Course code	**Operating Systems**	**L**	**T**	**P**	**J**	**C**
CSE2005		**2**	**0**	**2**	**4**	**4**
Pre-requisite	**-**	**Syllabus version**				
		v. xx.xx				

Course Objectives:

- Enable the student to appreciate the need to protection, isolation, abstract and virtualization.
- Understand and evaluate trade-offs between conflicting objectives in large scale system design.

Expected Course Outcome:

(1) Interpret the evolution of os functionality, structures and layers.
(2) Apply various types of system calls and to find the stages of various process states.
(3) Design a model scheduling algorithm to compute various scheduling criteria.
(4) Apply land analyse communication between inter process and synchronization techniques.
(5) Implement page replacement algorithms, memory management problems and segmentation.
(6) Differentiate the file systems for applying different allocation and access techniques.
(7) Representing virtualization and recommending real time system.
(8) Understand the various Operating system tasks and the principle algorithms for enumerating those tasks.
(9) Build a working model as a team to solve the challenging problems.

Student Learning Outcomes (SLO):	**2, 14, 17**

Module:1	**Introduction to OS:**	**2 hours**	**SLO: 2**

Functionality of OS - Structuring methods (monolithic, layered, modular, micro-kernel models) - Abstractions, processes, and resources - influence of security, networking, multimedia

Module:2	**OS Principles:**	**3 hours**	**SLO: 2**

System Calls – System/Application Call Interface - Protection User/Kernel modes - Interrupts – Processes and Threads - Structures (Process Control Block, Ready List etc)

Module:3	**Scheduling:**	**5 hours**	**SLO: 17**

Processes Scheduling - CPU Scheduling - Pre-emptive & non-pre-emptive - Resource allocation and management - Deadlocks – Deadlock Handling Mechanisms

Module:4	**Concurrency:**	**4 hours**	**SLO: 17**

Inter-process communication – Synchronization - Implementing Synchronization Primitives – Semaphores - Monitors - Multiprocessors and Locking - Scalable Locks - Lock-free Coordination

Module:5	**Memory management:**	**5 hours**	**SLO: 14**

Main Memory management - Memory allocation strategies – Caching -Virtual Memory Hardware – TLB - Virtual Memory OS techniques – Paging – Segmentation – Page Faults – Page Replacement – Thrashing – Working Set

Module:6	**Virtualization:**	**4 hours**	**SLO: 14**

Virtual Machines – Virtualization (Hardware/Software, Server, Service, Network) – Hypervisors -OS - Container Virtualization - Cost of virtualization

Module:7	**File System**	**3 hours**	**SLO: 17**

File system interface - file system implementation – File system recovery – Journaling - Soft updates & LFS - Distributed file system.

Module:8	**Security and Protection**	**3 hours**	**SLO: 2**

Mechanism Vs Policies – Access and authentication - models of protection – Memory Protection - Disk

Scheduling - OS performance, Scaling OS - Mobile OS:			
Recent Trends: Future directions in Mobile OS / Multi-core Optimization / Power efficient Scheduling.		1 Hour	
	Total Lecture hours:	**30 hours**	

Text Book(s)

1. Remzi H. Arpaci-Dusseau, Andrea C. Arpaci-Dusseau, Operating Systems, Three Easy Pieces, Arpaci-Dusseau Books, Inc (2015).
2. Abraham Silberschatz, Peter B. Galvin, Greg Gagne-Operating System Concepts, Wiley (2012).
3. Ramez Elmasri, A Carrick, David Levine, Operating Systems, A Spiral Approach - McGraw-Hill Science_Engineering_Math (2009).

, press, place

Reference Books

1. Robert Love-Linux Kernel Development-Addison-Wesley Professional (2010)
2. XV6, a simple, Unix-like teaching operating system, Russ Cox, Frans Kaashoek, Robert Morris.
3. Comer, Douglas-Operating System Design_ The Xinu Approach-Chapman & Hall (2015).
4. K. C. Wang (auth.)-Design and Implementation of the MTX Operating System-Springer International Publishing (2015).

