PDF powtórzeniowy z SO

(na podstawie subiektywnie wybranych zagadnień z wykładów)

Wykład 1

1. **Cechy dobrego systeu operacyjnego:**
   - przenośność: programy użytkowe działają na różnym sprzęcie
   - wydajność: efektywne wykorzystywanie sprzętu (procesora, pamięci, we/wy)
   - zapewnienie ochrony systemu oraz procesów (przed innymi procesami, niepowołanym dostępem, witusami, etc.)
   - wygoda użytkownika
   - możliwość ewolucji systemu

2. **Jądro systemu możemy scharakteryzować jako:**
   - część systemu na stałe przebywająca w pamięci
   - wykonuje się w trybie uprzywilejowanym
   - zarządza zasobami maszyny
   - udostępnia interfejs programom użytkownika w postaci wywołań systemowych

3. **Współdzielenie procesora (time sharing) w systemach wsadowych:**
   - programista/użytkownik nie ma kontroli nad wykonywaniem zadań
   - problemy w przypadku błędu kompilacji
   - niektóre typy zadań wymagają pracy interakcyjnej
   - podział czasu procesora realizowany jako:
     - korzystanie z terminala przez wielu użytkowników
     - otrzymywanie poleceń kontrolnych przez system z terminala
     - wykonywanie na przemian programów poszczególnych użytkowników
   - procesem nazywany jest załadowany do pamięci i w niej wykonywany program
   - wymiana i pamięć wirtualna jako nowe funkcje systemu operacyjnego

4. **Systemy równoległe:**
   - architektura postaci SMP (Symmetric Multiprocessing)
   - system ze współdzieloną pamięcią
   - każdy procesor wykonuje identyczną kopię systemu operacyjnego
   - pozwala na wykonywanie wielu procesów jednocześnie

5. **Systemy czasu rzeczywistego (real-time)**
   - reakcja na zdarzenie musi się zakończyć przed upływem określonego czasu: (hard real-time) – poprawność systemu nie zależy tylko od poprawności uzyskanej odpowiedzi, ale także od czasu oczekiwania na tę odpowiedź. Przykładami są:
     - odtwarzacz DVD, symulator samolotu, steroiwnik wtrysku paliwa, system naprowadzania rakiety, etc.
   - z reguły system działą w ROM'ie, brak pamięci dyskowej
   - konflikt z zasadą podziału czasu
   - nie gwarantują nieprzekraczania czasu reakcji

6. **Systemy rozproszone:**

- luźno związany system komputerowy, gdzie każdy procesor dysponuje własną pamięcią, niedostępną innym procesorom
- koordynacja i synchronizacja poprzez wymianę komunikatów
- zaletami są: współdzielenie zasobów, przyspieszenie obliczeń, większa wiarygodność systemu, możliwość komunikacji
- sieciowe systemy operacyjne, które umożliwiają współdzielenie plików i drukarek, dostarczają możliwość komunikacji, system na jednej maszynie wykonuje się niezależnie od pozostałych maszyn w sieci
- mniejsza autonomia poszczególnych maszyn

7. Systemy hand-held:
- Personal Digital Asistans
- telefony komórkowe
- cechy charakterystyczne: ograniczona pamięć, wolne procesory (niski pobór mocy)

==========

Wykład 2

1. Praca systemu komputerowego:
- procesor i urządzenia we/wy mogą pracować współbieżnie
- każdy kontroler we/wy obsługuje jeden typ urządzeń i posiada własny bufor
- procesor przesyła dane do/z pamięci oraz do/z lokalnych buforów
- we/wy przeprowadzane jest pomiędzy lokalnym buforem kontrolera a urządzeniem
- kontroler informuje o zakończeniu operacji zgłaszając **przerwanie**
- **SYSTEM OPERACYJNY OPIERA SIĘ NA PRZERWANIACH**

2. Przerwania:
- przerwania programowe (trap – pułapka):
  - wywołanie systemu operacyjnego (np. specjany rozkaz syscall w procesorach MIPS)
  - rozkaz pułapki (brk w x86)
- sprzętowe zewnętrzne (asynchroniczne względem programu):
  - kontroler we/wy informuje procesor o zajściu zdarzenia np. zakończenie transmisji danych, nadejście pakietu z sieci, przerwanie zegara, błąd parzystości pamięci
- sprzętowe wewnętrzne, głównie niepowodzenia (fault):
  - dzielenie przez zero
  - przepełnienie stosu
  - brak strony w pamięci (w przypadu stronicowania)
  - naruszenie mechanizmów ochrony

3. Obsługa przerwania:
a) wykonana przez system operacyjny

- zapamiętanie stanu procesora (rejestrów i licznika rozkazów)
- określenie rodzaju przerwania (przepytywanie – polling, wektor przerwań – tablica adresów indeksowana numerem przerwania)
- przejście do właściwej procedury obsługi
- odtworzenie stanu procesora i powrót z przerwania

b) może dotyczyć innego procesu niż zapamiętany – przełączanie kontekstu:

- wykonuje się proces A
- przerwanie zegara → zapamiętanie stanu procesu A
- system operacyjny stwierdza, że A zużył cały przydzielony kwant czasu procesora, system postanawia przekazać sterowanie procesowi B
- odtworzenie stanu procesu B (przełączenie kontekstu) → powrót przerwania
- wykonuje się proces B

4. Synchroniczne wejście – wyjście

- powrót z systemu operacyjnego po przeprowadzeniu operacji we/wy
- proces żądający operacji we/wy jest wstrzymywany na czas jej trwania (w tym czasie procesor może zostać przydzielony innemu procesowi)

a) Metoda I: programowane we/wy (Programmed Intput-Output, PIO)

