COURSE DESCRIPTION CARD - SYLLABUS

Course name

Operating Systems with Concurrency Programming

Course

Field of study	Year/Semester
Artificial Intelligence	1/2
Area of study (specialization)	Profile of study
	general academic
Level of study	Course offered in
First-cycle studies	English
Form of study	Requirements
full-time	compulsory

Number of hours

Lecture	Laboratory classes	Other (e.g. online)
30	30	
Tutorials	Projects/seminars	

Number of credit points

5

Lecturers

Responsible for the course/lecturer:	Responsible for the course/lecturer:
Dariusz Wawrzyniak, Ph.D.	Cezary Sobaniec, Ph.D.

Prerequisites

The student starting this module should have a basic knowledge of the computer structure and its working principle, selected elements of discrete mathematics, imperative programming skills (especially in C programming language) including implementation of simple algorithms. In respect to the social skills the student should show attitudes as honesty, responsibility, curiosity, and creativity.

Course objective

1. To acquaint students with theoretical and practical problems of the design and implementation of operating systems, especially resource management (e.g. processor, memory, I/O devices).
2. To teach students how to use a Unix-like operating system.
3. To develop the skills in concurrent programming as well as system programming including multitasking and multithreading, synchronisation mechanisms, and deadlock problem.

Course-related learning outcomes

Knowledge

1. has theoretical knowledge of operating systems working,
2. has basic knowledge regarding trends in operating systems,

3. has well-established knowledge of concurrent programming problems and hazards arising from inappropriate synchronisation.

Skills
1. can write concurrent programs — both process-based and threaded-based — applying inter-process communication and synchronisation mechanisms provided by an operating system,
2. can use basic commands of a Unix-like operating system, combine them into pipelines and scripts.

Social competences
1. understands that knowledge and skills related to computer science may become obsolete,
2. is aware of IT systems failures that led to major financial or social losses, or caused damage to health or even death.

Methods for verifying learning outcomes and assessment criteria

Learning outcomes presented above are verified as follows:
Lecture: Test with multiple-choice or open-ended questions with about 100 points to score and 50 points to pass.
Classes: practical assignments and tests.

Programme content

The lecture covers the following topics:
1. The definition and the functions the operating system, the classification of operating systems, system software structure and its relationship with the hardware, the principle of operation of system kernel.
2. File system
a) logical organization: the definition of the file and its attributes, access methods to a file, the interface for file operations, logical directory structure.
b) physical organization: disk block allocation (contiguous, chained, and indexed), free space handling (bit vector, linked list, grouping, counting), the implementation of a directory (linear list, hash table, index structure), implementation of file operations (buffer cache, the problem of integrity, concurrent access to a file).
3. The overall concept of a resource management and the notion of process and thread.
4. Concurrency programming:
a) concurrent programming abstraction: atomic operations and its interleaving,
b) general correctness conditions: safety and liveness,
c) mutual exclusion: problem formulation and its solution by means of atomic read and write operations on shared memory location (Peterson’s algorithm and Lamport’s algorithm),
d) architectural support: disabling interrupts, complex atomic operation (test-and-set, exchange),
e) operating system support: binary semaphores, counting semaphores, mutex locks, conditional variables,
f) language support: monitors, conditional critical regions,
f) classical synchronization problems: producer-consumer, readers-writers, dining philosophers, sleeping barber’s.
5. Resource management:

a) processor management: CPU scheduling, scheduling criteria and algorithms,
b) memory management: memory organization, memory allocation, creation of process image in memory, paging and segmentation, virtual memory,
c) management of I/O devices: classification of input/output devices, the structure of the I/O mechanism, the interaction between CPU and I/O devices, buffering and spooling.
6. Deadlock: system model, resource classification, definition, necessary conditions, deadlock detection, prevention and avoidance.

Laboratory classes cover the following topics:
1. Introduction to Unix-like operating system usage, system manual, the shell, editors.
2. Unix file system usage: directory structure, file operations, file types, access rights, searching for files.
3. Processes: priorities, signals, concurrent processes management.
4. Inter process communication using pipes: basic Unix filters and complex pipeline compositions.
5. Bourne's shell: environment variables, redirections, aliases, script programming constructs, functions, input processing.
6. Introduction to programming in Unix-like operating systems: the C compiler, error handling.
7. Processing file contents: file descriptors, opening files, reading and writing, implementation of simple Unix tools.
8. Managing processes: process creation, running external programs, basic coordination, redirection of standard streams.
9. Inter process communication using signals: handling of signals, stopping processes.
10. Inter process communication using pipes.
11. Inter process communication using shared memory.
12. Process synchronization using semaphores.
13. Multithreaded programming: thread creation and management.
14. Synchronization of threads: mutexes, conditional variables.

Teaching methods

1. Lectures: presentation of slides (multimedia showcase), discussion of problems, solving tasks on blackboard.
2. Classes: solving tasks, practical exercises, discussion, conducted in a computer laboratory under the control of Unix-like operating system.

Bibliography

Basic

1. Abraham Silberschatz, Greg Gagne, Peter B. Galvin: Operating System Concepts, 10th edition, John Wiley & Sons, 2018.
2. Andrew S. Tanenbaum, Herbert Bos: Modern Operating Systems, 4th edition, Prentice Hall, 2014.
3. William Stallings: Operating Systems, 9th edition, Pearson, 2018.
4. Michael Kerrisk: The Linux Programming Interface – A Linux and UNIX System Programming Handbook. No Starch Press, 2010.

Additional

1. Gary Nutt: Operating Systems, 3rd edition, Pearson, 2004.

2. Mordechai Ben-Ari: Principles of Concurrent and Distributed Programming, Addison Wesley, 2006.

3. Arnold Robbins: Unix in a Nutshell. O'Reilly Media, 2005.

Breakdown of average student's workload

	Hours	ECTS
Total workload	110	5,0
Classes requiring direct contact with the teacher	60	3,0
Student's own work (literature studies, preparation for laboratory classes/tutorials, preparation for tests/exam, project preparation) [1]	50	2,0

[1] delete or add other activities as appropriate