SYLLABUS

1. Data about the program of study

1.1 Institution	Technical University of Cluj-Napoca
1.2 Faculty	Faculty of Automation and Computer Science
1.3 Department	Computer Science
1.4 Field of study	Computer Science and Information Technology
1.5 Cycle of study	Bachelor of Science
1.6 Program of study / Qualification	Computer science / Engineer
1.7 Form of education	Full time
1.8 Subject code	47.1

2. Data about the subject

2.1 Subject name	***Operating systems design***				
2.2 Course responsible / lecturer	Assoc. prof. dr. eng. Coleşa Adrian - adrian.colesa@cs.utcluj.ro				
2.3 Teachers in charge of seminars / laboratory / project	Assoc. prof. dr. eng. Coleşa Adrian - adrian.colesa@cs.utcluj.ro As. drd. eng. Császár István - Istvan.Csaszar@cs.utcluj.ro				
2.4 Year of study	IV	2.5 Semester	1	2.6 Type of assessment (E - exam, C - colloquium, V - verification)	E
2.7 Subject category	*DF – fundamentală, DD – în domeniu, DS – de specialitate, DC – complementară*				DS
	DI – Impusă, DOp – opțională, DFac – facultativă				DOp

3. Estimated total time

3.1 Number of hours per week	5	of which: Course	2	Seminars	-	Laboratory	2	Project	1
3.2 Number of hours per semester	70	of which: Course	28	Seminars	-	Laboratory	28	Project	14

3.3 Individual study: 4/	
(a) Manual, lecture material and notes, bibliography	35
(b) Supplementary study in the library, online and in the field	0
(c) Preparation for seminars/laboratory works, homework, reports, portfolios, essays	42
(d) Tutoring	1
(e) Exams and tests	2
(f) Other activities:	0

3.4 Total hours of individual study (suma (3.3(a)...3.3(f)))	80
3.5 Total hours per semester (3.2+3.4)	150
3.6 Number of credit points	6

4. Pre-requisites (where appropriate)

4.1 Curriculum	Operating Systems
4.2 Competence	C programming; Define and use basic OS concepts and system calls

5. Requirements (where appropriate)

5.1. For the course	Blackboard / Whiteboard, Beamer
5.2. For the applications	64-bit Computers with hardware virtualization support, 64-bit Linux and Windows, VMware Workstation, Blackboard / Whiteboard

6. Specific competence

6.1 Professional competences	**C5**: Designing, managing the lifetime cycle, integrating and ensuring the integrity of hardware, software and communication systems • **C5.1**: Specifying the relevant criteria regarding the lifetime cycle, quality, security and the computing system's interaction with the environment and the human operator • **C5.2**: Using interdisciplinary knowledge for adapting the computing system to the specific requirements of the application field • **C5.3**: Using fundamental principles and methods for ensuring the security, the safety and ease of exploitation of the computing systems • **C5.4**: Proper utilization of the quality, safety and security standards in the field of information processing • **C5.5**: Creating a project including the problem's identification and analysis, its design and development, also proving an understanding of the basic quality requirements
6.2 Cross competences	N/A

7. Discipline objective (as results from the *key competences gained*)

7.1 General objective	Provide the students a clear understanding of an OS' internal structure, its main components' role and functionality, and the fundamental OS design and implementation strategies.
7.2 Specific objectives	Let the students: 1. Know and understand the possible OS internal structures. 2. Know and understand the possible design and implementation alternatives of the main OS components, like the scheduler, process and thread manager, memory manager etc. 3. Be able to analyze a specific OS design problems and find solutions to them. 4. Be able to implement in C or assembly different OS components and system calls. 5. Be able to work in team and manage relatively complex software projects.

