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	26.

2. Data about the subject

2.1 Subject name	***Operating Systems***				
2.2 Course responsible/lecturer	Conf. dr. eng. Adrian Coleşa – adrian.colesa@cs.utcluj.ro				
2.3 Teachers in charge of seminars/ laboratory/ project	Conf. dr. eng. Adrian Coleşa – adrian.colesa@cs.utcluj.ro Eng. Istvan Szekely – iszekely@bitdefender.com Eng. David Acs – dacs@bitdefender.com Eng. Laslo Ciople – lciople@bitdefender.com Eng. Lilla Nagy – lnagy@bitdefender.com				
2.4 Year of study	II	2.5 Semester	2	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*				DD
	DI – Impusă, DOp – opțională, DFac – facultativă				DI

3. Estimated total time

3.1 Number of hours per week	4	of which:	Course	2	Seminars		Laboratory	2	Project	
3.2 Number of hours per semester	56	of which:	Course	28	Seminars		Laboratory	28	Project	
3.3 Individual study:										
(a) Manual, lecture material and notes, bibliography										25
(b) Supplementary study in the library, online and in the field										10
(c) Preparation for seminars/laboratory works, homework, reports, portfolios, essays										28
(d) Tutoring										2
(e) Exams and tests										4
(f) Other activities:										0
3.4 Total hours of individual study (suma (3.3(a)…3.3(f)))						69				
3.5 Total hours per semester (3.2+3.4)						125				
3.6 Number of credit points						5				

4. Pre-requisites (where appropriate)

4.1 Curriculum	Computer Programming, Data Structures and Algorithms
4.2 Competence	C programming

5. Requirements (where appropriate)

5.1. For the course	Blackboard / Whiteboard, Beamer
5.2. For the applications	Computers, Linux, Windows, Blackboard / Whiteboard

6. Specific competence

6.1 Professional competences	**C3:** Problems solving using specific Computer Science and Computer Engineering tools (3 credits) • **C3.1**: Identifying classes of problems and solving methods that are specific to computing systems • **C3.2**: Using interdisciplinary knowledge, solution patterns and tools,

	making experiments and interpreting their results • **C3.3**: Applying solution patterns using specific engineering tools and methods • **C3.4**: Evaluating, comparatively and experimentally, the available alternative solutions for performance optimization • **C3.5**: Developing and implementing informatic solutions for concrete problems **C4**: Improving the performances of the hardware, software and communication systems (2 credits) • **C4.1**: Identifying and describing the defining elements of the performances of the hardware, software and communication systems • **C4.2**: Explaining the interaction of the factors that determine the performances of the hardware, software and communication systems • **C4.3**: Applying the fundamental methods and principles for increasing the performances of the hardware, software and communication systems • **C4.4**: Choosing the criteria and evaluation methods of the performances of the hardware, software and communication systems • **C4.5**: Developing professional solutions for hardware, software and communication systems based on performance optimization
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 what an OS is, its role and general functionality and the ability to use fundamental system calls of an OS.
7.2 Specific objectives	Let the students: 1. Know and understand the OS specific terminology. 2. Understand the general structure and functionality of an OS. 3. Understand the specific functionality of the most important OS components, like shell, process manager, file system, memory manager, security manager. 4. Understand the functionality of main synchronization mechanisms and be able to use them to solve real synchronization problems. 5. Be able to write C programs to use an OS's (Linux and Windows) system calls.

8. Contents

8.1 Lectures	Hours	Teaching methods	Notes
Introduction and basic concepts. OS's definition, role, evolution, components, main concepts (file, process, system calls). Basic hardware aspects: CPU, user and kernel mode, memory layers, I/O devices. Basic OS structure.	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	
The Shell (Command Interpreter). Definition, role, functionality, simple and complex commands. Standard input and output redirection.	2		
File systems (1). User Perspective. File and directory concept from the user point of view (definition, role, characteristics, operations).	2		
File systems (2). Windows and Linux File Systems. Permission rights and system calls.	2		
File systems (3). Implementation aspects. Implementation strategies overview, space management and related problems, hard and symbolic links.	2		
Process management. Process model: definition, role, characteristics. Linux and Windows process management system calls.	2		
Thread management. Thread model: user vs. kernel threads,	2		

implementation problems, usage, performance aspects. Basic scheduling algorithms (FIFO, SJF, Priority-based). Linux and Windows process thread system calls.		evaluate their own one; (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.	
Process synchronization (1). Theoretical aspects. Context, definition, synchronization mechanisms, techniques and problems (locks, semaphores, monitors, mutual exclusion, starvation, deadlock).	2		
Process synchronization (2). Classical synchronization patterns: producer/consumer, readers/writers, rendez-vous, barrier, dining philosopher, sleeping barber. Similarities between different synchronization mechanisms.	2		
Inter-process communication. Pipe files, shared memory, message queues, signals.	2		
Memory management (1). Context, definition, binding, basic techniques, space management, addresses translation, swapping.	2		
Memory management (2). Paging and segmentation.	2		
I/O Devices Management. Principles, disks, clocks, character-oriented terminals.	2		
Security aspects. Security policies and mechanisms. Basic program's vulnerabilities (buffer overflow).	2		

