UE18CS240: INTRODUCTION TO OPERATING SYSTEMS

Course Objectives:

The objective(s) of this course is to,

- Provide an understanding on the various components of an Operating System.
- The course focuses on fundamental problems and optimal solutions for resource management in operating systems such as process, disk and memory management.
- The course will introduce design principles and tradeoffs in the design of Operating Systems.
- The course will also introduce the interface for interacting with a contemporary Operating system such as Linux.

Course Outcomes:

At the end of the course, the student will be able to:

- Gain extensive knowledge on principles and modules of Operating Systems.
- Understand the design of various algorithms for scheduling and their relative performance.
- Design pieces of operating systems such as process management, concurrent processes and threads, memory management and virtual memory.
- Use tools and interface of the operating system.
- Explore design tradeoffs in designing various components of an Operating System.

Course Content:

1. **Introduction and CPU:** What Operating Systems Do? **Introduction to Virtualization of Resources:** CPU/ Memory, Concurrency, Persistence, The Process Abstraction, Process States, Description, Control, API (fork()/ exec()). **Scheduling:** Workload Assumptions, Metrics, Types of Scheduling - FIFO, SJF, Response Time, Round Robin, Multi - Level Feedback Queue. **Case Study:** Linux/ Windows/ UNIX Scheduling Algorithms.
2. **Concurrency:** Introduction and Threads, Types of Threads, Multi - Core/ Multi - Threading, Shared Data. **Thread API:** Thread Creation, Completion, Locks, Condition Variables, Compilation. **Mutual Exclusion and Synchronization:** Software Approaches, Principles of Concurrency, Hardware Support, Semaphores, Message Passing, Readers Writers Problem, pthread Locks. **Deadlocks and Starvation:** Principles of Deadlock, Tools for Detection.
3. **Memory:** Requirements, Partitioning, Paging, Segmentation, Memory API – malloc/ free, Errors. **Virtual Memory**: Hardware and Control Structures, OS Support, Address Translation, Dynamic Relocation, Segmentation, Paging, TLBs, Context Switches, Replacement Policy - LRU, Design Alternatives - Inverted Page Tables, Bigger Pages, Swapping. **Case Study:** Linux/ UNIX Memory Management.
4. **Persistence - I/O Devices:** System Architecture, Canonical Devices/ Protocol – Organization of I/O, CPU Overheads and Interrupts, DMA, OS Design Issues - Device Interaction, Device Driver, Buffering. **Disk Drives:** Performance Parameters - Geometry, I/ O Time Computation, Disk Scheduling Policies, Data Integrity and Protection – Checksum.

5. **File Systems:** File Organization and Access, Directories, Sharing, Security – Access Controls, Record Allocation, Secondary Storage Management. **Case Study:** UNIX/ Windows/ Linux File System. **FS Interface:** Creating/ Reading/ Writing, Random Access, fsync(), Renaming, Hard Links and Symbolic Links, Mounting File Systems. **Security:** Intruders and Malicious Software, Buffer Overflow, OS Hardening, Case Study: UNIX/ Windows.

Pre-requisite Courses: Data Structures.

Reference Book(s):

1. "Operating Systems - Internals and Design Principles", William Stallings, 9th Edition, Pearson, 2018.
2. "Operating Systems: Three Easy Pieces", Remzi Arpaci-Dusseau, Andrea Arpaci Dusseau, http://pages.cs.wisc.edu/~remzi/OSTEP/
3. "Advanced Programming in the Unix Environment", Richard Stevens, Stephen A Rago, Pearson, 3rd edition, 2017.
4. "Operating System Concepts", Abraham Silberschatz, Peter Baer Galvin, Greg Gagne, 9th Edition, John Wiley and Sons, 2013.
5. "Operating Systems", Harvey Deitel, Paul Deitel, David Choffnes, 3rd Edition, Prentice Hall.
6. "Modern Operating Systems", Andrew S Tannenbaum, 3rd Edition, Pearson.