Mode of Evaluation:

List of Challenging Experiments (Indicative)		**SLO: 14,17**
1.	In Lab experiments/Project work, the use of GIT repository is mandatory. The XV6 which is a learning OS may be used as a base template for adding OS modules and improving its functionality to create a working OS by the end of the course. Alternate base operating systems could be NachOS/PintOS/Xinu/MTX operating system. Emulators to use are Bochs/QEMU.	X hours
2.	Each experiment should require the student to submit a design document that details the reason for design choices in the module and alternatives considered. The experiment may be submitted before the next lab if not completed within class hours. Collaboration and discussion with co-students on the experiments is encouraged. However, plagiarism will be penalized severely as per University regulations. Every student shall encode his own experiments. • Write a boot loader - to load a particular OS say TinyOS/ KolibriOS image - code to access from BIOS to loading the OS - involves little assembly code – may use QEMU/virtual machines for emulation of hardware. • Recompile kernel with you own program for 'cat'. Your 'cat' should read and display contents of file on screen, check for errors, do it for multiple files while taking input via command line. • Create and execute a system call in user/privileged mode in ARM/X86 processor. Make it part of the kernel. • Allocate/free memory to processes in whole pages, find max allocatable pages, incorporate address translation into the program. • Create an interrupt to handle a system call and continue the previously running process after servicing the interrupt. • Write a Disk driver for the SATA interface. Take care to check readiness of the controller, locked buffer cache, accept interrupts from OS during the period, interrupting the OS again once done and clearing buffers.	X hours

	• Demonstrate the use of locks in conjunction with the IDE driver. • Implement DMA access, measure times required for varying sizes of the files. Compare the times taken for a conventional read write via a OS for the same files. • Create a simple context switching module that switches between processes taking care of all issues. Switch from process kernel thread to scheduler thread and back. • Compare the task creation times. Execute a process and kernel thread, determine the time taken to create and run the threads. • Write a program to put a process to sleep and then wake it up and then kill it when completed. • Run an experiment to determine the context switch time from one process to another and one kernel thread to another. Compare the findings. • Implement a Keyboard driver. • Compare the overhead of a system call with a procedure call. What is the cost of a minimal system call? • Determine the latency of individual integer access times in main memory, L1 Cache and L2 Cache. Plot the results in log of memory accessed vs average latency. • Determine the file read time for sequential and random access based of varying sizes of the files. Take care not to read from cached data - used the raw device interface. Draw a graph log/log plot of size of file vs average per-block time.		
	Total Laboratory Hours	30 hours	
	Project # Generally a team project [3 to 4 members] # Concepts studied in Operating Systems Course should have been used. # Innovative idea should have been attempted. # Design Document on the iterative improvements and design decisions made with justification in Digital format with all figures using software package like Xfig/Ooffice Draw to be submitted. # Assessment on a continuous basis with a min of 3 reviews. **Project List - Illustrative**	**60** [Non Contact hrs]	17

	Measurement Perform a cost / benefit analysis of dynamically loaded libraries vs. statics libraries. Run a standard workload on top of a virtual machine & native OS, measure the performance loss/benefit report the results. Measure the isolation capabilities of different isolation techniques and virtual machine monitors, such as processes, vservers, jails, Xen, and VMware. OS kernels Since the Virtual Machine runs a OS find the services that can be removed from it for redundancies as compared to the Host OS. (eg drivers, memory management etc.) Remove these components, recreate the Virtual Machine, now what is the resulting performance?	
	File Systems Create a file system optimized for a Virtual Machine. Analyze the performance of the Xen Virtual machine monitor. Understand how I/O works in a Virtual environment as compared to a normal OS. Disks fail often, data corruption, sector errors are common. Incorporate modules in your OS that will address disk errors. You may consider checksums, parity checks etc. Simulate the errors and show the OS is able to detect these errors. Show that file system benchmarks like Postmark, Andrew, TPC-B etc., do not do a good job. Determine the type of stress these benchmarks do. Create your own benchmark that will imitate these I/O loads and show how your benchmark is better. You may look into effects of caching, memory load of these benchmarks etc. Analyze how new applications like iTunes, iPhoto, iMovie etc. stresses the file system of OS X, in manners different than the conventional I/O workloads. Security Create a rootkit detection tool that accesses the DMA directly. It should be able to avoid the malicious rootkit's hiding capabilities. Scheduling and Synchronization Current mobiles have heterogeneous cores. Some cores more capable than others. Create a module that can find which thread needs high-performance core.	

