SYLLABUS

1. Data about the program of study

1.1 Institution	The 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. Adrian Coleşa – adrian.colesa@cs.utcluj.ro				
2.3 Teachers in charge of seminars/ laboratory/ project	Assoc. prof. dr. eng. Adrian Coleşa – adrian.colesa@cs.utcluj.ro Eng. Radu Portase – rportase@bitdefender.com Eng. Istvan Szekely – iszekely@bitdefender.com Eng. David Acs – dacs@bitdefender.com Eng. Balint Szabo – bszabo@bitdefender.com Eng. Laslo Ciople – lciople@bitdefender.com Eng. Bogdan Ionuț Lazăr – bilazar@bitdefender.com Eng. Istvan Csaszar - icsaszar@outlook.com				
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:										
(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; (4) propose 2-3	
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		
Process management (1). Definition of the process concept, system call mechanism and possible implementations, handle (file descriptor) management, basic system calls for process management.	2		

Process management (2). Process memory address space structure, argument passing on the stack, process creation strategies, multi-threading support.	2	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 – Seminars/Laboratory/Project	Hours	Teaching methods	Notes
Introduction. Presentation of the lab / project OS (Pintos or HAL9000).		(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.			
Thread management. *Support for managing multiple executions inside the OS kernel.*			
Synchronization mechanisms. *Implementation of locks, semaphores and condition variables.*			
Scheduling algorithms. Round-Robin, priority-based, multi-level feedback queue (MLFQ).			
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.			
User application support (2). Basic memory management. Implementation of basic system calls.			
User application support (3). Multi-threaded application support.			
Virtual memory (1). Lazy-loading mechanisms.			
Virtual memory (2). Memory-mapped files.			
Virtual memory (3). Swapping and page-replacement algorithms.			
File system (1). Basic aspects of file implementation.			
File system (2). Basic aspects of directory implementation.			
Lab examination.			

Bibliography

1. Lecture slides and laboratory text and support at http://moodle.cs.utcluj.ro/
2. Pintos and 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.	Online quiz tests using the Moodle platform and oral examination. Detailed discussion about design alternatives of different OS components. In exceptional situations (like those imposed by government for self-isolation and remote school activities), the online examination could also be taken by students from a remote location using Moodle and Teams platforms.	0.67
Seminar	-	-	-
Laboratory Project	Students must be able to develop different OS components writing code in C and assembly.	*Lab*: implementation of different problems in the lab OS. *Project*: presentation of design and implementation solutions. In exceptional situations (like those imposed by government for self-isolation and remote school activities), the online examination could also be taken by students from a remote location using Moodle and Teams platforms.	0.33

Minimum standard of performance:

Students must attend minimum **9 lecture classes** to be allowed to take the exam in the regular exam session. Students must attend minimum **7 lecture classes** to be allowed to take the exam in any re-examination sessions. Less than 7 attended lecture classes leads to the interdiction to take any course re-examination in the university year the course is taught.

Students must attend minimum **12 lab classes** to be allowed to take the exam in the regular exam session. Students must attend minimum **10 lab classes** to be allowed to take the exam in any re-examination sessions. Less than 10 attended lab classes leads to the interdiction to take any lab re-examination in the university year the course is taught.

Students must submit solutions for **at least 3 project assignments** (from the total no of 6 assignments) and receive at least 5 for each submitted assignment.

Students are allowed to take the final course examination only after passing the lab and project examination.

Be able to describe the internal aspects of the fundamental OS design principles, like locks, priority-based and RR scheduling, system calls, paging, virtual memory.

Be able to write functional C code that pass at least one test from the provided test set.

Date of filling in	Responsible	Title, first name, family name	Signature
	Course	Assoc.prof. dr. eng. Adrian Coleşa	
	Applications	Assoc.prof. dr. eng. Adrian Coleşa	
		Eng. Radu Portase	
		Eng. Istvan Szekely	
		Eng. Bogdan Ionuț Lazăr	
		Eng. David Acs	
		Eng. Balint Szabo	
		Eng. Laslo Ciople	
		Eng. Istvan Csaszar	

Date of approval in the department	Head of department Prof.dr.ing. Rodica Potolea
Date of approval in the Faculty Council	Dean Prof.dr.ing. Liviu Miclea