COURSE DESCRIPTION CARD - SYLLABUS

Course name

Principles of Concurrent Programming

Course

Field of study	Year/Semester
Bioinformatics	2/3
Area of study (specialization)	Profile of study
	general academic
Level of study	Course offered in
First-cycle studies	Polish
Form of study	Requirements
full-time	elective

Number of hours

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

Number of credit points

4

Lecturers

Responsible for the course/lecturer:
Dariusz Wawrzyniak
Institute of Computing Science, PUT
ul. Piotrowo 2, 60-965 Poznań
e-mail: Dariusz.Wawrzyniak@cs.put.poznan.pl

Responsible for the course/lecturer:
Anna Kobusińska
Institute of Computing Science, PUT
ul. Piotrowo 2, 60-965 Poznań
e-mail: Anna.Kobusinska@cs.put.poznan.pl

Prerequisites

The student starting this module should have a basic knowledge of the computer structure and its working principle, imperative programming skills, including implementation of simple algorithms and their complexity assessment. With respect to social skills, the student should show attitudes as honesty, responsibility, perseverance, curiosity, and creativity.

Course objective

1. To acquaint students with basic theoretical knowledge related to concurrent processing in computer systems and practical aspects of the implementation of concurrent processing in such systems.
2. To develop students' skills in solving problems related to concurrent computing in computer systems.

Course-related learning outcomes

Knowledge

1. Understands fundemental conepts of concurrent computing in operating systems (e.g. indeterminism,

deadlock).
2. Has basic knowledge of has basic knowlega of structured and object-oriented programming related to concurrent processing.
3. Has basic knowledge of combinatorial optimisation in concurrent processing.
4. Has basic knowledge of computer systems life cycle.

Skills

1. Is able to design a concurrent programs following a given specification, using appropriate methods, techniques and tools.
2. Is able to carry out an analysis of functionality and requirements of information processing systems in respect of concurrency issues.
3. Is able to gain information from literature, databases and other information sources (both in the native language and English).

Social competences

1. Understands the need for learning throughout their lives and enhance their competence.
2. Is able to collaborate and cooperate in a team fulfilling different roles.

Methods for verifying learning outcomes and assessment criteria

Learning outcomes presented above are verified as follows:

Formative assessment

a) Lectures:

• based on answers to questions related to the issues discussed at previous lectures.

b) Laboratory classes:

• evaluation of the student’s preparation for each laboratory session, and their skills associated with the performance of laboratory tasks,

• evaluation of knowledge and skills acquired at the laboratory classes based on two written tests in the semester.

Total assessment

a) Lectures:

• Evaluation of acquired knowledge based on the written exam consisting of 4 – 5 open-end questions with about 20 – 30 points to score for each question, agregating to 100 points for the whole exam. To get a passing grade in the exam a student must earn a minimum of 50% of the maximum score (i.e. 50 points).

• Discussion (on demand) of correct answers to the exam questions.

b) Laboratory classes:

• calculation of the evaluation in the form of a weighted arithmetic average: the weight of each of the two written tests conducted in a semester is 5, the weight of entrance tests is 2, and the weight obtained in the result of the evaluation of student’s knowledge necessary to prepare, and carry out the lab tasks is 1.

Additional elements cover:

• discussing more general and related aspects of the class topics,

• effective use of the knowledge gained during solving the given problem,
• comments leading to the improvement of the teaching materials and teaching process.

Programme content

The lecture covers the following topics:

Concurrent programming abstraction: the notion of process and thread, atomic operations and its interleaving. General correctness conditions: safety and liveness. Mutual exclusion: problem formulation and its solution through atomic read and write operations on shared memory location (Dekker's algorithm, Dijkstra's algorithm, Peterson's algorithm and Lamport's algorithm), the notion of safety and liveness. Architectural support: disabling interrupts, complex atomic operation (test-and-set, exchange). Operating system support: binary semaphores, counting semaphores, mutex locks, conditional variables. Classical synchronization problems: producer-consumer, readers-writers, dining philosophers, sleeping barber's. Language support: monitors, conditional critical regions. Deadlock: system model, resource classification, definition, necessary conditions, deadlock detection , prevention and avoidance. Processor scheduling.

Laboratory exercises are conducted in the form of fifteen two-hour classes that take place in the computer laboratory. The first laboratory session is devoted to is intended to introduce students to the principles and the evaluation of the laboratory classes. Tasks during the classes are conducted by each student individually.

The laboratory classes cover the following topics: processes and threads, threads synchronization – POSIX mechanisms. Signal handling. Interprocess communication and synchronization via links. Interprocess communication and synchronization via message queues. Interprocess communication and synchronization via semaphores and shared memory.

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), teamwork.

Bibliography

Basic

1. M. Ben-Ari, Podstawy programowania współbieżnego i rozproszonego, WNT, W-wa, 2016.
2. A. Silberschatz, G. Gagne, P.B. Galvin Podstawy systemów operacyjnych, WN PWN, W-wa, 2021.
3. M. J. Rochkind, Programowanie w systemie Unix dla zaawansowanych, WNT, Warszawa, 2007.

Additional

1. Z. Weiss, T. Gruźlewski, Programowanie współbieżne i rozproszone w przykładach i zadaniach, WNT, W-wa, 1993.

Breakdown of average student's workload

	Hours	ECTS
Total workload	100	4,0
Classes requiring direct contact with the teacher	60	2,5
Student's own work (literature studies, preparation for laboratory classes/tutorials, preparation for tests/exam, project preparation) [1]	40	1,5

[1] delete or add other activities as appropriate