COURSE DETAILS

"SISTEMI OPERATIVI"

SSD ING-INF/05

DEGREE PROGRAMME: BACHELOR DEGREE IN COMPUTER ENGINEERING

ACADEMIC YEAR: 2023-2024

GENERAL INFORMATION – TEACHER REFERENCES

TEACHER:
PHONE:
EMAIL:

SEE THE STUDY COURSE WEBSITE

GENERAL INFORMATION ABOUT THE COURSE

INTEGRATED COURSE (IF APPLICABLE): N.A.
MODULE (IF APPLICABLE): N.A.
CHANNEL (IF APPLICABLE): N.A.
YEAR OF THE DEGREE PROGRAMME (I, II, III): III
SEMESTER (I, II): I
CFU: 9

REQUIRED PRELIMINARY COURSES (IF MENTIONED IN THE COURSE STRUCTURE “REGOLAMENTO”)
Programmazione, Calcolatori Elettronici.

PREREQUISITES (IF APPLICABLE)
None.

LEARNING GOALS
The course aims to provide skills on the reference architectures of operating systems; on the methodologies used for resource management in a modern operating system; on tools for system programming; the use of a Unix platform at the user and administrator level; on the basic principles of concurrent programming. The exercises and laboratory activities are developed in the Linux environment and consist of concurrent programming applications and programming of Linux kernel modules.

EXPECTED LEARNING OUTCOMES (DUBLIN DESCRIPTORS)

Knowledge and understanding
The student must demonstrate: to know the problems related to the management of computing resources in multi-user and multi-programmed systems, to understand the methodological tools and basic algorithms used for the realization of high-performance processing systems that efficiently use computing resources, to illustrate the fundamental techniques inherent in concurrent and system programming.

Applying knowledge and understanding
The student must demonstrate to be able to apply the methodological tools learned for the design of processing systems, based on the use of the fundamental abstractions provided by operating systems (eg processes, threads, filesystems, inter-process communication). In addition, the student must demonstrate to be able to apply the methodological tools in order to develop new systems, using the knowledge and techniques of concurrent and system programming, and in order to diagnose the problems of computing systems due to incorrect synchronization and inefficient use of computing resources.

COURSE CONTENT/SYLLABUS
Introductory concepts. Historical evolution of the S.O. - Mono and multiprogramming - Batch, time sharing, real-time, mobile, cloud computing - Recall of architectural elements to support an OS (I/O mode, interruption management, user and supervisor mode, memory hierarchies) - Resource virtualization in the O.S. - The kernel - Call to supervisor - Layered, monolithic, modular, microkernel architectures.

Management and Scheduling of Processes and Threads. Process concept – States of a process – Descriptor of a process – Process queues – The change of context – Process management in Linux and Windows operating systems – Short, medium and long-term CPU scheduling – Evaluation parameters of scheduling algorithms – Starvation – Preemption – First Come First Served, Round Robin, Shortest Process Next algorithms, Shortest Remaining Time, multiple queues with feedback – Comparison between single-processor scheduling algorithms – Traditional UNIX scheduling – The O(1) and CFS schedulers of the Linux operating system – Scheduling in the Windows operating system – Multiprocessor scheduling: SMP, multicore, and hyperthreading architectures, scheduling with load sharing and dynamic load balancing, gang scheduling – Thread concept – Processes and threads – States of a thread – Threads at user level and at the core level – Multithreading programming models – Primitives for thread management – Notes on thread management in Linux, Windows, Java systems.

Concurrent programming. Competition and parallelism – Speed-up in concurrent and parallel architectures – Amdahl's law – Primitive forks/joins – Resource and resource manager concepts – Competition, cooperation, and interference – Race condition and synchronization – Global and local environment models – The interaction between processes in the global environment model – The problem of mutual exclusion: requirements, hardware support and solution. The problem of communication – Traffic lights – Communication via shared memory – The solution of mutual exclusion problems through traffic lights – Problems of cooperation in the global environment model:

producer/consumer problem and solutions through traffic lights, reader/writer problem and solutions through traffic lights – Monitors – Signal and continuous control strategies, signal and wait, and Hoare's solution – The realization of a monitor through traffic lights – The solution of the problems of mutual exclusion, producer/consumer and readers/writers through monitors – The interaction between processes in the local environment model – The primitives for message exchange – Direct and indirect, symmetric and asymmetric communication – Asynchronous and synchronous communication – Asynchronous and synchronous send – Receive blocking and non-blocking – Realization of synchronous send and receive using asynchronous primitives – Servant process – The deadlock problem – Conditions necessary for the deadlock – Methods for managing the deadlock – Deadlock prevention – Deadlock Avoidance and banker's algorithm – Detection and recovery of the deadlock – Comparison of strategies for managing the deadlock.

