KARTA OPISU PRZEDMIOTU - SYLABUS

Nazwa przedmiotu
Programowanie systemowe i współbieżne [S1Inf1>PSW]

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

Laboratorium
30

Inne (np. online)
0

Ćwiczenia
0

Projekty/seminaria
0

Liczba punktów ECTS

5,00

Koordynatorzy

dr hab. inż. Paweł Wojciechowski prof. PP
pawel.t.wojciechowski@put.poznan.pl

Wykładowcy

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 wiedzy na temat programowania współbieżnego, uwzględniając współczesne architektury komputerowe, oraz (w części laboratoryjnej) wiedzy z systemów operacyjnych w zakresie zarządzania procesami i mechanizmów synchronizacji. Rozwijanie u studentów umiejętności rozwiązywania problemów programowania współbieżnego oraz umiejętności 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 wykładów weryfikowana jest na egzaminie, który obejmuje pytania problemowe (wyżej punktowane) i testowe (niżej punktowane). Egzamin zostanie przeprowadzony w formie elektronicznej lub pisemnej. Próg zaliczeniowy: 50% punktów.

Umiejętności nabyte w ramach zajęć laboratoryjnych są weryfikowane na kilka sposobów:
- ocena przygotowania studenta do poszczególnych zajęć laboratoryjnych (sprawdzian "wejściowy"),
- ocena umiejętności związanych z realizacją ćwiczeń laboratoryjnych,
- ocenianie ciągłe na zajęciach (odpowiedzi ustne),
- ocena wiedzy i umiejętności związanych z realizacją zadań laboratoryjnych poprzez kolokwia,
- ocena wiedzy i umiejętności związanych z realizacją zadania projektowego poprzez wykonanie projektu w semestrze w ramach pracy domowej.
Możliwe jest uzyskiwanie punktów dodatkowych za aktywność podczas zajęć, a szczególnie za omówienia dodatkowych aspektów zagadnienia.
Próg zaliczeniowy: 50% punktów.

Treści programowe

Zagadnienia poruszane na wykładach:

I. Wstęp

1. Procesy i wątki.
2. Współbieżność vs równoległość.

II. Synchronizacja

1. Sekcja krytyczna i potrzeba współbieżnej koordynacji, omówione na przykładzie z życia. Podstawowe własności: wzajemne wykluczanie, wolność od zakleszczenia, wolność od zagłodzenia (lub wolność od blokady) oraz ograniczone oczekiwanie.

2. Dwa podstawowe wzorce synchronizacji: atomowość i synchronizacja warunkowa.
3. Atomowość osiągnięta przy użyciu pojedynczego zamka vs drobnoziarnistych zamków; zakleszczenie na zamkach.
4. Synchronizacja warunkowa, omówiona na przykładzie ograniczonego bufora.
5. Bariera synchronizacyjna.
6. Dwie podstawowe techniki służące do implementacji synchronizacji warunkowej i wzajemnego wykluczania (zamków): wirowanie (aktywne czekanie) i blokowanie (w module szeregowania wątków).
7. Poprawność: bezpieczeństwo (np. wzajemne wykluczanie, wolność od zakleszczenia) i żywotność (np. wolność od zagłodzenia, wolność od livelock).
8. Prawo Amdahla.

III. Architektura pamięci współdzielonej

1. Nowoczesne architektury sprzętowe z jednolitym dostępem do pamięci (ang. Uniform Memory Access, UMA) vs niejednolitym dostępem (ang. Nonuniform Memory Access, NUMA).
2. Rdzenie, pamięć podręczna (write-through vs write-back, nietrafienia w pamięci podręcznej) i warstwa komunikacyjna.
3. Lokalność czasowa i przestrzenna.
4. Systemy równoległe ze spójną pamięcią podręczną: protokoły osiągania spójności pamięci podręcznej oparte na katalogu vs wścibskie (ang. snoopy); podejścia oparte na rozgłaszaniu zapisów i unieważnianiu zapisów.
5. Spójność sekwencyjna a zrelaksowane (rozluźnione) modele pamięci.
6. Źródła niespójności: procesor, pamięć podręczna, warstwa komunikacyjna i kompilatory.
7. Instrukcje do synchronizacji w zrelaksowanej (rozluźnionej) architekturze pamięci: bariery (fences) oraz specjalne instrukcje load i store dostępu to pamięci.
8. Rozumienie współbieżnych programów wykonywanych w systemach ze zrelaksowaną (rozluźnioną) architekturą pamięci: spójność pamięci, porządek lokalny, porządek globalny, porządek programu, instrukcje omijające instrukcje, instrukcje w pełni synchronizujące vs częściowo synchronizujące (umożliwiają osłabienie porządku lokalnego dla większej wydajności).
9. TSO (Total Store Order) (np. X86) vs „bardziej zrelaksowane” maszyny (np. ARM).
10. Atomowe instrukcje maszynowe (TAS, swap, FAI, FAA, CAS i LL/SC).
11. Atomowy odczyt-modyfikacja-zapis dla dowolnej funkcji przy pomocy CAS i LL/SC.
12. Problem ABA w implementacji stosu przy użyciu listy i jego rozwiązanie za pomocą techniki wskaźników z licznikami.