- drukowanie wiersza tekstu na drukarce
- odrębny bufor we/wy w pamięci jądra
- dwa rejestry: printer_status_reg (czy może odebrać nast. Znak) oraz printer_data_reg (bajt wysyłany do drukarki)
- problem: bezczynne oczekiwanie procesora w pętli while (ponieważ drukarka jest znacznie wolniejsza od procesora)

b) Metoda II: we/wy sterowane przerwaniami (interrupt driven)

- zakończenie transmisji każdego znaku (niekoniecznie pojedynczego) potwierdzane jest przerwaniem
- zaleta: możliwość wykorzystania procesora w czasie przeprowadzania we/wy
- wada: większy narzut związany z przesyłaniemjednego bajtu (przejście do procedury, obsługa, potwierdzenie, powrót przerwania)

c) Metoda III: bezpośredni dostęp do pamięci (Direct Memory Access, DMA):

- przesyłanie bloku danych między urządzeniem a pamięcią odbywa się bez angażowania procesora
- jest współbieżne z pracą procesora
- gdy nadejdzie czas przesłania kolejnego bajtu, urządzenie DMA przejmuje (na chwilę) od procesora kontrole nad magistralą (cycle stealing)
- następuje transmisja bajtu pomiędzy urządzeniem i pamięcią. W tym czasie procesor ma zablokowany dostęp do magistrali (nie oznacza to zawsze zatrzymania pracy procesora, jeśli posiada on pamięć podręczną)
- po przesłaniu bajtu DMA zwalnia maistralę
- przerwanie generowane jest po przesłaniu kompletnego bloku danych
- bardzo mały narzut związany z przesłaniem bajtu
- metoda wykorzystywana w sytuacji, w której szybkość urządzenia zewn. jest zbliżona do szybkości pamięci
- DMA jestkolejnym modułem podłączonym do magistrali, bądź jest zintegrowane z kontrolerem urządzenia (nowsze systemy – magistrala PCI)
- kontroler DMA(Address – gdzie przesłać dane, Count – ile, Control – [odczyt,zapis]) zajmuje się zliczaniem bajtów, podawaniem adresu i sygnałów sterujących
- set_up_DMA_controller() - zaprogramuj DMA (nr urządzenia, adres bufora w pamięci, liczba bajtów)
- DMA zajmuje się zliczaniem bajtów i oczekiwaniem na gotowość urządzenia

5. Asynchroniczne wejście - wyjście

- powrót z systemu operacyjnego po zainicjalizowaniu we/wy
- proces co jakiś czas sprawdza, czy operacja się zakończyła (w tym rozwiązaniu w czasie trwania operacji we/wy możliwe jest wykorzystanie procesora przez ten sam proces (ogólnie chodzi o to, że jak tylko procesor dostanie sygnał o rozpoczęciu operacji we/wy inicjalizuje tę operację i od razu wraca do poprzedniej instrukcji i jest gotowy na ponowne przyjęcie sygnału do operacji we/wy, nie czeka na jej zakończenie)

6. Wykorzystanie pamięci podręcznych (caching):

- wykorzystanie szybkiej pamięci do przechowywania najczęściej używanych danych (pamięć podręczna procesora, pamięć podręczna dysku)
- wymaga wprowadzenia polityki zarządzania pamięcią podręczną
- problem spójności pamięci podręcznej – informacja przechowywana w pamięci podręcznej niezgodna z informacją przechowywaną w pamięci głównej:
  - przykład: system dwuprocesorowy, gdzie każdy z nich ma własną pamięć podręczną, a zawartość jednej komórki pamięci przechowywana w obydwu pamięciach podręcznych. Procesor A zapisuje tę komórkę, a procesor B próbuje odczytu.

7. Mechanizm obrony (protection):

- potrzeba zapewnienia, że 'źle sprawujący się program' nie zakłuci pracy innych programów i samego systemu operacyjnego. Program użytkownika nie może być w stanie wykonać pewnych operacji
- przykłady: bezpośrednia komunikacja z urządzeniami we/wy → ochrona we/wy, dostęp do pamięci należącej do innych procesów lub do systemu → ochrona pamięci, zablokowanie przerwań, zmiana wektora przerwań → ochrona systemu przerwań, nieskończona pętla → ochrona procesora
- program użytkownika nie ma prawa wykonać żadnej z powyższych operacji

8. Podwójny tryb pracy:

- procesor może wykonywać instrukcje w jednym z dwóch trybów: tryb jądra (instrukcje wykonywane przez system operacyjny) lub w trybie użytkownika (instrukcje wykonywane przez program użytkownika)
- **instrukcje uprzywilejowane** – mogą być wywoływane wyłącznie w trybie jądra, próba wykonania ich w trybie użytkownika powoduje przerwanie i przejście do systemu operacyjnego
- mechanizm ochrony:
  - instrukcje uprzywilejowane: instrukcje do komunikacji z urządzeniami we/wy
  - blokowanie/odblokowanie przerwań
  - zmiana wektora przerwań (aby zapewnić, że program użytkownika nigdy nie wykona się w trybie jądra)
- przerwanie zegara:
  - gwarantuje ochronę procesora przed programem użytkownika z nieskończoną pętlą

