



DISCIPLINE ACADEMIC SHEET
ACADEMIC YEAR 2025 – 2026

1. PROGRAMME DATA

1.1. Higher Education Institution	UNIVERSITY OF CRAIOVA
1.2. Faculty	Automation, Computers and Electronics
1.3. Department	Computers and Information Technology
1.4. Field of Study	Computers and Information Technology
1.5. Study level	L(License / Undergraduate)
1.6. Organizational form	IF
1.7. Study Program (name/code) /Qualification	Computers / L2060101010

2. DISCIPLINE DATA

2.1 Discipline Name	Operating Systems						
2.2 Course Activities Holder	conf.dr. ing. Stefan Udristoiu						
2.3. Practical Activities Holder	conf.dr. ing. Stefan Udristoiu						
2.4. Study Year	3	2.5. Semester	1	2.6. Evaluation type	E	2.7. Discipline regime	DOB

3. ESTIMATED TOTAL TIME (hours per semester of teaching activities)

3.1. Number of hours per week	5	in which: 3.2 course	3	3.3. sem./lab./pr.	0/2/0
3.4. Total hours of curriculum	70	in which: 3.5 course	42	3.6. sem./lab./pr.	0/28/0
3.7. Time distribution - hours/week					hours
Study after manual, course support, bibliography and notes					20
Additional documentation in library, on specialized electronic platforms and field					10
Training seminars / labs, homework, portfolios and essays					8
Tutorial					2
Examinations					9
Other activities.....					6
Total hours per individual activities					55
3.8. Total hours per semester					125
3.9. Number of credits					5

4. PRECONDITIONS (where appropriate)

4.1. of curriculum	The prerequisites for this discipline are: Computer Programming, Programming Techniques, Data Structures, Object Oriented Design, and Computer's Architecture
4.2. of competence	The student is required to have fundamental knowledge about programming, specifically in Java and C/C++ languages, and about hardware structure and hardware operation.

5. CONDITIONS (where appropriate)

5.1. of the course	Whiteboard and video projector helpful but not mandatory.
5.2. of seminar/ laboratory/project	Computers with Windows and Linux operating systems (or SSH access to a Linux system), plus development tools for C/C++ and Java.

6. COURSE OBJECTIVES – expected learning outcomes that are supported by completing and passing the course

Knowledge	Studentul/absolventul descrie, identifică, sumarizează concepte și metode elementare privitoare modelarea, analiza, proiectarea și testarea sistemelor de calcul, cu microcontrolere sau procesoare, sistemelor de operare, sistemelor de prelucrare grafică și a sistemelor de achiziție date. Studentul/absolventul elaborează modele pentru diferite componente ale sistemelor de calcul, hardware și software și evaluează caracteristicile funcționale și nefuncționale. Studentul/absolventul utilizează metode și instrumente specifice pentru analiza, proiectarea și implementarea sistemelor de achiziție, de prelucrare grafică, de prelucrare și afișare a datelor . Studentul/absolventul proiectează și implementează sisteme funcționale de complexitate mică/medie cu microprocesoare.
Skills (Abilities)	Studentul/absolventul descrie, identifică, sumarizează concepte și metode elementare privitoare modelarea, analiza, proiectarea și testarea sistemelor de calcul, cu microcontrolere sau procesoare, sistemelor de operare, sistemelor de prelucrare grafică și a sistemelor de achiziție date. Studentul/absolventul elaborează modele pentru diferite componente ale sistemelor de calcul, hardware și software și evaluează caracteristicile funcționale și nefuncționale. Studentul/absolventul utilizează metode și instrumente specifice pentru analiza, proiectarea și implementarea sistemelor de achiziție, de prelucrare grafică, de prelucrare și afișare a datelor . Studentul/absolventul proiectează și implementează sisteme funcționale de complexitate mică/medie cu microprocesoare.
Responsibility and autonomy	Studentul/ absolventul arată spirit de inițiativă și acțiune pentru actualizarea cunoștințelor profesionale, economice și de cultură organizațională.

7. CONTENT

7.1. Course	The method of implementation	Teaching methods	Allocated time (hours)
1. Introduction Operating Systems Principles. 1.2. Operating Systems Classification. 1.3. Computer Hardware Structure. 1.4. Operating Systems Structure.	face to face (week 1)	Explaining, Demonstrating, Collaborating	3
A. Processes in Linux	face to face (week 2)	Explaining, Demonstrating, Collaborating	3
B. Pipes in Linux	face to face (week 3)	Explaining, Demonstrating, Collaborating	3
C. Linux Threads	face to face (week 4)	Explaining, Demonstrating, Collaborating	3

