**Nazwa przedmiotu:**

Systemy operacyjne 2

**Koordynator przedmiotu:**

Dr hab. inż. Leszek Opalski, Mgr inż. Marcin Borkowski

**Status przedmiotu:**

Obowiązkowy

**Poziom kształcenia:**

Studia I stopnia

**Program:**

Informatyka i Systemy Informacyjne

**Grupa przedmiotów:**

Wspólne

**Kod przedmiotu:**

1030-IN000-ISP-0245

**Semestr nominalny:**

4 / rok ak. 2023/2024

**Liczba punktów ECTS:**

3

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

1. godziny kontaktowe – 45 h; w tym
a) obecność na wykładach – 30 h
b) obecność na laboratoriach – 15 h
2. praca własna studenta – 45 h, w tym
a) przygotowanie do wykładów i sprawdzianów wykładowych – 15 h
b) przygotowanie do zajęć laboratoryjnych – 30 h
Razem 90 h, co odpowiada 3 pkt. ECTS

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

1. obecność na wykładach – 30 h
2. obecność na laboratoriach – 15 h
Razem 45 h, co odpowiada 2 pkt. ECTS

**Język prowadzenia zajęć:**

polski

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

1. obecność na laboratoriach – 15 h
2. przygotowanie do zajęć laboratoryjnych – 30 h
Razem 45 h, co odpowiada 2 pkt. ECTS

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

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

**Wymagania wstępne:**

Systemy operacyjne 1

**Limit liczby studentów:**

Laboratorium (ćwiczenia komputerowe) – 15-24 os. /grupa

**Cel przedmiotu:**

Celem przedmiotu jest zapoznanie studentów z zaawansowanymi problemami implementacji współczesnych systemów operacyjnych, a także kształtowanie umiejętności wykorzystania zaawansowanych mechanizmów systemowych do poprawnej realizacji (w środowisku POSIX/UNIX/Linux) aplikacji wieloprocesowych/wielowątkowych, wykorzystujących różne środki komunikacji międzyprocesowej i synchronizacji oraz komunikację sieciową. Po ukończeniu kursu studenci:
1. posiadają wiedzę na temat:
- algorytmów planowania przydziału czasu procesora/procesorów i metod ich oceny
- organizacji i efektywnego wykorzystania pamięci systemu komputerowego
- zasady działania i efektywnego wykorzystania pamięci wirtualnej
- środków komunikacji pomiędzy procesami (pamięci współdzielonej, kolejek komunikatów, łączy), podstawowych środków i schematów synchronizacji oraz problemu zakleszczeń
- komunikacji za pomocą gniazd sieciowych
- problemów bezpieczeństwa i ochrony zasobów systemów komputerowych
2. umieją:
- wyznaczać wskaźniki jakości algorytmów przydziału czasu procesora, czy średni czas dostępu do pamięci
- wykrywać zakleszczenia algorytmów przydziału zasobów za pomocą algorytmu bankiera
- zaprojektować, napisać w języku C i przetestować stworzoną przez siebie prostą aplikację sieciową w układzie klient-serwer, wykorzystując interfejs gniazd sieciowych, kolejki komunikatów lub łącza.

**Treści kształcenia:**

Wykład:
Komunikacja międzyprocesowa i synchronizacja. Komunikacja między procesami jednego systemu komputerowego: łącza nazwane (FIFO) i nienazwane (pipe), pamięć dzielona, kolejki komunikatów; synchronizacja współpracujących procesów i wątków: sposoby realizacji i interfejs programisty. Problem zakleszczania: model i metody postępowania.
Realizacja systemów plików. Organizacja wybranych systemów plików. Dostęp i ochrona plików, tryby dostępu do plików.
Podsystem wejścia/wyjścia. Urządzenia znakowe i blokowe; tryby dostępu. Moduły obsługi urządzeń wejścia/wyjścia.
Zarządzanie pamięcią. Organizacja pamięci operacyjnej i dostępu do pamięci przez procesor. Algorytmy przydziału pamięci. Modele wykorzystania pamięci przez proces. Pamięć wirtualna: budowa, własności, interfejs programisty.
Komunikacja sieciowa. Model komunikacji OSI. Interfejs gniazd w programowaniu połączeniowych i bezpołączeniowych aplikacji sieciowych (w układzie klient-serwer). Problemy komunikacji i synchronizacji aplikacji sieciowych. Przykłady usług sieciowych.
Szeregowanie zadań. Realizacja współbieżności procesów i wątków. Algorytmy szeregowania zadań (systemów jedno- i wieloprocesorowych). Specyfika szeregowania zadań w systemach czasu rzeczywistego. Ocena jakości algorytmów szeregowania.
Ochrona i bezpieczeństwo systemów komputerowych. Cele ochrony. Uwierzytelnianie, kontrola dostępu do zasobów, integralność, niezaprzeczalność, poufność - koncepcje i realizacje. Standardy oceny bezpieczeństwa.
Przegląd współczesnych systemów operacyjnych. Linux, MS Windows, QNX, FreeRTOS. Wirtualizacja systemów komputerowych.
Laboratorium:
Łącza nazwane (FIFO) i nienazwane (pipe) (3g). Komunikacja międzyprocesowa (pamięć dzielona, kolejki komunikatów) (3g). Interfejs gniazd (3g). Synchronizacja (semafory) (3g). Indywidualna poprawa jednego ćwiczenia (3g).