IV. Zakleszczenie

1. Warunki zakleszczenia: wyłączne użycie, trzymanie (zasobów) i oczekiwanie, nieodwołalność (brak wywłaszczenia zasobów) i cykliczność.
2. Graf alokacji zasobów (ang. resource allocation graph, RAG).
3. Metody zapewnienia wolności od zakleszczenia:
 - łamanie wybranych warunków zakleszczenia,
 - zapobieganie zakleszczeniom przez konstrukcję,
 - wykrywanie zakleszczenia i powrót do stanu sprzed zakleszczenia,
 - unikanie zakleszczenia (tj. bezpieczne przydzielanie zasobów).
4. Unikania zakleszczenia bazujące na RAG.
5. Algorytm bankiera Dijkstry do unikania zakleszczenia.

V. Teoria

1. Projektowanie bezpiecznych współbieżnych obiektów.
2. Spójność sekwencyjna.
3. Liniowość (ang. linearizability) i jej formalna definicja (historie sekwencyjne, dobrze uformowane, legalne i liniowe); obiekty liniowe.
4. Porównanie spójności sekwencyjnej i liniowej na przykładach.
5. Własności liniowości: kompozycyjność (lokalność) i nieblokowalność, wraz z dowodami.
6. Zrównoleglenie posortowanej listy używając techniki ręka-w-rękę (parowania zamków) dla większej wydajności.
7. Transakcje, uszeregowalność (ang. serializability) i ścisła uszeregowalność.
8. Porównanie własności uporządkowania (spójność sekwencyjna, liniowość, uszeregowalność i ścisła

uszeregowalność).
9. Żywotność - blokowanie vs nieblokowanie.
10. Warianty postępu nieblokującego: wolność od oczekiwania (ang. wait freedom) (≡ wolność od zagłodzenia), wolność od blokady (ang. lock freedom) (≡ wolność od livelock) i wolność od przeszkody (ang. obstruction freedom).
11. Uczciwość (słaba kontra silna) na przykładach.
12. Liczba konsensusu opisująca względną „moc" prymitywów atomowych.
13. Wyścig w dostępie do danych i relacja zachodzi-przed (ang. happens-before).
14. Wstęp do modeli pamięci na poziomie języka.

VI. Zamki z aktywnym czekaniem (ang. spin locks)

1. Algorytmy historyczne do rozwiązywania wzajemnego wykluczenia (implementacji zamków):
a) algorytm Dekkera dla dwóch wątków i jego uogólnienie do n wątków,
b) algorytm Petersona dla dwóch wątków, ze szkicem dowodu,
c) algorytm „piekarniany" Lamporta.
2. Algorytmy używające TAS:
a) zamek test-and-set (TAS),
b) zamek test-and-test-and-set (TTAS),
c) zamek z wykładniczym wycofaniem.
4. Zamek z biletami Mellor-Crummey'a i Scott'a (z FAI).
5. Rozszerzony interfejs: limit czasu, trylocks.
6. Zamek z powtórzeniami (ang. reentrant lock) i jego przykładowa implementacja.
7. Problem czytelników i pisarzy (włączając zagadnienia uczciwości).
8. Zamek czytelnik-pisarz i jego przykładowa implementacja za pomocą CAS i FAA.
9. Inne warianty zamków: zamki kolejkowe i sekwencyjne.
10. Unikanie niektórych zamków przy pomocy techniki aktualizacji z kopią tylko-do-odczytu (ang. read-copy update, RCU); ogólny schemat ustawiania okresu karencji w celu bezpiecznego odzyskiwania pamięci.