==========

Wykład 3

1. Pojęcie procesu:

- program to plik wykonywalny na dysku i jest pojęciem statycznym, a proces to uruchomiony i wykonywany program w pamięci i ma naturę dynamiczną (zmianie ulegają np.: licznik rozkazów, rejestry procesora, wskaźnik stosu), procesor ma przestrzeń adresową (kod, dane zainicjowane i niezainicjowane, stos)
- z punktu widzenia procesora na komputerze wykonywany jest jeden program, a z punktu widzenia systemu jednocześnie jest wykonywanych wiele programów
- stany procesu:
  - nowy – utworzony
  - gotowy – czeka na przydział procesora (w tej chwili wykonuje się inny proces)
  - aktywny – wykonywane są instrukcje procesu (jeden procesor – jeden proces)
  - oczekujący (uśpiony) – proces czeka na zdarzenie (np. zakończenie we/wy), tak jak przy DMA, proces, który zainicjował transmisję DMA, lub wysłał znag do drukarki był usypiany w oczekiwaniu na przerwanie
  - zakończony – zakończył działanie
- przejścia między stanami procesu:
  - nowy → gotowy – procesor przechodzi do kolejki procesów gotowych **PLANISTA DŁUGOTERMINOWY** w systemach wsadowych
  - gotowy → aktywny – proces otrzymuje przydział procesora
  - aktywny → gotowy – procesor odebrany i przekazany innemu procesowi
  - aktywny → oczekujący – proces przechodzi w stan oczekiwania na zajście zdarzenia (np. na zakończenie transmisji we/wy)
  - oczekujący → gotowy – zdarzenie na które czekał, nastąpiło(np przerwanie)
  - aktywny → zakończony – proces zakończył pracę (np. exit, błąd ochrony)
- dodatkowy stan: zawieszony (kiedy oczekuje bardzo długo na operacje we/wy, żądanie użytkownika, brak pamięci w systemie)
- przejściami do i z stanu zawieszenia zarządza **PLANISTA ŚREDNIOTERMINOWY**
- **PLANISTA KRÓTKOTERMINOWY** – najważniejszy i występujący w każdym systemie, rozstrzyga problem '**któremu procesowi w stanie gotowym mam przydzielić procesor**'

2. Blok kontrolny procesu:

- PCB służy do przechowywania informacji o procesie istotnych z punktu widzenia systemu operacyjnego (stan procesu, id procesu, licznik rozkazów, rejestry procesora, informacja o przydzielonej pamięci, otwartych plikach, połączeniach sieciowych
- przełączenie kontekstu to zmiana procesu, a przełączanie trybu, to zmiana trybu pracy procesora (jądro ↔ użytkownik), model użytkownika zakłada, że system operacyjny jest kolekcją procedur wywołanych przez procesy. Przełączanie trybu następuje w wyniku wywołania f. Systemowej – przerwanie programowe, lub sprzętowe

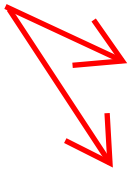
3. Utworzenie i zakończenie procesu:

- proces rodzicielski tworzy potomny, ten zaś może tworzyć kolejne potomnie ← drzewo procesów. Proces rodzicielski może współdzielić z potomnym części, lub całość zasobów, lub nic nie współdzielić. Wykonują się współbieżnie, lub rodzic czeka na dziecko. Odrębna przestrzeń adresowa dla procesów potomnych (fork) lub ta sama przestrzeń adresowa (clone).
- Proces może zakończyć się: na własne żądanie (exit lub automatycznie przy końcu maina), w wyniku akcji innego procesu (sygnał SIGKILL ← kill w shelu), przes system operacyjny (naruszenie mechanizmów ochrony, przekroczenie ograniczeń, cascading termination)

4. Wielowątkowość:

- jeden proces wykonuje się w wielu współbieżnych wątkach (thread, lekki proces - lightweight)
- każdy wątek ma swój stan, wartości rejestrów i licznika rozkazów, własny stos (**zmienne lokalne funkcji)**, ma dostęp do przestrzeni adresowej, plików i innych zasobów procesu ← **WSPÓŁDZIELONE PRZEZ WSZYSTKIE WĄTKI**

odp do pytania o serwer www



- **PROCESY SĄ OD SIEBIE ODIZOLOWANE, WĄTKI ZAŚ, NIE**
- utworzenie i zakończenie wątku zajmuje znacznie mniej czasu niż procesu, możliwość szybkiego przełączania kontekstu między wątkami tego samego procesu, komunikacja bez pośrednictwa systemu operacyjnego
- wady: 'źle zachowujący się wątek' może zakłócić pracę innych wątków tego samego procesu
- przy wątkach na poziomie użytkownika system operacyjny nie jest świadomy ich istnienia, zarządzanie jest przeprowadzane przez bibliotekę w przestrzeni użytkownika, dwa wątki nie mogą się wykonywać współbieżnie na ddwóch różnych procesorach, nie można odebrać procesora jednemu wątkowi i przekazać drugiemu
- przy wątkach na poziomie jądra wątek jest istotą systemu operacyjnego, podlegają szeregowaniu przez jądro, w SMP mogą się wykonywać na różnych procesorach (przetwarzanie równoległe)
- problemy: kiedy proces otrzymuje sygnał, otrzymują go wszystkie wątki, lub wybrany wątek, lub aktywny wątek. Kiedy proces wykonuje fork, nie wie, czy duplikować działający, czy wszystkie wątki, podobnie przy exit, czy zakończyć proces, czy aktywny wątek.
- Nie będe opisywał funkcji pthread_ to raczej każdy pamięta.