**Metody oceny:**

Wykład:
W semestrze odbędą się 2 sprawdziany wykładowe w terminach określonych w harmonogramie.
Studenci mają też prawo do odbycia każdego sprawdzianu w dodatkowym terminie (terminie sprawdzianu poprawkowego). Obowiązuje ten sam zakres materiału co w pierwotnym terminie. Student musi zgłosić wykładowcy zamiar przystąpienia do sprawdzianu poprawkowego na dobę przed tym terminem. Ostatnio uzyskany wynik z danego sprawdzianu liczy się do oceny. Suma punktów do zdobycia za sprawdziany wykładowe wynosi 100.
Laboratorium:
Do dyspozycji studenta są 4 ćwiczenia oceniane (L1, L2, L3, L4) oraz zajęcia (L5) poświęcone poprawie oceny z laboratorium.
- Ćwiczenia L1 i L3 mają układ: wejściówka sprawdzająca wiedzę techniczna wymaganą na zajęciach (z wykładu i man'a), tutorial ukazujący praktyczne aspekty użycia wiedzy technicznej związanej z tematyką zajęć i na koniec zadanie programistyczne ("łatwe"). Sprawdzana jest poprawność działania opracowanego programu na podstawie osiągania przez studentów z góry wyznaczonych punktów kontrolnych. Student okazuje efekt działania programu na ekranie monitora, prowadzący może zażądać pokazania również kodu. Maksymalna liczba punktów za każde z ćwiczeń: 25.
- Ćwiczenie L2 i L4 ma układ: wejściówka sprawdzająca poprawność rozumowania koncepcyjnego związanego z następującym zadaniem programistycznym ("trudnym"). W trakcie zajęć sprawdzane jest zaawansowanie wykonania zadania poprzez osiąganie poszczególnych punktów kontrolnych. Sprawdzeniu podlega nie tylko poprawność działania opracowanego programu ale i jakość kodu, który jest oceniany po zajęciach przez prowadzących. Maksymalna liczba punktów: 25.
Szczegółowy sposób wyliczenia punktów oceny będzie podany dla każdego zadania oddzielnie.
- Rozwiązania wszystkich zadań (kod) podlegających ocenie muszą zostać przekazane prowadzącym w wymaganej formie.
- Zajęcia L5 poświęcone są na poprawianie oceny. Student może wybrać jeden temat, który chciałby poprawić lub nadrobić. Forma zadania będzie zgodna z tematem, który poprawia student. Nie ma możliwości poprawiania ani nadrabiania więcej niż jednego tematu w semestrze.
- Zajęcia odbywają się (bez przerw) wg harmonogramu. Zawartość grup laboratoryjnych oraz przydział grup do terminów harmonogramu zostanie uzgodniona z reprezentacją przed pierwszymi zajęciami.
Uwagi ogólne:
- Wszystkie oceniane prace muszą być wykonywane samodzielnie. Niesamodzielność pracy, bądź korzystanie przez studenta z niedozwolonych materiałów powoduje uzyskanie z danej pracy/sprawdzianu 0p.
- W czasie wykonywania ocenianych prac pisemnych (jak wejściówki laboratoryjne) nie można korzystać z żadnych materiałów pisanych, nagrań dźwiękowych, środków komunikacji elektronicznej.
- W czasie realizacji tutoriali dozwolona (a wręcz zalecana) jest komunikacja studentów z prowadzącymi, a także pomiędzy sobą - jednak tak, by nie przeszkadzać osobom postronnym.
- W czasie rozwiązywania ocenianych programistycznych zadań laboratoryjnych można korzystać z tutoriala, własnych materiałów, rozwiązań zadań przykładowych, dokumentacji systemowej oraz Internetu jednak tak, by oceniana praca była dziełem samodzielnym.
Punkty z wszystkich ocenianych elementów sumują się do 200. Do zaliczenia wymagane jest co najmniej 100p, oceny wyższe są wyznaczane co 20p metodą proporcjonalną. Dodatkowo aby zaliczyć przedmiot konieczne jest zdobycie co najmniej 50p z części laboratoryjnej i 50p z części wykładowej.

