KARTA OPISU PRZEDMIOTU - SYLABUS

Nazwa przedmiotu
Systemy operacyjne z programowaniem współbieżnym

Przedmiot

Kierunek studiów
Sztyczna inteligencja
Studia w zakresie (specjalność)

Poziom studiów
pierwszego stopnia
Forma studiów
stacjonarne

Rok/semestr
1/2
Profil studiów
ogólnoakademicki
Język oferowanego przedmiotu
angielski
Wymagalność
obligatoryjny

Liczba godzin

Wykład
30
Ćwiczenia

Laboratoria
30
Projekty/seminaria

Inne (np. online)

Liczba punktów ECTS

5

Wykładowcy

Odpowiedzialny za przedmiot/wykładowca:
dr inż. Dariusz Wawrzyniak

Odpowiedzialny za przedmiot/wykładowca:
dr inż. Cezary Sobaniec

Wymagania wstępne

Rozpoczynając ten moduł, student powinien posiadać podstawową wiedzę z zakresu struktury i funkcjonowania komputera, wybranych elementów matematyki dyskretnej oraz umiejętność programowania imperatywnego (szczególnie w języku C) w tym implementacji prostych algorytmów. W zakresie kompetencji społecznych uczeń powinien wykazywać się takimi postawami, jak: uczciwość, odpowiedzialność, ciekawość i kreatywność.

Cel przedmiotu

1. Zapoznanie studentów z teoretycznymi i praktycznymi problemami projektowania i implementacji systemów operacyjnych, a zwłaszcza zarządzania zasobami (np. procesorem, pamięcią, urządzeniami wejścia/wyjścia).
2. Zapoznanie studentów z obsługą systemu operacyjnego typu Unix.
3. Rozwijanie umiejętności programowania współbieżnego i systemowego, z uwzględnieniem wielozadaniowości i wielowątkowości, stosowanie mechanizmów synchronizacji oraz rozwiązywania problemu zakleszczenia.

Przedmiotowe efekty uczenia się

Wiedza

1. ma wiedzę teoretyczną z zakresu działania systemów operacyjnych,
2. ma podstawową wiedzę dotyczącą trendów w systemach operacyjnych,
3. ma ugruntowaną wiedzę na temat problemów programowania współbieżnego i zagrożeniach wynikających z niewłaściwej synchronizacji.

Umiejętności

1. potrafi tworzyć programy współbieżne — zarówno wieloprocesowe, jak i wielowątkowe — stosując mechanizmy komunikacji międzyprocesowej i synchronizacji zapewniane przez system operacyjny,
2. potrafi posługiwać się podstawowymi poleceniami uniksopodobnego systemu operacyjnego, łączyć je w potoki i tworzyć skrypty.

Kompetencje społeczne

1. rozumie, że wiedza i umiejętności związane z informatyką mogą się zdezaktualizować,
2. zna przypadki awarii systemów informatycznych, które doprowadziły do poważnych strat finansowych lub społecznych, spowodowały uszczerbek na zdrowiu, a nawet śmierć.

Metody weryfikacji efektów uczenia się i kryteria oceny

Efekty uczenia się przedstawione wyżej weryfikowane są w następujący sposób:
Wykład: Test z pytaniami wielokrotnego wyboru lub otwartymi, z możliwością uzyskania około 100 punktów, z progiem 50 punktów na zaliczenie i progiem na kolejne oceny co 10 punktów.
Zajęcia laboratoryjne: zadania praktyczne i testy.

Treści programowe

Wykład obejmuje następujące zagadnienia:
1. Definicja i funkcje systemu operacyjnego, klasyfikacja systemów operacyjnych, struktura oprogramowania systemowego i jego związek ze sprzętem, zasada działania jądra systemu.
2. System plików
a) organizacja logiczna: definicja pliku i jego atrybutów, metody dostępu do pliku, interfejs operacji na plikach, logiczna struktura katalogów.
b) organizacja fizyczna: przydział bloków dyskowych (ciągły, łańcuchowy i indeksowy), zarządzanie wolną przestrzenią (wektor bitowy, lista połączona, grupowanie, zliczanie), implementacja katalogu (lista liniowa, tablica haszująca, struktura indeksowa), implementacja operacji na plikach (buforowanie,