8. Contents

8.1 Lectures	Hours	Teaching methods	Notes
General structure of an OS. Possible OS structures (monolithic, layered, micro-kernel, virtual machine, exokernel), its components, their functionality, role, interconnectivity.	2	(1) use beamer slides, combined with blackboard illustration; (2) interactions with students: ask their opinion relative to the presented subject; (3) give each class a short evaluation test; let students discuss and argue each other their solution; give them the good solution and let them evaluate their own one;	
Process and thread management (1). Scheduling algorithms. FCFS, SJF, Priority-based, Lottery. Priority inversion.	2		
Process and thread management (2). Scheduling algorithms: RR, MLFQ. Use cases: Solaris, Windows and Linux scheduling policies.	2		
Synchronization mechanisms (1). General Design Principles. Hardware mechanisms used for implementation of higher-level synchronization mechanisms. Design and implementation of locks, semaphores, condition variables. Deadlock avoidance.	2		
Synchronization mechanisms (2). Linux and Windows Use Cases. The synchronization mechanisms provided by Linux and Windows. The way they are implemented.	2		
Synchronization mechanisms (3). Deadlock. Deadlock avoidance, prevention and detection algorithms.	2		
User Application Support (1). Definition of the process concept, system call mechanism and possible implementations, handle (file descriptor) management, basic system calls for process management.	2		

User Application Support (2). Process memory address space structure, argument passing on the stack, process creation strategies, multi-threading support.	2	(4) propose 2-3 interesting study cases of OSes to be prepared and presented by students; (5) students are invited to collaborate in research projects.	
Memory management (1). General aspects, design and implementation alternatives of different memory management techniques and mechanisms: contiguous allocation, segmentation, and paging.	2		
Memory management (2). Paging specific problems like page table hierarchical structure, memory sharing, page tables for Intel architecture.	2		
Memory management (3). Virtual memory's design and implementation aspects: swapping and lazy loading. Page replacement algorithms.	2		
File systems (1). General Design Aspects. Design and implementation alternatives of file systems concepts (files, directories), storage space management. Advantages and disadvantages.	2		
File systems (2). Linux and Windows File Systems. Design and implementation of Ext2 and NTFS.	2		
Security aspect. Subject review. Basic security aspects design. Overview of all presented subjects.	2		

Bibliography

1. A. Silberschatz, G. Gagne, P. B. Galvin, *Operating Systems Concepts*, 7th edition, Wiley, 2005, ISBN 978-0-471-69466-3
2. A. Tanenbaum, A. Woodhull. *Operating Systems Design and Implementation*. 3rd edition, Prentice Hall, 2006, ISBN: 0131429388
3. Daniel Pierre Bovet, *Understanding Linux Kernel*, O'Reilly & Associates, 2001, ISBN 0-596-00002-2.

8.2 Applications – Laboratory	Hours	Teaching methods	Notes
Introduction. Presentation of the lab / project HAL9000 OS.	2	(1) students are presented a very brief overview of the most important and difficult aspects of the working subject; (2) students are given at the beginning of each class a short evaluation quiz; (3) students are given a hands-on tutorial to practice with working subject's aspects and to solve problems (4) students are given challenging problems for extra credit;	
OS Debugging. Techniques and tools to debug an OS.	2		
Thread management. Support for managing multiple executions inside the OS kernel.	2		
Synchronization mechanisms. Implementation of locks, semaphores and condition variables.	2		
Scheduling algorithms. Round-Robin, priority-based, multi-level feedback queue (MLFQ).	2		
User application support (1). System call mechanism. Learn the way the system calls are implemented and used. Basic system call handling in the OS kernel.	2		
User application support (2). Basic memory management. Implementation of basic system calls.	2		
User application support (3). Multi-threaded application support.	2		
Virtual memory (1). Lazy-loading mechanisms.	2		
Virtual memory (2). Memory-mapped files.	2		
Virtual memory (3). Swapping and page-replacement algorithms.	2		
File system (1). Basic aspects of file implementation.	2		
File system (2). Basic aspects of directory implementation.	2		
Lab examination.	2		

