KARTA OPISU PRZEDMIOTU - SYLABUS

Nazwa przedmiotu

Programowanie systemowe i współbieżne

Przedmiot

Kierunek studiów

Informatyka

Rok/semestr

2 / 3

Studia w zakresie (specjalność)

-

Profil studiów

ogólnoakademicki

Poziom studiów

pierwszego stopnia

Język oferowanego przedmiotu

polski

Forma studiów

stacjonarne

Wymagalność

obligatoryjny

Liczba godzin

Wykład

30

Laboratoria

30

Inne (np. online)

0

Ćwiczenia

0

Projekty/seminaria

0

Liczba punktów ECTS

5

Wykładowcy

Odpowiedzialny za przedmiot/wykładowca:

Prof. dr hab. inż. J. Brzeziński

email: Jerzy.Brzezinski@cs.put.poznan.pl

tel. tel. (0-61) 665-2903, fax: (0-61) 877 1525,

Instytut Informatyki

ul. Piotrowo 2, 60-965 Poznań

Odpowiedzialny za przedmiot/wykładowca:

dr hab. inż. Anna Kobusińska

email: Anna.Kobusinska@cs.put.poznan.pl

tel. tel. (0-61) 665-2964, fax: (0-61) 877 1525,

Instytut Informatyki

ul. Piotrowo 2, 60-965 Poznań

Wymagania wstępne

Student rozpoczynający ten przedmiot powinien posiadać podstawową wiedzę z zakresu funkcjonowania systemów operacyjnych prezentowaną w ramach przedmiotu Systemy Operacyjne. Powinien także posiadać umiejętności: programowania, definiowania niskopoziomowych struktur danych i rozwiązywania podstawowych problemów niskopoziomowego kodowania algorytmów, nabyte w ramach przedmiotu Programowanie niskopoziomowe. Student powinien posiadać także umiejętność pozyskiwania informacji ze wskazanych źródeł, jak również rozumieć konieczność poszerzania swoich kompetencji i mieć gotowość do podjęcia współpracy w ramach zespołu.

Cel przedmiotu

Przekazanie studentom podstawowej wiedzy nt. elementów programowania współbieżnego oraz

szczegółowej wiedzy z systemów operacyjnych w zakresie zarządzania procesami, mechanizmów synchronizacji i przeciwdziałania zakleszczeniom. Rozwijanie u studentów umiejętności rozwiązywania prostych problemów programowania współbieżnego oraz stosowania wybranych mechanizmów synchronizacji do rozwiązania klasycznych problemów synchronizacji. Kształtowanie u studentów umiejętności pracy zespołowej w trakcie realizacji projektu na zajęciach laboratoryjnych.

Przedmiotowe efekty uczenia się

Wiedza

1. ma uporządkowaną i podbudowaną teoretycznie wiedzę ogólną w zakresie programowania systemowego i współbieżnego, oraz wiedzę szczegółową w zakresie zakresu funkcjonowania systemów operacyjnych
2. ma podstawową wiedzę o cyklu życia systemów operacyjnych, a w szczególności o zasadach zarządzania procesami, mechanizmach synchronizacji i przeciwdziałania zakleszczeniom
3. zna podstawowe techniki, metody oraz narzędzia wykorzystywane w procesie rozwiązywania zadań informatycznych, głównie o charakterze inżynierskim, z zakresu kluczowych zagadnień programowania systemowego i współbieżnego

Umiejętności

1. potrafi, formułując i rozwiązując zadania informatyczne, zastosować odpowiednio dobrane metody programowania systemowego i współbieżnego, w tym metody analityczne
2. potrafi ocenić złożoność obliczeniową algorytmów i problemów współbieżnych
3. potrafi - zgodnie z zadaną specyfikacją - zaprojektować (sformułować specyfikację funkcjonalną i wymagania pozafunkcjonalne dla wybranych charakterystyk jakościowych) oraz zrealizować szeroko rozumiany system informatyczny, dobierając język programowania odpowiedni do danego zadania programistycznego oraz używając właściwych metod, technik i narzędzi programowania współbieżnego
4. ma umiejętność formułowania algorytmów współbieżnych i ich implementacji z użyciem przynajmniej jednego z popularnych narzędzi

Kompetencje społeczne

1. rozumie, że w informatyce wiedza i umiejętności z zakresu programowania współbieżnego bardzo szybko stają się przestarzałe
2. ma świadomość znaczenia wiedzy z zakresu programowania systemowego i współbieżnego w rozwiązywaniu problemów inżynierskich oraz zna przykłady i rozumie przyczyny wadliwie działających systemów informatycznych, które doprowadziły do poważnych strat finansowych i społecznych

Metody weryfikacji efektów uczenia się i kryteria oceny

Efekty uczenia się przedstawione wyżej weryfikowane są w następujący sposób:

Wiedza nabyta w ramach wykładu jest weryfikowana na egzaminie pisemnym o charakterze problemowym. W trakcie egzaminu student rozwiązuje pytania problemowe lub testowe. Za każde zadanie problemowe można uzyskać 10 punktów, za zadanie testowe 1 punkt. Próg zaliczeniowy: 50% punktów.

Umiejętności nabyte w ramach zajęć laboratoryjnych weryfikowane są podstawie:
- oceny przygotowania studenta do poszczególnych sesji zajęć laboratoryjnych (sprawdzian "wejściowy")

oraz ocenę umiejętności związanych z realizacją ćwiczeń laboratoryjnych,
- ocenianie ciągłe, na każdych zajęciach (odpowiedzi ustne),
- ocenę wiedzy i umiejętności związanych z realizacją zadań laboratoryjnych poprzez kolokwia,
- ocenę wiedzy i umiejętności związanych z realizacją zadania projektowego poprzez realizację projektu w semestrze, realizowanego przez studenta jako praca domowa

Możliwe jest uzyskiwanie punktów dodatkowych za aktywność podczas zajęć, a szczególnie za omówienia dodatkowych aspektów zagadnienia.