D. Linux Synchronization (Mutexes, Conditions and Semaphores)	face to face (week 5)	Explaining, Demonstrating, Collaborating	3
E. Classical Problems implemented in Linux (Producer/Consumer, Philosophers, Barber etc)	face to face (week 6)	Explaining, Demonstrating, Collaborating	3
F. Java Threads and Synchronization	face to face (week 7)	Explaining, Demonstrating, Collaborating	3
G. Classical Problems implemented in Java (Producer/Consumer, Philosophers, Barber etc)	face to face (week 8)	Explaining, Demonstrating, Collaborating	3
H. Windows Threads and Synchronization	face to face (week 9)	Explaining, Demonstrating, Collaborating	3
I. Classical Problems implemented in Windows (Producer/Consumer, Philosophers, Barber etc)	face to face (week 10)	Explaining, Demonstrating, Collaborating	3
J. CPU Scheduling Algorithms; Network management and programming	face to face (week 11)	Explaining, Demonstrating, Collaborating	3
K. Memory Management	face to face (week 12)	Explaining, Demonstrating, Collaborating	3
L. File Systems and disk management	face to face (week 13)	Explaining, Demonstrating, Collaborating	3
M. Concurrent access on files.	face to face (week 14)	Explaining, Demonstrating, Collaborating	3
Bibliography:			
1. Abraham Silberschatz, Greg Gagne, Peter B. Galvin - Operating System Concepts 10th Edition (UCV Library)			
2. Andrew Tanenbaum , Herbert Bos - Modern Operating Systems 4th Edition - https://csc-knu.github.io/sys-prog/books/Andrew%20S.%20Tanenbaum%20%20Modern%20Operating%20Systems.pdf			

7.2. Seminar/laboratory/project	The method of implementation	Teaching methods	Allocated time (hours)
1. Introduction in Linux (basic commands, editing, compiling and executing programs)	face to face (week 1)	Problem solving. Teacher's constant feedback.	2
2. Processes and threads in	face to face	Problem solving. Teacher's	2

Linux	(week 2)	constant feedback.	
3. Linux pipes.	face to face (week 3)	Problem solving. Teacher's constant feedback.	2
4. Linux synchronization (mutexes, condition variables, semaphores)	face to face (week 4-5)	Problem solving. Teacher's constant feedback.	4
5. Java synchronization (threads, monitors, condition variables)	face to face (week 6-7)	Problem solving. Teacher's constant feedback.	4
6. Windows synchronization (threads, monitors, events, semaphores, critical sections, wait functions)	face to face (week 8-9)	Problem solving. Teacher's constant feedback.	4
7. Concurrent file access in Linux.	face to face (week 10-11)	Problem solving. Teacher's constant feedback.	3
8. Concurrent file access in Windows	face to face (week 11-12)	Problem solving. Teacher's constant feedback.	3
9. Sockets in Linux	face to face (week 13)	Problem solving. Teacher's constant feedback.	2
10. Sockets in Windows	face to face (week 14)	Problem solving. Teacher's constant feedback.	2

Bibliography:

1. The Linux Kernel documentation - <https://docs.kernel.org/>
2. The Single UNIX® Specification - <https://pubs.opengroup.org/onlinepubs/7908799/index.html>
3. MSDN - <https://learn.microsoft.com/en-us/windows/win32/sync/synchronization>
4. Java Documentation - <https://docs.oracle.com/en/java/>

8. Correlating the content of the discipline with the expectations of representatives from the epistemic community, professional associations and representative employers in the field related to the program

The content of the discipline is constantly updated based on my experience as a software developer and software architect (30 years of experience working for Novus, MuseGlobal, Netdania and ITSIX) and from continuous feedback from fellow programmers from most of the software companies based in Craiova.

9. EVALUATION

Activity Type	9.1. Evaluation criteria	9.2. Evaluation methods	9.3. Final mark weight
	Level and quality of student interaction and feedback during classes. Problem solving skills.	Asking constantly for solutions. Inserting defects in solutions and asking students to discover them and correcting them. Final writing exams consisting of problems covering the most important chapters (processes and pipes, linux and java synchronization, windows synchronization)	The final mark represents the average of the maximum mark obtained for each chapter during the semester (including laboratory, seminar and course activities result) and the final exam.

	L: Problem solving skills and the level of understanding of theoretical concepts. The level of work and constancy is also recorded.	During the entire semester, students are required to solve problems and to present them, responding to a series of “What if?” and “Why” questions.	The final mark represents the average of the maximum mark obtained for each chapter during the semester (including laboratory, seminar and course activities result) and the final exam.

9.6. Minimum standard of performance

The minimum standard of performance consists of abilities to solve problems which require starting and waiting processes, creating and joining threads which performs activities that do not require synchronizations and communicating simple messages using pipes. They can solve those problems during laboratories (or seminar activities) or at the final exam.

Date of Completion	Signature of the Course Activities Holder,	Signature of the Practical Activities Holder,
23.08.2025	conf.dr. ing. Stefan Udristoiu	conf.dr. ing. Stefan Udristoiu

Date of Approval in the Department	Signature of the Department Director,
25.08.2025	Ș.I.dr.ing. Nicolae ENESCU

Date of Approval in the Faculty	Signature of the Dean,
	Prof.dr.ing. Dorin ȘENDRESCU