==========

Wykład 4

1. Potrzeba synchronizacji:

- aby procesy wykonywały się współbieżnie
- kiedy proces A wykonuje jakąś czynność (np. inkrementuje zmienną), a proces B wypisuje zmienną, żeby proces B nie wypisywał zmiennej przed zakończeniem inkrementowania jej przez proces A
- aby utrzymać wspólne zasoby w spójnym stanie

2. Wyścigi:

- wyścig - gdy wynik zależy od kolejności wykonywania instrukcji procesów. Poprawny program to taki, w którym nie ma wyścigów
- występuje wtedy, gdy do jednego obiektu w danej chwili odnosi się więcej, niż jeden proces
- dostęp do tego obiektu powinien się znajdować w **sekcji krytycznej**

3. Problem sekcji krytycznej:

- w założeniu w sekcji krytycznej w jednej chwili może znajdować się tylko jeden proces, który na przemian przebywa w sekcji krytycznej przez skończony czas,lub wykonuje inne czynności
- rozwiązanie: czynności wykonywane przy wejściu do sekcji - **protokół wejście**, przy wyjściu z sekcji - **protokół wyjścia**

- **kiedy proces A jest w sekcji krytycznej i proces B również chce się tam znaleźć, jest wstrzymywany do czasu, aż proces A wyjdzie z sekcji krytycznej**
- **warunki dla tego rozwiązania:** w danej chwili tylko jeden roces w sekcji (wzajemne wykluczanie). Proces, który nie jest w sekcji, nie może blokować innych procesów przed wejściem do sekcji krytycznej (postęp). Proces nie może czekać na wejście do sekcji w nieskończoność (ograniczone czekanie)
- **rozwiązania problemu:**
  - wyłączenie przerwań (nie działa w systemach wieloproc., problemy z ochroną wykorzystywane do synchronizacji w obrębie jądra
  - rozwiązanie z czekaniem aktywnym: algorytm Petersona dla 2óch procesów (wymaga 3 zmiennych), algorytm piekarni dla wielu procesów, rozwiązania wykorzystujące specjalne instrukcje maszynowe, np. rozkaz zmiany (XCHG rejestr, pamięć)

4. Czekanie aktywne:

- marnowany jest czas procesora (można go wykorzystać na wykonanie innego proc)
- uzasadnione, gdy czas oczekiwania jest stosunkowo krótki (najlepiej krótszy od czasu przełączania kontekstu) lub liczbie procesów odpowiada liczba procesorół
- alternatywą aktywnego czekania jest przejście procesu w stan zablokowany (semafory, monitory)

5. Semafor liczący

- 3 operacje – nadanie wartości początkowej ← liczba wywołań operacji wait bez wstrzymywania, Wait() i signal() ← operacje atomowe
- **definicja klasyczna na aktywnym czekaniu**:
  - wait: jeżeli Semafor S>0, to S=S-1, w przeciwnym wypadku wstrzymaj (przełącz w stan oczekujący) wykonywanie procesu ← wstrzymany przez semafor
  - signal: jeżelisą procesy wstrzymane przez semafor, to obudź jeden z nich, w przeciwnym wypadku S=S+1
  - sem: _init, _wait, _post, _trywait
- **rozwiązanie problemu sekcji krytycznej dla 1 procesu**:

```
Semaphore Sem(1); //tworzymy semafor
void Process() {
    while (1) {
        Sem.Wait();
        // Proces wykonuje swoja sekcje krytyczna
        Sem.Signal()
        // Proces wykonuje pozostale czynnosci
    }
}
```

- **rozwiązanie problemu sekcji krytycznej dla conajwyżej K procesów w sekcji** (zastosowanie semafora do zapewnienia określonej kolejności wykonywania instrukcji procesów) – chcemy, aby instrukcja A jednego procesu wykonała się po instrukcji B drugiego: używamy semafora S zainicjowanego na 0:

```
S.Wait();       |       B;
A;              |       B.Signal();
```

- **semafory binarne:**
  - zmienna może przyjmować tylko wartości 0 ← WaitB wstrzymuje proces lub 1 ← można wejść do semafora
  - WaitB, SiglanB

6. Blokada (zakleszczenie, deadlock):

- zbiór procesów jest w stanie blokady, kiedy każdy z nich czeka na zdarzeine, które może zostać spowodowane przez jakiś inny procez z tego zbioru (np. sytuacja na skrzyżowaniu, kiedy wszystkie drogi są zablokowane przez samochody i żaden nie może się cofnąć)
- Inny przykład:

```
Proces P0              Proces P1
A.Wait();              B.Wait();
B.Wait();              A.Wait();
.                      .
.                      .
B.Signal();            A.Signal();
A.Signal();            B.Signal();
```

Procesy będą czekały w nieskończoność

- **stan blokady ma miejsce ↔ gdy w grafie alokacji zasobów występuje cykl** (proces czeka na zwolnienie zasobu → proces wszedł w posiadanie zbioru i tak w kółko)
- jedną z metod uniknięcia blokady jest niedopuszczenie do powstania takiego cyklu (każdy proces wchodzi w posiadanie zasobów w określonym porządku – identycznym dla każdego procesu)

7. Zagłodzenie (starvation):

- proces czeka w nieskończoność, mimio iż zdarzenie, na które czeka występuje (reagują na nie inne procesy)
- **przykład:** jednokierunkowe przejście dla pieszych (w danej chwili może przechodzić tylko jedna osoba) – osoby czekające tworzą kolejkę, z kolejki wybierana jest zawsze najwyższa osoba, bardzo niska osoba może czekać w nieskończoność
- rozwiązanie: zamiast kolejki priorytetowej należy użyć kolejki FIFO – wybierana jest ta osoba, która zgłosiła się najwcześniej
- **przykład II:** z grupy rpocesów gotowych **planista krótkoterminowy** przydziela procesor najpierw procesom profesorów, a w dalszej części procesom studentów – jeżeli jest wiele profesorów, to w kolejce procesów gotowych znajdzie się zawsze co najmniej 1 i student będzie czekał w nieskończoność na przydział procesora