The Management of the Central Memory. Aspects characterizing memory management: relocation, allocation, organization of virtual space, loading – Swapping – Multiple partition management – Pagination: address translation scheme, pagination architecture, TLB, Page table structure – Segmentation: address translation scheme, segmentation architecture – Segmentation with Pagination – Virtual memory – Pagination on demand – Algorithms for page replacement – Degenerate paging activity (thrashing) – The working set model – Memory management in the Linux operating system: user-space and kernel-space allocators, memory zones, buddy system, page cache, page frame reclaim algorithm – Notes on memory management in the Windows operating system.

I/O management. I/O operations – Virtualization of I/O resources – Device independent layer, device-dependent layer – Drivers – Secondary memory structure – Disks – Disk access scheduling with reference to cylinders and sectors – I/O caching and buffering – I/O scheduling algorithms FIFO, SCAN, and variants – Disk scheduling in Linux OS – RAID architectures – Solid state drives.

File Management. Logical organization of the file system: directories and files - Directory and file operations - Access methods - File descriptor - File sharing - Directory structure for file sharing - Links for sharing - File protection - Logical organization of the file system - File allocation methods: contiguous, linked and indexed list allocation - Free block management - Inode and file management in Unix - The Virtual File Linux system and ext2, ext3 and ext4 file systems – The Windows NTFS File System – Journaling File Systems – Log-structured File System – The Linux F2FS File System for SSD.

Primitives for process management and threads in the UNIX/Linux OS. Primitives for creating and terminating processes: fork, exec, exit, wait – IPC resource management – Primitives for shared memory management – Primitives for traffic light management – use of semop for the realization of primitive wait and signal – Examples of use: solution of problems of mutual exclusion and communication (producer / consumer and readers / writers), object-based and object-oriented realization of a type Monitor – Primitives for the management of queues of messages and use examples – The POSIX Threads primitives for thread management, and use examples.

Insights on the UNIX/Linux operating system. Linux OS installation – The shell and basic commands (filesystem navigation, permission assignment, software installation, program compilation) – Program compilation: makefiles, static and dynamic libraries – Process I/O channels – Pipes – Environment variables – Shell scripting – UNIX signals – Linux kernel configuration and compilation – Development of system calls and kernel modules.

Insights into virtualization and Android mobile OS. Uses of virtualization. Virtualization architectures. CPU virtualization (CPU virtualizability, trap-and-emulate technique, full virtualization using DBT, paravirtualization, Intel VT hardware support). Memory virtualization (shadow page tables, extended page tables). I/O virtualization (full virtualization, PV I/O, I/O passthrough, IO-MMU, SR-VIO). Example of VMware technologies. Android history and design goals. Applications in Android. Process and memory management (life cycle of Activities, Intent, OOM). Application Security Model. Inter-Process Communication in Android.

READINGS/BIBLIOGRAPHY

Textbooks adopted

- Ancillotti, Boari, Ciampolini, Lipari, "*Sistemi Operativi*", McGraw Hill.
- Stallings, "*Operating Systems: Internals and Design Principles", 6th ed.*, Pearson Education

Recommended textbooks

- Silberschatz, Galvin, Gagne, "*Sistemi operativi - sesta edizione*", Addison Wesley
- Tanenbaum, "*I Moderni Sistemi Operativi – terza edizione*", Pearson Education

Didactic handouts and transparencies of the lessons available on the teacher's website.

TEACHING METHODS

The teaching will be delivered through lectures (about two thirds of the total hours of the course), with classroom and laboratory exercises (about one third of the total hours of the course). The lectures will introduce the theoretical aspects related to the abstractions realized by operating systems, to the algorithms for the efficient management of computing resources (CPU, memory, I/O), and to the problems, algorithms and the most common mechanisms for the development of competing systems (eg traffic lights, mutex, monitors, shared memory, message queues). In classroom and laboratory exercises, students will practically deepen the theoretical aspects, independently developing competing programs with the Linux operating system and the C programming language.

EXAMINATION/EVALUATION CRITERIA

a) Exam type:

Exam type	
written and oral	X
only written	
only oral	
project discussion	
other	

In case of a written exam, questions refer to:	**Multiple choice answers**	
	Open answers	
	Concurrent programming exercises	X

b) Evaluation pattern:

The evaluation will take into account in a uniform way both the course of the written test (programming exercises competitor), and the conduct of the oral exam.