Principle of Operating Systems

Contacts: 4L

Credits: 4

Prerequisites: C programming, Linux basics

Course Description: This course examines the important problems in operating system design and implementation. The operating system provides an established, convenient, and efficient interface between user programs and the bare hardware of the computer on which they run. The operating system is responsible for sharing resources (e.g., disks, networks, and processors), providing common services needed by many different programs (e.g., file service, the ability to start or stop processes, and access to the printer), and protecting individual programs from interfering with one another. The course will start with a brief historical perspective of the evolution of operating systems over the last fifty years and then cover the major components of most operating systems. This discussion will cover the tradeoffs that can be made between performance and functionality during the design and implementation of an operating system. Particular emphasis will be given to three major OS subsystems: process management (processes, threads, CPU scheduling, synchronization, and deadlock), memory management (segmentation, paging, swapping), and file systems; and on operating system support for distributed systems.

Course Topics: will include the following:

Week 1: Overview of operating systems, functionalities and charateristics of OS.

Week 2: Hardware concepts related to OS, CPU states, I/O channels, memory hierarchy, microprogramming

Week 3: The concept of a process, operations on processes, process states, concurrent processes,process control block, process context.

Week 4: UNIX process control and management, PCB, signals, forks and pipes.

Week 5: Interrupt processing, operating system organisation, OS kernel FLIH, dispatcher.

Week 6: Job and processor scheduling, scheduling algorithms, process hierarchies.

Week 7: Problems of concurrent processes, critical sections, mutual exclusion, synchronisation, deadlock, mutual exclusion, process co-operation, producer and consumer processes.

Week 8: Semaphores: definition, init, wait, signal operations.

Week 9: Use of semaphores to implement mutex, process synchronisation etc., implementation of semaphores.

Week 10: Interprocess Communication (IPC), Message Passing, Direct and Indirect

Week 11: Deadlock: prevention, detection, avoidance, banker's algorithm.

Week 12: Memory organisation and management, storage allocation.

Week 13: Virtual memory concepts, paging and segmentation, address mapping.

Week 14: Virtual storage management, page replacemant strategies.

Week 15: File organisation: blocking and buffering, file descriptor, directory structure

Week 16: File and Directory structures, blocks and fragments, directory tree, inodes, file descriptors, UNIX.

Textbook and References:

[1] Silberschatz, Galvin, Gagne, *Operating System Concepts Essentials*, John Wiley & Sons, **2011**.

[2] A. S. Tanenbaum, *Modern Operating Systems*, 3rd ed., Prentice Hall Press, **2007**.

[3] D. Bovet, M. Cesati, *Understanding The Linux Kernel*, 3rd ed., OReilly, **2005**.