8.2 Applications – Project	Hours	Teaching methods	Notes
Overall project presentation: assignments, scheduler, structure,	1	Discuss with students	

expectations, working mode, and evaluation criteria.		their proposed solutions for project assignments	
Assess and grade the design of Threads module	1		
Assess and grade the implementation of Threads module	1		
Assess and grade the design of Userprog module	1		
Assess and grade the implementation of Userprog module	1		
Assess and grade the design of Virtual Memory module	1		
Assess and grade the implementation of Virtual Memory module	1		

Bibliography

1. Lecture slides and laboratory text and support at http://moodle.cs.utcluj.ro/
2. HAL9000 manual.

9. Bridging course contents with the expectations of the representatives of the community, professional associations and employers in the field

OS knowledge is a fundamental requirement in the CS field. The OSD course presents techniques for hardware and software resources management, which are applicable on any complex management software application. Besides, it provides students detailed knowledge about modern OSes' internals, making them capable of developing more efficient applications. We follow the ACM curricula guide. OSD course’s curriculum also maps the IT companies expectations, especially those dealing with direct access to OS services or developing kernel drivers or modules. Such companies are, for instance, system and data security and antivirus detection companies. Usually the teachers in charge of lab classes are former graduate students of our CS program with consistent experience in industry, in companies like those mentioned above. They are permanently consulted regarding the OS course curriculum and its applicability in real projects in industry.

10. Evaluation

Activity type	Assessment criteria	Assessment methods	Weight in the final grade
Course	Students must understand fundamental OS structure and design alternatives and be able to explicitly describe it. They must also be able to apply their knowledge to give solutions to specific OS design problems.	Detailed discussion about design alternatives of different OS components.	50%
Seminar	-	-	-
Laboratory	Minimum attendance and participation in laboratory sessions. Correctness and functionality of the source code for the solved problems. Knowledge of the main data structures and mechanisms of the operating system used	Implementation of different problems in the lab OSD	15%
Project	Consistency and correctness of the design of the developed operating system modules Correctness and functionality of the code corresponding to the operating system modules. Ability to design and implement solutions for extending or improving components of the operating system used	Presentation of design and implementation proposed solutions	25%

Minimum standard of performance

Criteria. Students must be able to describe the internal characteristics and functionality of key operating system (OS) concepts, including locks, priority-based and round-robin (RR) scheduling, system calls, paging, and virtual memory. Students must be able to write functional C code implementing core OS components, such as thread scheduling, system calls, and virtual memory management. The submitted code must **successfully pass at least one test** from the provided test suite.

Attendance requiremenst

Lab attendance

- Minimum 12 lab sessions required to take the final exam in the regular exam session
- Minimum 10 lab sessions required to take the exam in any re-examination session
- Fewer than 10 lab sessions attended → **not eligible** for any lab re-examination during the academic year

Project attendance

- Minimum 6 project sessions required to take the final exam in the regular exam session
- Minimum 5 project sessions required to take the exam in any re-examination session
- Fewer than 5 project sessions attended → **not eligible** for any lab re-examination during the academic year

Lecture attendance

- Minimum 9 lecture sessions required for the regular exam session
- Minimum 7 lecture sessions required for re-examination sessions
- Fewer than 7 lecture sessions attended → **not eligible** for any course re-examination during the academic year

Grading policy

Lab

- Minimum **5** in the final lab exam

Project

- Minimum average **5** for project assignments

Only one project assignment may be submitted **per deadline**, whether during the semester or in any re-examination session.

Overall course

- Final course exam can be taken **only after passing the lab and project component**
- Minimum **5** in the final course exam
- Minimum **5** final average to **pass the course**

Date of filling in 26.02.2025	**Responsible**	**Title First name Last name**	**Signature**
	Course	Assoc.prof.dr.eng. Adrian COLEȘA	
	Applications	Assoc.prof.dr.eng. Adrian COLEȘA	
		As.drd.eng. István CSÁSZÁR	

Date of approval in the department	Head of department, Prof.dr.eng. Rodica Potolea
Date of approval in the Faculty Council	Dean, Prof.dr.eng. Vlad Mureșan