problem integralności, współbieżny dostęp do pliku).
3. Ogólna koncepcja zarządzania zasobami oraz pojęcie procesu i wątku.
4. Programowanie współbieżne:
a) abstrakcja programowania współbieżnego: operacje atomowe i ich przeplot,
b) ogólne warunki poprawności: bezpieczeństwo i żywotność,
c) wzajemne wykluczanie: formułowanie problemu i jego rozwiązywanie za pomocą atomowych operacji odczytu i zapisu danych w pamięci współdzielonej (algorytm Petersona i algorytm Lamporta),
d) wsparcie na poziomie architektury systemu komputerowego: wyłączanie przerwań, złożone operacje atomowe (test-and-set, exchange),
e) wsparcie na poziomie systemu operacyjnego: semafory binarne, semafory zliczające, blokady wzajemnego wykluczania (muteksy), zmienne warunkowe,
f) wsparcie w języku wysokopoziomowym: monitory, warunkowe regiony krytyczne,
f) klasyczne problemy synchronizacji: producent-konsument, czytelnicy-pisarze, pięciu filozofów, śpiący fryzjer(zy).
5. Zarządzanie zasobami:
a) zarządzanie procesorem: szeregowanie procesorów, kryteria i algorytmy szeregowania,
b) zarządzanie pamięcią: organizacja pamięci, alokacja pamięci, tworzenie obrazu procesu w pamięci, stronicowanie i segmentacja, pamięć wirtualna,
c) zarządzanie urządzeniami wejścia/wyjścia: klasyfikacja urządzeń wejścia/wyjścia, struktura mechanizmu wejścia/wyjścia, interakcja między procesorem a urządzeniami wejścia/wyjścia, buforowanie i spooling.
6. Zakleszczenie: model systemu, klasyfikacja zasobów, definicja, warunki konieczne, wykrywanie zakleszczeń, zapobieganie zakleszczeniom i unikanie zakleszczeń.

Zajęcia laboratoryjne obejmują następujące zagadnienia:
1. Wprowadzenie do obsługi uniksopodobnego systemu operacyjnego, instrukcja obsługi systemu, powłoka, edytory.
2. Użytkowanie uniksowego systemu plików: struktura katalogów, operacje na plikach, typy plików, prawa dostępu, wyszukiwanie plików.
3. Procesy: priorytety, sygnały, zarządzanie procesami współbieżnymi.
4. Komunikacja międzyprocesowa z wykorzystaniem potoków: podstawowe filtry uniksowe i złożone formy potokowe.
5. Powłoka Bourne'a: zmienne środowiskowe, przekierowania, aliasy, konstrukcje do programowania skryptów, funkcje, przetwarzanie danych wejściowych.
6. Wprowadzenie do programowania w systemach uniksowych: kompilator C, obsługa błędów.
7. Przetwarzanie zawartości plików: deskryptory plików, otwieranie plików, odczyt i zapis, implementacja prostych narzędzi uniksowych.
8. Zarządzanie procesami: tworzenie procesów, uruchamianie programów zewnętrznych, podstawowa koordynacja, przekierowywanie standardowych strumieni.
9. Komunikacja między procesami z wykorzystaniem sygnałów: obsługa sygnałów, zatrzymywanie procesów.
10. Komunikacja między procesami z wykorzystaniem potoków.

11. Komunikacja między procesami z wykorzystaniem pamięci współdzielonej.
12. Synchronizacja procesów z wykorzystaniem semaforów.
13. Programowanie wielowątkowe: tworzenie i zarządzanie wątkami.
14. Synchronizacja wątków: muteksy, zmienne warunkowe.

Metody dydaktyczne

1. Wykłady: prezentacja multimedialna (slajdy), omawianie wybranych problemów, rozwiązywanie przykładowych zadań.
2. Zajęcia: ćwiczenia praktyczne w środowisku uniksopodobnego systemu operacyjnego prowadzone w pracowni komputerowej, rozwiązywanie zadań i dyskusja.

Literatura

Podstawowa

1. Abraham Silberschatz, Greg Gagne, Peter B. Galvin: Operating System Concepts, 10th edition, John Wiley & Sons, 2018.
2. Andrew S. Tanenbaum, Herbert Bos: Modern Operating Systems, 4th edition, Prentice Hall, 2014.
3. William Stallings: Operating Systems, 9th edition, Pearson, 2018.
4. Michael Kerrisk: The Linux Programming Interface – A Linux and UNIX System Programming Handbook. No Starch Press, 2010.

Uzupełniająca

1. Gary Nutt: Operating Systems, 3rd edition, Pearson, 2004.
2. Mordechai Ben-Ari: Principles of Concurrent and Distributed Programming, Addison Wesley, 2006.
3. Arnold Robbins: Unix in a Nutshell. O'Reilly Media, 2005.

Bilans nakładu pracy przeciętnego studenta

	Godzin	ECTS
Łączny nakład pracy	110	5,0
Zajęcia wymagające bezpośredniego kontaktu z nauczycielem	60	3,0
Praca własna studenta (studia literaturowe, przygotowanie do zajęć laboratoryjnych/ćwiczeń, przygotowanie do kolokwiów/egzaminu, wykonanie projektu)[1]	50	2,0

[1] niepotrzebne skreślić lub dopisać inne czynności