(Computer Science Courses for Undergraduate Programme of study with Computer Science discipline as one of the three Core Disciplines)

DISCIPLINE SPECIFIC CORE COURSE (DSC04): Operating Systems

CREDIT DISTRIBUTION, ELIGIBILITY AND PRE-REQUISITES OF THE COURSE

Course title & Code	Credits	Credit distribution of the course			Eligibility criteria	Pre-requisite of the course (if any)
		Lecture	Tutorial	Practical/ Practice		
DSC04: Operating Systems	**4**	**3**	**0**	**1**	Pass in Class XII	DSC 01 Programming using Python/ A course in C/C++/Python at plus 2 level.

Learning Objectives

This course introduces the students to Operating Systems and its importance in computer systems. The focus is to explain the common services provided by an operating system like process management, memory (primary, secondary & virtual) management, I/O management, file management. The course talks about the various functional components of the operating and their design.

Learning outcomes

On successful completion of the course, students will be able to

- gain knowledge of different concepts of the operating System and its components.
- learn about shell scripts and would be able to use the system in an efficient manner.

SYLLABUS OF DSC04

Unit 1 (4 hours)

Introduction: Operating Systems (OS) definition and its purpose, Multiprogrammed and Time Sharing Systems, OS Structure, OS Operations: Dual and Multi-mode, OS as resource manager.

Unit 2 (10 hours)

Operating System Structures: OS Services, System Calls: Process Control, File Management, Device Management, and Information Maintenance, Inter-process Communication, and Protection, System programs, OS structure- Simple, Layered, Microkernel, and Modular.

Unit 3 (9 hours)

Process Management: Process Concept, States. Process Control Block, Context Switch, Process scheduling, Schedulers, Overview of threads and Scheduling Algorithms: First Come First Served, Shortest-Job-First, Priority & Round-Robin.

Unit 4 (9 hours)

Memory Management: Physical and Logical address space, Swapping Contiguous memory allocation strategies - fixed and variable partitions, Segmentation, Paging, virtual memory: Demand Paging.

Unit 5 (8 hours)

File and Input / Output Device Management: File Concepts, File Attributes, File Access Methods, Directory Structure: Single-Level, Two-Level, Tree-Structured, and Acyclic-Graph Directories, Magnetic Disks, Solid-State Disks, Magnetic Tapes.

Unit 6 (5 hours)

Shell Scripting: Shell variables, parameter passing conditional statements, iterative statements, writing and executing shell scripts, utility programs (cut, paste, grep, echo, pipe, filter etc.)

Essential/recommended readings

1. Galvin, S. P. B., Gagne, G., *Operating System Concepts*, 9th edition, John Wiley Publications, 2016.
2. G. Nutt, Operating Systems, Pearson, 2009
3. Das, S., *Unix: Concepts and Applications*, 4th edition, TMH, 2009.

Additional References

1. Dhamdhere, D. M., *Operating Systems: A Concept-based Approach,* 2nd edition, Tata McGraw-Hill Education, 2017.
2. Kernighan, B. W., Pike, R., *The Unix Programming Environment*, Englewood Cliffs, NJ: Prentice-Hall, 1984.
3. Stallings, W., *Operating Systems: Internals and Design Principles*, 9th edition, Pearson Education, 2018.
4. Tanenbaum, A. S., *Modern Operating Systems.* 3rd edition, Pearson Education, 2007.

Suggested Practical List (If any): (30 Hours)

Practical exercises such as

1. Execute various LINUX commands for:
   i. Information Maintenance: wc, clear, cal, who, date, pwd
   ii. File Management: cat, cp, rm, mv, cmp, comm, diff, find, grep
   iii. Directory Management : cd, mkdir, rmdir, ls

2. Execute various LINUX commands for:
   i. Process Control: fork, getpid, ps
   ii. Communication: Input-output redirection, Pipe
   iii. Protection Management: chmod, chown, chgrp
3. Write a program(using fork() and/or exec() commands) where parent and child execute:
   i. same program, same code.
   ii. same program, different code.
   iii. before terminating, the parent waits for the child to finish its task.
4. Write a program to calculate sum of n numbers using Pthreads.
5. Write a program to generate a Fibonacci Series of numbers using Pthreads.
6. Write a program to implement best-fit and worst-fit allocation strategies
7. Write a program to copy files using system calls and using pthreads and compare timings.
8. Write a program to implement FCFS scheduling algorithm.
9. Write a program to implement SJF scheduling algorithm.
10. Write a program to implement non-preemptive priority based scheduling algorithm.