VII. Szeregowanie (wątków), semafory, monitory i CCR

1. Szeregowanie wątków: multipleksowanie współbieżnych wątków (jądra systemu operacyjnego) na jednym rdzeniu (coroutines, bloki kontekstowe, lista gotowych wątków, wywłaszczenie i procedury reschedule, yield i transfer).
2. Semafory Dijkstry i ich implementacja przez blokowanie wątków (w module szeregowania wątków).
3. Semafory binarne vs. zliczające.
4. Przydział N zasobów przy użyciu semafora.
5. Problem konsumenta-producenta i bufor ograniczony.
6. Implementacja bufora ograniczonego za pomocą semaforów.
7. Monitory i zmienne warunkowe.
8. Klasyczny monitor Hoare'a i jego implementacja za pomocą semaforów.
9. Implementacja bufora ograniczonego za pomocą monitora Hoare'a.
10. Monitory w Java/C#.
11. Porównanie monitorów „klasycznych" i w Java/C#.
12. Problem zagnieżdżonego monitora.
13. Warunkowy region krytyczny (CCR).
14. Implementacja bufora ograniczonego za pomocą CCR.

VIII. Bariery

1. Przykład użycia flag i barier.
2. Scentralizowany algorytm "odwracania znaczenia" implementujący barierę.
3. Bariera bazująca na rozpowszechnianiu.
4. Rozmyte bariery poprawiające wydajność programów z barierami.
5. Rozmyty wariant bariery z "odwracaniem znaczenia".
6. Sprzętowe bariery AND oraz operacja „Eureka".
7. Wykonania równoległe w serii w Cilk (rozszerzenie C dla obliczeń równoległych).

IX. Pamięć transakcyjna

1. Pamięć transakcyjna (ang. transactional memory, TM) i jej potencjalne zalety (np. skalowalność, kompozycyjność i zapobieganie zakleszczeniom).
2. Warunkowa synchronizacja z retry i orElse.
3. Programowa pamięć transakcyjna (ang software transactional memory, STM): gwarancje postępu, buforowanie aktualizacji spekulatywnych (undo logging vs redo logging), rozwiązywanie konfliktów (chętnie kontra leniwie), śledzenie dostępu (rekordy „własności").
4. Walidacja w STM i zagadnienia poprawności (atomowość pojedynczego zamka, nieprzezroczystość).
5. Niespójny stan z powodu interakcji transakcji STM z kodem nietransakcyjnym (przykłady „publikacji" i „prywatyzacji").
6. Pamięć transakcyjna sprzętowa (ang. hardware transactional memory, HTM); rozszerzenie protokołu koherencji pamięci podręcznej MESI.
7. Zalety i wady HTM.

X. Algorytmy nieblokujące dla współbieżnych struktur danych

1. Przykład pojedynczego licznika atomowego z CAS.
2. Algorytm bezzamkowy (lock-free) Treiber'a implementujący stos (za pomocą CAS).
3. Algorytm bezzamkowy M&S Michael'a i Scott'a implementujący kolejkę (za pomocą CAS).
4. Lista H&M Harris'a i Michael'a - ogólny pomysł.
5. Uniwersalna konstrukcja (ang. universal construction) Herlihy'iego do konstruowania nieblokujących struktur danych z ich specyfikacji sekwencyjnej.

XI. Współbieżność bez danych współdzielonych

1. Współbieżny dostęp przez jawną komunikację.
2. Aktywne obiekty w języku programowania Ada, rendezvous i selektywne spotkania.
3. Bufor ograniczony za pomocą strażników (ang. guards) w Ada.
4. Przekazywanie wiadomości w systemach operacyjnych oraz w mechanizmach zdalnego wywołania procedury (RPC).
5. Przykład synchronicznego przekazywania wiadomości (z semantyką rachunku CSP) w języku programowania OCCAM.
6. Aktorzy i asynchroniczne przekazywanie komunikatów w języku programowania Erlang.
7. Agenci mobilni i asynchroniczne przekazywanie wiadomości (z semantyką rachunku pi) w języku programowania Nomadic Pict.

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

Literatura

Literatura do wykładów została wyszczególniona w materiałach dostępnych na e-kursach.

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	125	5,00
Zajęcia wymagające bezpośredniego kontaktu z nauczycielem	62	2,50
Praca własna studenta (studia literaturowe, przygotowanie do zajęć laboratoryjnych/ćwiczeń, przygotowanie do kolokwiów/egzaminu, wykonanie projektu)	63	2,50