**Nazwa przedmiotu:**

Operating Systems

**Koordynator przedmiotu:**

dr hab. inż. Leszek J. Opalski, prof. nzw. PW

**Status przedmiotu:**

Obowiązkowy

**Poziom kształcenia:**

Studia I stopnia

**Program:**

Informatyka

**Grupa przedmiotów:**

Wspólne

**Kod przedmiotu:**

brak

**Semestr nominalny:**

4 / rok ak. 2009/2010

**Liczba punktów ECTS:**

3

**Liczba godzin pracy studenta związanych z osiągnięciem efektów uczenia się:**

**Liczba punktów ECTS na zajęciach wymagających bezpośredniego udziału nauczycieli akademickich:**

**Język prowadzenia zajęć:**

polski

**Liczba punktów ECTS, którą student uzyskuje w ramach zajęć o charakterze praktycznym:**

**Formy zajęć i ich wymiar w semestrze:**

|  |  |
| --- | --- |
| Wykład: | 30h |
| Ćwiczenia: | 0h |
| Laboratorium: | 15h |
| Projekt: | 0h |
| Lekcje komputerowe: | 0h |

**Wymagania wstępne:**

Good command of C programming language

**Limit liczby studentów:**

**Cel przedmiotu:**

\* Lecture is to teach foundation of operating systems design and implementation. \* Lab provides exposition to introductory level (mostly POSIX) kernel API programming.

**Treści kształcenia:**

Lecture \* Introduction to Operating systems: computer hardware review, basic operating system concepts, operating system structures. \* Processes and threads: models of processes and threads, signals and traps, CPU scheduling, interfaces to processes, threads and signals. \* Synchronization and communication: shared memory and message queues, critical section problem and its solutions, classical problems of synchronization. Deadlocks: model, detection, prevention, avoidance, recovery. \* Memory management: computer memory structure, contiguous memory allocation, swapping, paging, segmentation, virtual memory, memory-mapped files. \* Input/output: principles of I/O hardware operation, application I/O interface, kernel I/O subsystem, transforming I/O request to hardware operations. \* File systems: file system interface, file system implementations. \* Protection and security: security threats, principles and implementation of protection, computer-security standards. Lab Basic shell programming, processes, threads, signals, inter-process communication with: pipes (named and anonymous), shared memory and message queues, synchronization with semaphores/mutexes and conditional variables. Note: Lab activities are organized in 2-hour blocks and start in the 4th week of semester. Timetable of all lecture and lab activities (including dates of lab tutorials, as well as of lecture and lab tests) will be posted on the Web before the 3rd week of the semester.

**Metody oceny:**

\* Lecture: There are two (close notes) tests in the semester (weeks 8 and 15). A retake test will be given a week after the last lecture. Total 30p can be earned at the best. \* Lab: There is one introductory lab (ungraded), and 3 regular labs. Each regular lab has two parts. 1. Assisted problem solving. The students are assisted in solving a specially prepared programming problem. The problem is discussed and its solution planned to be done in several steps. At each stage of problem solving students can modify their own solution of the preceeding step or they can use a pre-prepared (by teachers) piece of code. The pre-prepared code is analysed by teachers and compared to students solutions. To activate student preparation for the lab - there is given a short graded quiz (0-3pts) before start of problem solving. 2. 90 min. programming test (0-7pts). The students are assigned a programming task, similar in scope to the problem which was solved in part 1. They are supposed to re-use pieces of previous solution to solve the new problem - just demonstrating understanding of concepts presented earlier. It is essential that students demonstrate their own ability to solve problems, and so any cooperation is strictly forbidden during this lab. However, students can use books, notes and Web resources to speed up implementation of their own solution. Each student can sign up for one extra (retake) test - a week before it is given (week 15 of the semester). In such a case the student is to solve a problem similar to the one given during programming tests 1, 2, or 3 (selection is up to the student). The grade of the extra test (0-7pts) replaces the grade originally earned for the selected test (0pts is assumed in case of absence during the original test). The final grade is based on the sum of points earned at the laboratory (3\*10p=30p) and during 2 class tests (30p in total) in a standard way, i.e. >90% -=> 5, >80% => 4.5 etc.

**Egzamin:**

**Literatura:**

Textbook: Silberschatz, P. Galvin, G. Gagne, Applied Operating Systems Concepts, 7th ed., J. Wiley & Sons, 2004. Auxiliary reading: \* W. Stallings, Operating Systems, 4th ed., Prentice-Hall, 2001 \* S. Tanenbaum, Modern Operating Systems, Prentice-Hall, 2001 \* W. Richard Stevens, Advanced Programming in the UNIX Environment, Addison-Wesley, 1992 \* U. Vahalia, UNIX Internals: The New Frontiers, Prentice-Hall, 1996 Lab: Free Software Foundation, The GNU C Library, available at: http://www.gnu.org/software/libc/manual

**Witryna www przedmiotu:**

**Uwagi:**

## Efekty przedmiotowe