**Egzamin:**

nie

**Literatura:**

Podstawowa:
1. A. Silberschatz, P.B. Galvin, G. Gagne, Podstawy systemów operacyjnych, wyd. 6 zm. i rozsz., WNT 2005. Uwaga: wersje angielskojęzyczne są aktualizowane corocznie.
2. The GNU C Library Manual, http://www.gnu.org/software/libc/manual/.
Uzupełniająca:
1. W. Stallings, Systemy operacyjne - architektura, funkcjonowanie i projektowanie, Wyd. 9, Helion, 2018.
2. A.S. Tanenbaum, H. Bos, Systemy operacyjne, Wyd. IV, Helion, 2015.
3. M.J. Rochkind, Programowanie w systemie Unix dla zaawansowanych (wyd. 2), WNT, 2007.
4. W.R. Stevens, Programowanie w środowisku systemu UNIX, WNT 2002.

**Witryna www przedmiotu:**

**Uwagi:**

## Charakterystyki przedmiotowe

### Profil ogólnoakademicki - wiedza

**Charakterystyka W01:**

Ma uporządkowaną, podbudowaną teoretycznie wiedzę ogólną w zakresie architektury systemów operacyjnych z rodziny UNIX oraz technologii sieciowych

Weryfikacja:

wejściówki na laboratoriach, testy audytoryjne

**Powiązane charakterystyki kierunkowe:** K\_W05

**Powiązane charakterystyki obszarowe:**

**Charakterystyka W02:**

Zna metody, techniki i narzędzia stosowane przy rozwiązywaniu prostych zadań informatycznych z zakresu budowy systemów komputerowych opartych o system Linux/Unix, sieci komputerowych i technologii sieciowych

Weryfikacja:

ocena z zadań wykonywanych podczas laboratorium, testy audytoryjne

**Powiązane charakterystyki kierunkowe:** K\_W11, K\_W13

**Powiązane charakterystyki obszarowe:**

**Charakterystyka W03:**

Zna podstawowe problemy ochrony, wie o podstawowych środkach zapewnienia bezpieczeństwa systemów komputerowych

Weryfikacja:

test audytoryjny, wejściówki na laboratoriach

**Powiązane charakterystyki kierunkowe:** K\_W14

**Powiązane charakterystyki obszarowe:**

### Profil ogólnoakademicki - umiejętności

**Charakterystyka U01:**

Potrafi pozyskiwać informacje z dokumentacji technicznej systemu (man) oraz z standardu POSIX (głównie w języku angielskim), potrafi przekształcić uzyskaną wiedzę na pisanie kodu niezależnego od platformy uniksowej

Weryfikacja:

ocena z zadań wykonywanych podczas laboratorium

**Powiązane charakterystyki kierunkowe:** K\_U05, K\_U07

**Powiązane charakterystyki obszarowe:**

**Charakterystyka U02:**

Potrafi przygotować, skompilować, sprawdzić i uruchomić program w języku C przy pomocy prostych narzędzi linii poleceń systemu

Weryfikacja:

ocena z zadań wykonywanych podczas laboratorium

**Powiązane charakterystyki kierunkowe:** K\_U11

**Powiązane charakterystyki obszarowe:**

**Charakterystyka U03:**

Ma umiejętność posługiwania się systemem Linux/Unix na poziomie API

Weryfikacja:

ocena z zadań wykonywanych podczas laboratorium

**Powiązane charakterystyki kierunkowe:** K\_U15

**Powiązane charakterystyki obszarowe:**

**Charakterystyka U04:**

Ma umiejętność pisania prostych aplikacji do komunikacji sieciowej

Weryfikacja:

ocena z zadań wykonywanych podczas laboratorium

**Powiązane charakterystyki kierunkowe:** K\_U18, K\_U30

**Powiązane charakterystyki obszarowe:**

**Charakterystyka U05:**

Potrafi rozwiązywać proste zadania z zakresu systemów operacyjnych za pomocą metod analitycznych i symulacyjnych

Weryfikacja:

wejściówki na laboratoriach, testy audytoryjne

**Powiązane charakterystyki kierunkowe:** K\_U09

**Powiązane charakterystyki obszarowe:**

### Profil ogólnoakademicki - kompetencje społeczne

**Charakterystyka K01:**

Rozumie konieczność ciągłego śledzenia zmian w dokumentacji nowych wersji bibliotek systemowego API oraz standardów takich jak POSIX

Weryfikacja:

test audytoryjny, wejściówki na laboratorium

**Powiązane charakterystyki kierunkowe:** K\_K01

**Powiązane charakterystyki obszarowe:**

**Charakterystyka K02:**

Rozumie potrzebę pisania kodu przenośnego oraz poprawnego podziału programu na biblioteki i funkcję umożliwiającego łatwe ponowne wykorzystanie kodu

Weryfikacja:

ocena z zadań wykonywanych podczas laboratorium

**Powiązane charakterystyki kierunkowe:** K\_K04, K\_K05

**Powiązane charakterystyki obszarowe:**