==========

Wykład 5

1. Regiony krytyczne:

- **współdzielona zmienna** v typu T jest deklarowana jako: **var** v: **shared** T; dostęp do zmiennej v jest wykonywany przy pomocy operacji: **region** v **when** B **do** S ; ← B jest wyrażeniem logicznym i tak długo jak instrukcja S się wykonuje, żaden inny proces nie może się odwołać do zmiennej v. Jeżeli wyrażenie B nie jest spełnione,

to proces jest wstrzymywany do momentu spełnienia B

- według mnie bardzo to przypomina połączenie muteksa ze zmienną warunkową: blokowanie zmiennej współdzielonej przed innymi procesami aż do spełnienia pewnego warunku zawartego w **region** v **when** B **do** S np. :

```
region x when count < n //[count < n] ← warunek
//'wlozenie' do regionu krytycznego zmiennej x
    do begin
    //ciag instrukcji modyfikujacych zmienna współdzielona x
    count:=count + 1;
    end;
```

2. Idea monitora (zmiennej warunkowej):

- udostępnianie procesom operacji pozwalającej procesowi wejść w stan uśpienia (zablokowania) ← wait(), oraz signal() ← pozwalającej na uśpienie obudzonego procesu
- wait() jest operacją atomową, która powoduje jednocześnie uśpienie procesu i wyjście z sekcji krytycznej
- tylko jeden proces/wątek może w danej chwili przebywać w procedurze monitora, gwarantuje do automatyczne wzajemne wykluczanie ← 'proces przebywa wewnątrz monitora'

3. Zmienne warunkowe (condition):

- zmienne warunkowe są rozwiązaniem problemu czekania wewnątrz monitora, aż nadejdzie zdarzenie sygnalizowane przez inny proces – jeżeli rzeczony proces po prostu zacznie czekać, nastąpi blokada, bo żaden inny proces nie będzie mógł wejść do monitora i zasygnalizować zdarzenia
- proces, który aktualnie znajduje się wewnątrz monitora może:
  - wywołać operację wait(), która zawiesza wykonanie procesu i jednocześnie zwalnia monitor innym procesom
  - wywołać operację signal(), która wznawia **jeden** proces zawieszony przez wait
  - signallAll() ← wznawia wszystkie zawieszone procesy
- aby uniknąć problemów z semantyką najlepiej przyjąć, że operacja signal() jest ostatnią operacją procedury monitora

- synchronizacji w Javie opisywać nie będę, po chyba nie ma sensu, podobnie funkcji POSIX Threads, znamy je z psów :]

4. Przekazywanie komunikatów (message pasing):

- send (odbiorca, dane);
- recive (nadawca, dane);
- na ogółwymaga kopiowania danych → spadek wydajności
- dobra metoda sychronizacji da systemów rozproszonych
- ogólnie to raczej nie ma się co nad tym rozwodzić, wydaje mi się mało prawdopodobne, żeby wśród wielu innych, ważniejszych tematów temat komunikatów się pojawił na egzaminie.

==========

Wykład 6

1. Rodzaj planowania:

- planowanie długoterminowe – decyzja o dodaniu proccesu do puli procesów wykonywanych (systemy wsadowe) ← określa stopień wieloprogramowości
- planowanie średnioterminowe – decyzja o dodaniu (usunięciu) procesu do/z puli procesów częściowo lub całkowicie obecnych w pamięci ← związane z wymianą i zarządzaniem pamięcią
- planowanie krótkoterminowe – decyzja o przyznaniu procesowi (w stanie Gotowy) procesora
- planowanie dysku – decyzja o wyborze żądania we/wy z pośr→óð żądań zgłoszonych przez procesy
- w interakcyjnych systemach z podziałem czasu planowanie długoterminowe (często również średnioterminowe) może nie występować – np. Linux

2. Kryteria planowania:

- w systemach wsadowych kryteriami są: stopień wykorzystania procesora, przepustowość – liczba procesów wykonywanych w ciągu jednostki czasu
- w systemach interakcyjnych kryterium to czas reakcji na zdarzenie (response time)
- systemy czasu rzeczywistego – zaspokojenie terminu (w razie niespełnienia terminu następuje awaria systemu) oraz przewidywalność

3. Planowanie z/bez wywłaszczenia (preemption):

- planowanie możemy wykonać, gdy proces:
  - przeszedł od stanu aktywnego do stanu oczekiwania (uśpienia) n.p. z powodu zgłoszenia zamówienia we/wy
  - proces przeszedł ze stanu aktywnego do stanu gotowego (z powodu przerwania)
  - proces przeszedł od stanu oczekiwania do stanu gotowego
  - proces zakończył pracę
- jeżeli planowania dokonujemy wyłącznie w sytuacjach 1 i 4 to mówimy o **planowaniu bez wywłaszczania.** Procesowi nigdy nie zostanie odebrany procesor, chyba że on sam się go zrzeknie
- jeżeli planowania dokonujepy dodatkowo w sytuacjach 2 i 3to mówimy o **planowaniu z wywłaszczeniem**

4. First Come First Served (FCFS):

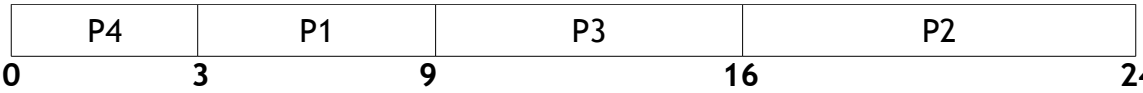
- proces, który jako pierwszy został dodany do kolejki procesów gotowych jest wykonywany jako pierwszy
- procesy otrzymują procesor na zasadzie FIFO
- **NIE WYSTĘPUJE WYWŁASZCZANIE**
- problemem jest to, że proces o dużym zapotrzebowaniu na procesor opóźnia wszystkie procesy czekające za nim w kolejce. Rozwiązanie:

5. Shorted Job First (SJF):

- procesy o najmniejszym zapotrzebowaniu na procesor (najkrótszej fazie procesora) wykonują się jako pierwsze
- występują dwie wersje: SJF bez wywłaszczania i SJF z wywłaszczaniem (Shortest Remaining Time First, SRTF)
- jest optymalny, ponieważ optymalizuje średniczas oczekiwania procesu na procesor
- **przykład I (bez wywłaszczania):**

proces:	czas trwania fazy:
P1	6
P2	8
P3	7
P4	3

używając SJF możemy zaplanować te procesy zgodnie z diagramem Gantta:

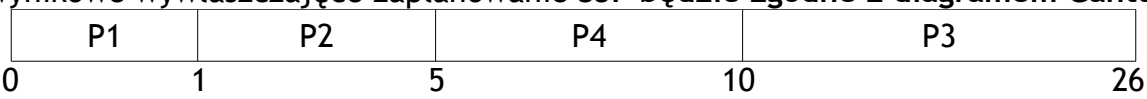


czas oczekiwania dla P1 wynosi 3ms, 16ms dla procesu P2, 9ms dla procesu P3 i 0ms dla procesu P4. Średni czas czekiwania wynosi: (3+16+9+0)/4=7ms. Gdybyśmy użyli FCFS średni czas wyniósłby 10.25 ms. SJF jest bardziej optymalny dlatego, że **umieszczenie krótkiego procesu przed długim w większym stopniu zmniejsza czas oczekiwania krótkiego procesu, niż wydłuża się czas oczekiwania długiego**, w rezultacie zmniejsza się czas średni.

- **przykład II (SJF w zywłaszczaniem):**
  - konieczność wyboru między SJF wywłaszczającym, a niewywłaszczającym powstaje wtedy, gdy w kolejce procesów gotowych pojawia się nowy proces a poprzednijeszcze używa procesora. Nowy proces może mieć krótszą następną fazę procesora niż to, co zostało do wykonania w procesie bieżącym.
  - wywłaszczający SJF usunie wtedy dotychczasowy proces z procesora, a niewywłaszczający pozwoli mu dokończyć fazę.

proces:	czas przybycia	czas trwania fazy:
P1	0	8
P2	1	4
P3	2	9
P4	3	5

wynikowe wywłaszczające zaplanowanie **SJF będzie zgodne z diagramem Gantta:**



Proces P1 startuje w chwili 0, ponieważ jest jedynym procesem w kolejce. Proces P2 nadchodzi w chwili 1,a **czas pozostały procesowi P1 (7ms) jest większy od czasu wymaganego przez proces P2 (4ms) toteż proces P1 jest wywłaszczany,** a proces P2 przejmuje procesor etc. Średni czas oczekiwania w tym przykładzie wynosi:((10-1)+(1-1)+(17-2)+(5-3))/4=6.5ms. Niewywłaszczający SJF zająby 7.75ms

6. Planowanie z wykorzystaniem priorytetów:

- każdy proces otrzymuje liczbę, zwaną priorytetem, procesy o równych priorytetach planuje się algorytmem FCFS, a proces o najwyższym priorytecie (najważniejszy) otrzymuje procesor
- im większa faza procesora, tym niższy priorytet (mniej ważny) i na odwrót
- SJF jest szczególnym przypadkiem planowania priorytetowego, gdzie długość fazy jest priorytetem
- zagłodzenie: procesy o niewielkim priorytecie mogą oczekiwać w nieskończoność (np. przy SJF wywłaszczającym jeśli wciąż będą przychodziły nowe procesy o wyższym priorytecie, proces o bardzo niskim będzie czekał na swoją kolej)

- rozwiązanie: postarzanie (aging) ← zwiększanie priorytetu procesu długo oczekującego

7. Planowanie rotacyjne (round robin):

- algorytm przypomina FCFS,tyle, że jest z wywłaszczaniem przy pomocy przerwania zegara sysetmu operacyjnego (tak jak w FCFS też kolejka FIFO)
- planista przegląda kolejkę i przydziela procesom kwant czasu procesora i ekspediuje proces do procesora (ustawia czasomierz na przerwanie po upływie przydzielonego kwantu)
- jeśli po upływie przydzielonego czasu proces nie zakończy cyklu procesora:
  - nastąpi przerwanie, proces jest wywłaszczany i dodawany na koniec kolejki FIFO procesów gotowych i kolejny proces z kolejki jest expediowany do procesora
- algorytm ten jest powszechny w systemach z podziałem czasu
- dobieranie odpowiedniego kwantu czasu:
  - typowa wartość, to 10ms
  - zbyt duży → algorytm działa jak FCFS
  - zbyt mały → straty związane z przełączeniem kontekstu (procesy nie będą mogły się zakończyć w przydzielonym kwancie czasu i będą wciąż wywłaszczane i zamieniane z kolejnym procesem)

8. Przykładowe zadanie na egzamin:

- zostanie rozwiązane na końcu, wraz z odpowiedziami na przykładowe pytania z forum

9. Planowanie z wykorzystaniem kolejek wielopoziomowych (multilevel queue):

- kolejka procesów gotowych podzielona na kilka kolejek, np.:
  - kolejka procesów pierwszoplanowych (interakcyjnych)
  - kolejka procesów drugoplanowych
- każda kolejka ma swój własny algorytm planowania, np.:
  - kolejka pierwszoplanowych ← rotacyjny
  - drugoplanowych ← FCFS
- możliwości podziału czasu procesora między kolejki (pierwszoplanowe zawsze pierwsze)
- każda kolejka ma przydzielony pewien stopień wykorzystania procesora (time slice) ← pierwszoplanowy 80%, drugoplanowy 20%