	Write a scheduling algorithms that can execute in lower-performance core. Compare the performance of synchronization in two similar OSes (eg Linux , System V). Determine the best kernel for fine-grained locking. (Find the number of locks required to perform a task)	

Mode of evaluation:			
Recommended by Board of Studies	DD-MM-YYYY		
Approved by Academic Council	No. xx	Date	DD-MM-YYYY

CO-PO MAPPING:

	PO 2	**PO 11**	**PO 14**	**PO 17**
CO1	*	*		
CO2		*		*
CO3		*		*
CO4		*		*
CO5		*		*
CO6		*		*
CO7		*		*
CO8			*	
CO9				*

Knowledge Areas that contain topics and learning outcomes covered in the course

Knowledge Area	**Total Hours of Coverage**
CS: OS (Operating Systems)/ CE:OPS (Operating Systems)	15
CS: IAS (Information Assurance and Security)	5
CS: SF (System Fundamental)	6
CS: PD (Parallel and Distributed)	4

Body of Knowledge coverage

KA	Knowledge Unit	Topics Covered	Hours
CS:OS CE:OPS1	OS Overview OPS1 Design principles	Role of OS, functionality, Issues with security.	2
	OS Principles OPS1 Design principles	Structuring methods, design Issues - monolithic, layered, modular, micro-kernel models; Processes; Resources; Program Interfaces; User/Kernel Modes, Interrupts.	2
	OS Concurrency CE-OPS2 Concurrency	Structures (Process Control Block, Ready List etc), dispatching and context switching, the role of interrupts, managing atomic access to OS objects, synchronization primitives, multiprocessor issues.	3
	OS/Scheduling and Dispatch CE-OPS2 Concurrency CE-OPS3 Scheduling and dispatch	CPU Scheduling - pre-emptive & non-pre-emptive, schedulers, deadlocks.	2
	OS/Memory Management CE-OPS4 Memory management	Physical memory and memory management hardware, thrashing – working Set, Caching.	2

	OS/Security and Protection CE-OPS6 Security and protection OS/Virtual Machines OS/File Systems CE-OPS7 File systems OS/System Performance Evaluation	Overview, policy/mechanism separation, Protection, access control, and authentication. Types of virtualization, paging and virtual memory, hypervisors, cost of virtualization. File systems, standard implementation techniques, journaling and log-structured file systems. What is to be evaluated?	2 2 2 1
CS: PD	OS Parallel & Distributed,	Distributed file system, threads, locks, thread concurrency.	4
CS: SF	CS System Fundamentals	Processes and threads, Thread parallelism, virtualization, OS containers, protection, journaling file system, OS Performance.	5
CS: IAS-	CS: IAS(Information Assurance and Security)	Security and Protection - mechanism Vs policies – access and authentication - models of protection – memory protection.	3
		Total hours	**30**

Where does the course fit in the curriculum?

This course is a

- Core Course.
- Suitable from 3th semester onwards.
- Knowledge of any one programming language is essential.
- An understanding of Data Structures is desirable

What is covered in the course?

Module 1: Introduction: This module introduces the functionality of an Operating System, the issues in design of a OS, Different approaches to create the OS, and more importantly the abstraction of all underlying systems.

Module 2: OS Principles: System calls, their interfaces, API's are introduced in this module. The important isolation via use of kernel and user modes is introduced. The module also covers processes, threads and how they are managed.

Module 3: Concurrency: Concurrent access of resources, mechanisms to implement them, issues that arise when we deal with more than one processor/core is discussed here. Use of locks and approaches to Lock free coordination is highlighted.

Module 4: Scheduling of processes, the algorithms for the same, design decisions to pre-empt or not a running process are important concepts discussed. Deadlocks when processes try to access shared resources & mechanisms to break and avoid deadlocks are the issues discussed in this module.

Module 5: Main Memory, its hierarchy, use of caches are introduced in this module. The need for virtual memory concepts, TLB hardware, the use of pages and management of the pages are highlights of the topics discussed in this module.

