also called the bounded-buffer problem.) illustrates the need for synchronization in systems where many processes share a resource. In the problem, two processes share a fixed-size buffer. One process(producer) produces information and puts it in

the buffer, while the other process (consumer) consumes information from the buffer. These processes do not take turns accessing the buffer, they both work concurrently. Herein lies the problem. What happens if the producer tries to put an item into a full buffer? What happens if the consumer tries to take an item from an empty buffer?

In order to synchronize these processes, we will block the producer when the buffer is full, and we will block the consumer when the buffer is empty.

Procedure:

1. Write the Algorithm.
2. Write the program.
3. Run the programs on the terminal.

Result:

Thus the program for solving producer and consumer problem using semaphore was executed successfully.

Experiment 5

(Viva Voice)

Plan a viva-voice for the Class and discuss the status of the knowledge level of the students with them

Experiment 6

(Memory management-1)

Aim: To write a C program to implement memory management using paging technique.

Theory:

What do you mean by paging technique?

Procedure:

1. Write the Algorithm.
2. Write the program.
3. Run the programs on the terminal.

Result:

Thus the program for implementing the paging concept was executed and the output was verified successfully.

Experiment 7

(Memory management-II)

Aim: To write a C program to implement memory management using segmentation

Theory:

What do you mean by segmentation?

Procedure:

1. Write the Algorithm.
2. Write the program.
3. Run the programs on the terminal.

Result:

Thus the program for implementing the segmentation concept was executed and the output was verified successfully.

Experiment 8

(FILE MANIPULATION)

8(a) (Displays the file and Directory)

Aim: To write a program for file manipulation for displays the file and directory in UNIX.FACILITIES

Theory:

What do you mean by file manipulation?

Procedure:

1. Write the Algorithm.
2. Write the program.
3. Run the programs on the terminal.

Result:

Thus the File manipulation program was executed and verified successfully.

8(b) (Creating new Directory)

Aim: To write a program to perform file manipulation for creating a new directory.

Theory:

What do you mean by file manipulation for creating a new directory?

Procedure:

1. Write the Algorithm.
2. Write the program.
3. Run the programs on the terminal.

Result:

Thus the File manipulation program was executed and verified successfully.

Experiment 9

(Fork Execution)

9 (a) (Simple fork execution)

Aim: To write a program to show the FORK Execution.

Theory:

What do you mean by fork?

Procedure:

1. Write the Algorithm.
2. Write the program.
3. Run the programs on the terminal.

Result:

Thus the Fork Execution program was executed and verified successfully.

9(b) (fork system call)

Aim: To write a program for implement the Fork System calls in UNIX.

Theory:

What do you mean by fork system call?

Procedure:

1. Write the Algorithm.
2. Write the program.
3. Run the programs on the terminal.

Result:

Thus the Fork Execution program was executed and verified successfully.

UPES

Experiment 10

(Banker's algorithm)

10(a) Aim: To implement Banker's algorithm for a multiple resources.

Theory:

This uses a deadlock avoidance policy. Banker's algorithm is applied to arbitrary number of processes and arbitrary number of resource classes each with multiple instance.

The banker's algorithm mainly consists of the following matrices:

i). The resources assigned matrix which shows the number of resources of each type are currently assigned to each process.

ii). Resources still needed matrix indicates how many resources each process need in order to complete. For this the process must state total resources needed before executing the program.

It also consists of three vectors.

P vector => Processed resources | A vector => Available resources |

E vector => Existing resources

Algorithm to check if a state is safe or not:

I). Look for a row R whose count unmet resources needs are all smaller than A. if no such row Exists, the system is deadlocked since no process can run to completion.

II). Assume the process of the row chosen requests all the resources it needs & finishes. Mark the process as terminated & add its resources to A vector.

III). Repeat steps I & II until either all processes are marked terminated.

If several processes are eligible to be choosen in step1, it does not matter which one is selected.

Procedure:

1. Write the Algorithm.
2. Write the program.
3. Run the programs on the terminal.

Result:

Banker's algorithm to avoid deadlock is implemented successfully.

10(b) (Dinning Philosopher's problem)

Aim: To implement dinning philosopher’s problem.

Theory:

Five philosophers are seated around a circular table. Each philosopher has a place of spaghetti and he needs two forks to eat. Between each plate there is a fork. The life of a philosopher consists of alternate period of eating & thinking. When a philosopher gets hungry, he tries to acquire his left fork, if he gets it, it tries to acquire right fork.

In this solution, we check after picking the left fork whether the right fork is available or not. If not, then philosopher puts down the left fork & continues to think. Even this can fail if all the philosophers pick the left fork simultaneously & no right forks available & putting the left fork down again. This repeats & leads to starvation.

Now, we can modify the problem by making the philosopher wait for a random amount of time instead of same time after failing to acquire right hand fork. This will reduce the problem of starvation.

Procedure:

1. Write the Algorithm.
2. Write the program.
3. Run the programs on the terminal.

Result:

Dinning philosopher’s problem was implemented successfully