10. Wielopoziomowe kolejki ze sprzężeniem zwrotnym (multilevel feedback queue):

- proces może być przemieszczany pomiędzy kolejkami
- jeżeli zużywa za dużo czasu procesora, zostaje przemieszczony do kolejki o niższym priorytecie
- proces, który czeka bardzo długo może zostać przeniesiony do kolejki o wyższym priorytecie ← zapobieganie złagodzeniu (coś jak postarzanie w alg priorytetowym)
- specyfikacja:
  - liczba kolejek
  - alg planowania dla każdej kolejki
  - metoda do awansowania procesu do kolejki o wyższym priorytecie
  - **metoda do degradowania do kolejki o niższym priotytecie**

==========

Wykład 7

1. Hierarchia pamięci:

- rejestry procesora ← dostęp najszybszy, ale najmniejsza pojemność
- pamięć podręczna (cache – każdy chyba pamięta co to jest)
- pamięć operacyjna (DRAM) ← średnia szykość i średni koszt
- dyski ← stosunkowo tania, nieulotna, o dużej pojemniści, ale za razem o b. dużym czasie dostępu

2. Systemy wieloprogramowe – partycje o niezmieniającym się rozmiarze:

- pamięć operacyjna podzielona na obszary ostałym rozmiarze (poszczególne obszary mogą mieć różne rozmiary)
- program ładowany jest do dowolnego obszaru ← wspólna kolejka programów, oddzielne kolejki dla każdego obszaru
- określanie adresów danych i instrukcji odbywa się :
  - w chwili kompilacji (tym samym nie znamy adresu pod który program zostanie załadowany)
  - w chwili ładowania programu (generowany jest kod relokowalny, który może zostać załadowany pod dowolny adres w pamięci) ← często wykorzystuje się tablicę relokacji. Kod raz załadowany do pamięci nie może być relokowany ← adresy bezwzględne, obliczone np. względem początku kodu programu
  - w chwili wykonywania programu (kod może być przemieszczany) ← wsparcie sprzętowe np. rejestr bazowy

3. Fragmentacja:

- **wewnętrzna** ← obszar pamięci przydzielony dla procesu jest większy niż niezbędny, poważny problem przy partycjach o stałych rozmiarach
- **zewnętrzna** ← całkowita pojemność wolnej pamięci jest wystarczająca na zaspokojenie żądania procesu, ale wolna pamięć nie stanowi ciągłego bloku

4. Dynamiczne partycjonowanie:

- rozmiar partycji jest dokładnie równy rozmiarowi procesu ← brak fragmentacji wewnętrznej
- po pewnym czasie wystąpi fragmentacja zewnętrzna ('dziury w opamięci')
- kompakcja ← procesy są przesuwane w paimęci tak, aby wolna pamięć tworzyła jeden duży blok (likwiduje fragmentację, jest czasochłonna, problemy z we/wy, wymaga określania adresów w chwili wykonywania programu) ← powstaje coraz większa fragmentacja zewnętrzna
- **First-fit** ← umieszczenie programu w pierwszym bloku o wystarczającym rozmiarze
- **Best-fit** ← w największym bloku o wystarczającym rozmiarze
- **Next-fit** ← następny obszar pasujący po ostatnio wybranym
- **Worst-fit** <-w największym bloku o wystarczającym rozmiarze

5. Wymiana (swapping):

- proces może być tymczasowo przeniesiony z pamięci operacyjnej do pam. Pomocniczej (backing store)
- zwolniona pamięć może zostać wykorzystana przez inne procesy
- wymianie podlegają kompletne procesy

6. Stronicowanie:
- inne ropzwiązanie fragmentacji zewnętrznej
- pamięć fizyczna dzieli się na bloki stałej długości zwane **ramkami** (np. 4KB)
- przestrzeń adresowa procesu podzielona na strony, której rozmiar jest równy rozmiarowi ramki
- adres generowany przez procesor jest podzielony na 2 części:
  - numer strony (p) oraz przesunięcie na stronie (d)
  - strona 4KB=2^12 bajtów, adres ma 32 bity
- każdy proces ma własną tablicę stron, która jest przechowywana w pamięci
- adres fizyczny początku tablicy stron jest przechowywany w **rejestrze długości tablicy stron** ← wartości tych rejestrów są zmienne przy przełączaniu kontekstu

7. Bufor TLB (Translation Lookaside Bufer):
- przechowywany w procesorze, zawiera numer strony oraz odpowiadający mu numer ramki dla najczęściej uzywanych rąk
- jeśli numer strony występuje w buforze TLB nie ma potrzeby dostępu do tablicy stron, jeśli nie ma to para [strona,ramka] jest pobierana z tablicy stron i wprowadzana do bufora TLB ← przy przełączaniu kontekstu wymaga opróżnienie bufora TLB

8. Realizacja ochrony przy stronicowaniu:
- naruszenie sprzętowego mechanizmu ochrony (np. przez proces użytkownika) powoduje zgłoszenie wyjątku (przerwania) stronicowania
- bit ważności związany z każdą stroną i przechowywany w tablicy stron:
  - srona ważna → obecna w pamięci
  - nieważna → próba dostępu powoduje wyjątek stronicowania
- bit ochrony przed zapisem – próba zmodyfikowania strony powoduje wyjątek
- bit trybu jądra → próba dostępu z procesu użytkownika powoduje wyjątek, ochrona jądra przed programami użytkowników