Module 6: Abstraction, virtualization and relation to main memory are concepts to be discussed here. OS virtualization via containers is a current topic introduced. The cost of virtualization is also to be evaluated in terms of system design.

Module 7: In this module, file systems, its implementation, recovery in case of file system failures, Log Structured file system/journaling for this purpose is highlighted.

Module 8: Important concepts of security of the OS, its protection, policies for the same, authentication models is introduced in this module. OS performance measurements and related issues are discussed.

What is the format of the course?

This Course is designed with 100 minutes of in-classroom sessions per week. The course requires that pre-class reading material be shared with the students. This could be in videos or documents/chapters of a book. It could take a form of flipped classroom also. The student's comprehension of the pre-class reading material will be assessed by a 'in-video' quiz or a short quiz at the beginning of the class. Failure to complete this pre-class work may lead to restriction in allowing classroom participation.

There is a 100-minute lab session per week conducted as described earlier. The course also requires 200 minutes of non-contact time spent on implementing course related project. The key part of the course is that the lab and project components together give a hands-on experience to design an operating system. It can also be evaluated for its performance.

Generally, this course should have the combination of lectures, in-class discussion, case studies, guest-lectures, mandatory pre-class reading material & quizzes.

How are students assessed?

- Students are assessed on a combination classroom discussion(continuous), quizes, lab assignment submissions (8-10 experiments), one group project and continuous, final assessment tests.
- Additional weightage will be given based on innovative projects implemented.
- Students can earn additional weightage based on certificate of completion of a related MOOC course.

Session wise plan

Class Hour	**Lab Hour** (related mapping)	**Topic Covered**	**levels of mastery**	**Reference Book**	**Rema-rks**
1		Introduction to OS: - Functionality of OS - OS Design issues, Structuring methods	Familiarity	2,1	
2		Structuring methods (monolithic, layered, modular, micro-kernel models)	Familiarity	2	
3		Abstractions, processes, and resources - influence of security, networking, multimedia	Familiarity	1	
4	1,3, 13	OS Principles: System Calls – System/Application Call Interface	Usage	1	

5	2	Protection User/Kernel modes - Interrupts – Processes	Usage	1,Ref 1	
6	10,12	Processes and Threads - Structures (Process Control Block, Ready List etc)	Usage, Assessment	1, 2, Ref 1	
7		Scheduling: Processes Scheduling - CPU Scheduling	Usage	2, 1	
8	5,9	CPU Scheduling - Pre-emptive & non-pre-emptive	Assessment	2, 1	
9	11	Resource allocation and management - Deadlocks	Familiarity	2	
10		Deadlocks – Deadlock Handling Mechanisms	Usage	2, 1	
11		Concurrency:Inter-process communication – Synchronization	Familiarity	2	
12		Implementing Synchronization Primitives	Usage	1, 2	
13		Semaphores - Monitors	Assessment	1, 2	
14	7	Multiprocessors and Locking - Scalable Locks - Lock-free Coordination	Usage	1	
15	4,8	Memory management: Main Memory management - Memory allocation strategies – Caching	Familiarity	2	
16	15	Virtual Memory Hardware – TLB - Virtual Memory OS techniques	Usage	2,1	
17		Paging – Segmentation – Page Faults	Usage	2,1	
18		Page Faults – Page Replacement	Assessment	2,1	
19		Page Replacement – Thrashing – Working Set	Familiarity	2,1	
20		Virtualization: Virtual Machines – Virtualization (Hardware/Software, Server, Service, Network)	Familiarity	3	
21		Virtualization (Hardware/Software, Server, Service, Network)	Familiarity	1	
22		Hypervisors -OS virtualization	Familiarity	1	
23		Container Virtualization - Cost of virtualization	Usage	1	Container – ref online

24	6,16	File System – File system interface - file system implementation	Familiarity	1,2	
25		File system recovery – Journaling - Soft updates & LFS	Usage	1,2	online
26		Distributed file system	Familiarity	1,2	
27		Security and Protection - Mechanism Vs Policies – Access and authentication - models of protection	Familiarity	3,1,2	
28		Disk Scheduling	Familiarity	2, 1	
29	14	OS performance, Scaling OS	Usage	2,1	
30		Mobile OS: Future directions.	Familiarity		Online