Bibliography

1. Andrew Tanenbaum. *Modern Operating System*, 2nd Edition, Prentice-Hall, 2005, ISBN 0-13-092641-8.
2. A. Silberschatz, P. Galvin, G. Gagne, Operating Systems Concepts, 8th Edition, Wiley, 2010
3. Remzi H. Arpaci-Dusseau, Andrea C. Arpaci-Dusseau, Operating Systems: Three Easy Pieces, online available at http://pages.cs.wisc.edu/~remzi/OSTEP/

8.2 Applications – Seminars/Laboratory/Project	Hours	Teaching methods	Notes
Laboratory presentation: Purpose, contents, strategies, requirements. **Get familiar with Linux OS**: main characteristics, basic commands, access rights.	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;	
Linux batch scripts: basic Linux commands, command line structure, scripts, command line parameters, variables, control flow commands, functions.	2		
Linux system calls to access data in files: basic system calls to store and retrieve data to and from regular user files: open, read, write, lseek, close.	2		
Linux system calls for file and directory manipulation: system calls to rename or remove a file, link a file to more directories, get information about a file or directory, change permission rights and listing a directory contents.	2		
Linux system calls for process management: system calls for creating a new process, terminating an existing process, waiting for a child process to terminate, loading another executable into an existing process etc.	2		
Linux threads: Linux implementation of POSIX functions used to create and manage threads: pthread_create, pthread_join, pthread_exit etc.	2		
Synchronization mechanisms (1): Linux semaphores. Linux system calls to create and use semaphores: semget, semctl, semop.	2		
Synchronization mechanisms (2): POSIX locks and condition variables. Linux functions used to create and use POSIX locks and condition variables: pthread_mutex_lock, pthread_mutex_unlock, pthread_cond_wait, pthread_cond_signal.	2		
Inter-process Communication Mechanisms (IPC): Linux named (FIFO) and nameless pipes. System calls for managing and using pipes: pipe and mkfifo.	2		
Memory management: ELF executable file format. Virtual vs.	2		

physical address space. Dynamically allocated memory.			
Memory management: memory-mapped files, shared memory.	2		
Security aspects: buffer overflow detection and correction.	2		
Subject review and exam simulation.	2		
Lab examination	2		

Bibliography

1. Lecture slides and laboratory text and support at http://moodle.cs.utcluj.ro/
2. M. Mitchell, J. Oldham, A. Samuel, Advanced Linux Programming, New Riders Publishing, 2001

*Se vor preciza, după caz: tematica seminariilor, lucrările de laborator, tematica și etapele proiectului.

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. We follow the ACM curricula guide. We also consult relevant IT companies about their practical expectations regarding OS knowledge and adapt accordingly our course contents. In this sense, Linux and Windows are the most used OSes. Usually the teachers in charge of lab classes are former graduate students of our CS program with consistent experience in industry. 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 concepts and be able to correctly define them. They must also be able to apply their knowledge to solve user-space problems related to or dependent by an OS.	Quiz tests during lecture classes. Written or online examination using platforms like Moodle and Teams, with subjects requiring students to apply the theoretical learned OS related aspects to give a solution to proposed problem. 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.	0.67
Seminar			
Laboratory	Students must be able to develop C programs that use different OS system calls to solve practical, problems related to or dependent by an OS.	Quiz tests during lab classes. Lab assignments during semester. Final lab examination with programming problems, whose solution has to be implemented in C and run on computers. In exceptional situations (like those imposed by government for self-isolation and remote school activities), the lab final examination could also be taken by students from a remote location	0.33
Project			

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 are allowed to take the final course examination only after passing the lab examination.
Be able to define the fundamental OS principles and concepts, like process, thread, file, directory, lock, semaphore, paging.
Be able to write C program to use fundamental system calls in Linux for working with files, processes, threads, synchronization mechanisms and memory.

Date of filling in	**Responsible**	**Title, first name, family name**	**Signature**
	Course	Conf.dr.eng. Adrian Colesa	
	Applications	Conf.dr.eng. Adrian Colesa	
		Eng. Istvan Szekely	
		Eng. David Acs	
		Eng. Laslo Ciople	
		Eng. Lilla Nagy	

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