COURSE DESCRIPTION CARD - SYLLABUS

Course name
Principles of Concurrent Programming [S1Bioinf1>PPW]

Course

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

Number of hours

Lecture 30	Laboratory classes 30	Other 0
Tutorials 0	Projects/seminars 0	

Number of credit points

4,00

Coordinators

dr hab. inż. Anna Kobusińska prof. PP
anna.kobusinska@put.poznan.pl

dr inż. Dariusz Wawrzyniak
dariusz.wawrzyniak@put.poznan.pl

Lecturers

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:

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 module program covers the following topics:
1.
2. Concurrent programming abstraction.
3.
4. The concepts of safety and liveness.
5.
6. The problem of mutual exclusion.
7.
8. The concept and role of the operating system (including processes, resources and threads).
9.
10. Synchronisation and communication mechanisms supported by the operating system.

11.
12. Classical synchronisation problems.
13.
14. Language support for synchronisation mechanisms.
15.
16. Deadlock problem.
17.
18. Processor scheduling.
19.

Course topics

The following topics are discussed in the lecture:
1.
2. Concurrent programming abstraction: introduction of the concept of atomic operations and their interleaving.
3.
4. The problem of mutual exclusion and its solution based on read/write atomic operations in a shared memory (e.g. Dekker, Dijkstra, Peterson, Lamport algorithms).
5.
6. Complex atomic operations, e.g.: test-and-set, exchange and their application in the implementation of mutual exclusion.
7.
8. The concept of the operating system as a resource manager (including the concepts of process and thread) and its role in the implementation of concurrent processing.
9.
10. Synchronisation mechanisms supported by the operating system: binary and counting semaphores, POSIX standard mechanisms (locks and conditional variables).
11.
12. Classical synchronisation problems: producer-consumer, readers and writers, five philosophers, sleeping barbers.
13.
14. Language support for synchronisation mechanisms: monitors and conditional critical regions.
15.
16. Deadlock problem: definition of deadlock, necessary and sufficient conditions for deadlock, counteracting deadlock (prevention, avoidance, detection and elimination).
17.
18. Processor scheduling: concept and scheduling algorithms, evaluation criteria.
19.

The laboratory classes cover the following topics at the OS service level in C:
1.
2. File access.
3.
4. Process management.
5.
6. Signal handling.
7.
8. Interprocess communication and synchronisation via pipes.
9.
10. Interprocess communication and synchronisation via IPC mechanisms.
11.

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	4,00
Classes requiring direct contact with the teacher	60	2,50
Student's own work (literature studies, preparation for laboratory classes/ tutorials, preparation for tests/exam, project preparation)	40	1,50