Próg zaliczeniowy: 50% punktów.

Treści programowe

W ramach wykładu przedstawiane są następujące zagadnienia:

1) Wprowadzane są elementy programowania współbieżnego (grafy przepływu procesów, oraz notacje "and", "fork-join-quit", "parbegin-parend")

2) Wprowadzane są grafy sieci Petriego i omawiane jest zastosowanie sieci Petriego w modelowaniu procesów współbieżnych

3) Omawiany jest problem wzajemnego wykluczania oraz prezentowane i analizowane są przykładowe programowe sposoby jego rozwiązania, obejmujące m. in, algorytmy: Dekkera, Dijkstry, Petersona dla dwóch i n procesów, Lamporta

4) Definiowane są mechanizmy synchronizacji: sprzętowe (instrukcje test-and-set, aktywne czekanie, blokowanie systemu przerwań), systemowe (semafory binarne i ogólne, operacje lock i unlock, operacje enq i deq, operacja wait i post, operacje block i wakeup, liczniki zdarzeń), programowe (regiony krytyczne, warunkowe regiony krytyczne, monitory, implementacje programowych mechanizmów synchronizacji) i komunikacyjne (synchroniczne i asynchroniczne operacje wymiany komunikatów send i receive).

5) Przedstawiane są zastosowania wybranych mechanizmów synchronizacji do rozwiązywania klasycznych problemów synchronizacji (wzajemnego wykluczania, problemu producenta-konsumenta, problemu czytelników-pisarzy, problemu pięciu filozofów).

6) Omawiane jest zarządzanie procesami: pojęcie procesu, graf stanów procesów, problem szeregowania zadań w ujęciu probabilistycznym i deterministycznym (kryteria oceny uszeregowania), algorytmy szeregowania.

7) Wprowadzana jest definicja zakleszczenia, warunki konieczne i dostateczne zakleszczenia, przeciwdziałanie zakleszczeniom (podejście zapobiegania, unikania oraz detekcji i likwidacji).

Na zajęciach laboratoryjnych studenci implementują mechanizmy oferowane przez jądro systemu UNIX, ponadto konfrontują uzyskane w czasie wykładów wiadomości z praktyczną implementacją algorytmów i mechanizmów synchronizacji. W ramach laboratoriów omawiane są następujące zagadnienia:

1) Operacje na plikach zwykłych i przykłady zastosowania operacji plikowych z wykorzystaniem funkcji systemowych systemu UNIX

2) Obsługa procesów: tworzenie i usuwanie procesów, uruchamianie programów, przekierowania standardowych strumieni: wejścia, wyjścia i wyjścia diagnostycznego; przykłady użycia systemowych funkcji obsługi procesów

3) Tworzenie i obsługa łączy nazwanych i nienazwanych, przykłady błędów w synchronizacji procesów korzystających z łączy

4) Mechanizmy IPC: dostęp do pamięci współdzielonej, obsługa semaforów i kolejek komunikatów. Wykorzystanie poznanych mechanizmów do synchronizacji procesów; implementacja algorytmów poznanych na wykładzie z użyciem wybranych mechanizmów IPC

5) Obsługa i zarządzanie wątkami

Metody dydaktyczne

1. Wykład: prezentacja multimedialna, ilustrowana przykładami podawanymi na tablicy.

2. Ćwiczenia laboratoryjne: prezentacja multimedialna ilustrowana przykładami podawanymi na tablicy oraz wykonanie zadań podanych przez prowadzącego - ćwiczenia praktyczne.

Literatura

Podstawowa

1. Operating Systems: Design and Implem., Tanenbaum A., Prentice-Hall Intern. Ed., 2008
2. Podstawy systemów operacyjnych, Silberschatz A., Galvin P.B., WNT, 2006
3. Operating System Concepts, 8th, Update Edition, Abraham Silberschatz, Peter B. Galvin, Greg Gagne, Wiley&Sons, 2011
4. Program. w systemie Unix dla zaawansowanych, Marc J. Rochkind, WNT, 2008
5. System operacyjny LINUX, Cezary Sobaniec, Nakom, 2002
6. Unix i Linux. Przewodnik administratora systemów. Wydanie IV, E. Nemeth, i inni, WNT, 2011

Uzupełniająca

1. Operating Systems - A Modern Perspective, 3rd Edition , Nutt, G.J, Addison-Wesley Pub, 2003
2. Operating Systems, 3/E, Deitel I inni, Prentice Hall Intern, 2004
3. The Linux Programming Interface, Michael Kerrisk, No Starch Press, 2010
4. Advanced Programming in the Unix Environment (3rd Edition), R.Stevens, S.Rago, O'Reilly, 2013
5. Linux System Programming: Talking Directly to the Kernel and C Library, R. Love, O'Reilly, 2007

6. Linux Kernel Development, R. Love, Addison-Wesley, 2010
7. Operating Systems: Internals and Design Principles (8th Edition), Stallings W., Prentice Hall Intern, 2018

Bilans nakładu pracy przeciętnego studenta

	Godzin	ECTS
Łączny nakład pracy	120	5,0
Zajęcia wymagające bezpośredniego kontaktu z nauczycielem	64	3,0
Praca własna studenta (studia literaturowe, przygotowanie do zajęć laboratoryjnych, przygotowanie do kolokwiów/egzaminu, wykonanie projektu)[1]	56	2,0

[1] niepotrzebne skreślić lub dopisać inne czynności