

Course Specification Document

Title	Advanced Operating Systems
--------------	----------------------------

Credits	5 ECTS
----------------	--------

Aims	This course aims to explore some of the concepts covered in the fundamentals of operating systems course, providing the student with the knowledge and skills necessary to understand the scheduling algorithms of the CPU, main and virtual memory management, mechanisms for process synchronization, and solving the mutual exclusion problem, and the structure of secondary storage. This equips the student to develop applications that efficiently utilize computer resources and enables them to modify certain system components and engage in kernel-level programming.
-------------	--

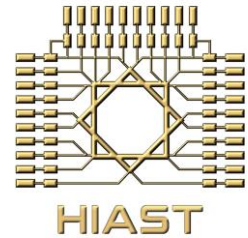
Intended learning outcomes

On successful completion of this course, the student will be able to:

- Explain CPU scheduling algorithms in the majority of well-known operating systems and compare their performance.
- Explain the differences between various models of multithreading.
- Explain advanced methods for managing main and virtual memory, including those adopted in well-known operating systems.
- Understand the challenges of process synchronization and getting acquainted with various methods to achieve this synchronization.
- Understand cases of mutual exclusion between processes and recognize different methods to address them.
- Familiarize himself with the structure of secondary storage and understanding the input/output scheduling algorithms for hard drives
- Build simple Kernel Module and installing it in the system.

Syllabus

- **CPU scheduling:** Review of basic CPU scheduling algorithms, advanced scheduling algorithms such as Multilevel Queue, Multilevel Feedback Queue, Multiple-Processor Scheduling, Real-Time Scheduling, case studies on scheduling in Linux, Solaris, and Windows systems.
- **Threads:** Multithreaded processes, kernel-level and user-level threads, relationship between user-level and kernel-level threads.
- **Process synchronization:** Race conditions, critical sections, semaphores, monitors, case studies on synchronization in Windows, Linux, Solaris.
- **Deadlocks:** System model, description of deadlock phenomenon, methods used to handle deadlocks, resolving deadlock situations.



- **Advanced Memory Management:** Logical and physical addresses, contiguous allocation, segmentation, paging, multi-level paging, paging with segmentation, addressing in a Pentium processor.
- **Virtual Memory:** Demand paging, page replacement, allocation of frames, thrashing.
- **Secondary-Storage:** Disk structure, disk scheduling, disk management, , RAID structure
- **Practical Sessions:**
 - **Linux Kernel:** Introduction to the Linux kernel, exploring the kernel source code, setting up and compiling the kernel.
 - **Makefiles:** Introduction to Makefiles and their structure.
 - **Kernel Module Programming for Linux:** Environment setup, Hello World module, passing command line arguments, compilation using Makefile, linking with the running kernel, executing examples of modules and linking them with the kernel.
 - **Kernel API:** Understanding and describing error codes, working with character strings, memory allocation and deallocation, lists, dealing with locks and critical regions.
 - **Character Device Driver:** Identifying character devices, defining and allocating identifiers, understanding operations and data structures, exchanging data between user and kernel.
 - **I/O Access and Interrupts:** Accessing registers and gates of a device, implementing and registering interrupt handling functions, practical application on the keyboard for kernel-level Keylogger implementation.
 - **Timers, Workqueues and Kernel Threads:** Understanding deferred tasks and their types, data structures for deferred tasks such as Tasklets, data structures for scheduled tasks such as Timers, understanding kernel-level Threads.
 - **Small Project on Kernel Programming.**