9. Stronicowanie dwupoziomowe:
- tablica stron pierwszego poziomu przechowuje numery stron w tablicy stron 2 poziomu, a każdej pozycji w tablicy stron pierwszego poziomu odpowiada tablica stron drugiego poziomu
- tablica stron drugiego poziomu przechowuje numery stron
- stronicowanie eliminuje fragmentację zewnętrzną, a wewnętrzna jest ograniczona do rozmiaru strony, łatwa realizacja ochrony pamięci i dynamcznego wzrostu obszaru przydzielonej pamięci, możliwość implementacji pamięci wirtualnej

==========

Wykład 8

1. Wprowadzenie:
- system operacyjny pozwala na wykorzystanie pamięci o pojemności większej od zainstalowanej pamięci RAM
- proces 'widzi' logiczną przestrzeń adresową, od 0 do max_adress, a system operacyjny bez współpracy z nim przesyła dane do/z dysku

- przechowywanie części stron w pamięci RAM, a część na dysku
- D ← bit modyfikacji, R ← bit odwołania, V ← bit ważności
- V (Valid) dla 1 – strona w pamięci, dla 0 – strony nie ma w pamięci
- R – automatycznie ustawiany na 1, jeżeli nastąpi odwołanie
- D – ustawiany na 1, jeżeli zmodyfikowana
- R i D zerowane przez system
- system musi wiedzieć, gdzie znaleźć stronę

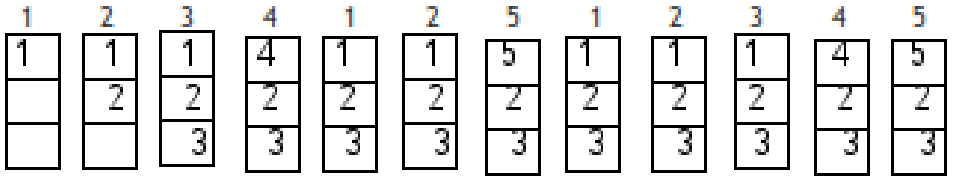
2. Obsługa błedu strony (kiedy V==0):

- przerwanie, po którym powinna być możliwość wznwienia przerwanej instrukcji
- sprawdzenie przyczyny błędu (brak strony w pamięci, błąd ochrony, błędny adres)
- w przypadku braku strony:
  - uśpienie procesu
  - znalezienie wolnej ramki
  - wczytanie strony z dysku do ramki
  - modyfikacja pozycji w tablicy stron (V=1, D=0, R=0)
  - wznowienie procesu
- w przypadku braku wolnej ramki:
  - wybranie jednej ze stron obecnych w pamięci (ofiarę) poprzez zastosowanie jednego z algorytmów zastępowania stron

3. Algorytm FIFO:

- zastąp stronę, która została zapisana do pamięci jako pierwsza:
  odwołania: 1,2,3,4,1,2,5,1,2,3,4,5

blad w tabeli



4. Algorytm drugiej szansy:

- modyfikacja FIFO, tyle że sprawdzany jest bit odniesienia R:
  - jeżeli R==0 (brak odniesienia) to strona wybierana na ofiarę
  - jeśli R==1 to R ustawiamy na 0, strona przesuwana na koniec kolejki (druga szansa i przechodzimy do kolejnej strony w kolejce
- z tego wynika, że jeśli strona będzie eksploatowna dosyć często, nigdy nie będzie zastąpiona
- kolejka cykliczna

5. Algorytm zegarowy:

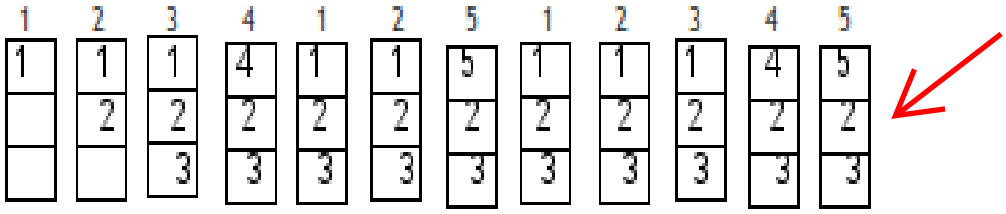
- jest innym sformuowaniem algorytmu drugiej szansy:
  - jeśli R==0, to zastąp stronę
  - jeśli R==1 to ustaw R na 0 i przesuń wskazówkę dalej

6. Algorytm optymalny:

- najniższy współczynnyk braku stron ze wszystkich algorytmów, lecz niemożliwy do zaimplementowania w praktyce
- zastępuje stronę, do której nie będziemy się odwoływać przez najdłuższy czas
  odwołania: 1,2,3,4,1,2,5,1,2,3,4,5
- (patrzymy,która będzie najwcześniej użyta: np. chcemy wstawić 4, to wstawiamy

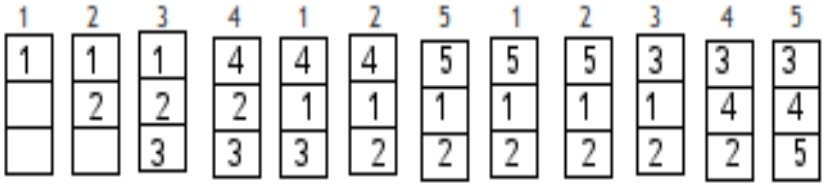
ją na miejsce 1, ponieważ 1 będzie wcześniej wstawiana, niż 2 i 3)

tu tez chyba blad



7. Algorytm LRU (Last Recently Used):

- zastąp tę stronę, która nie była najdłużej używana (zakładamy, że nie będzie dalej potrzebna)
- brak anomalii Belady'ego
  odwołania: 1,2,3,4,1,2,5,1,2,3,4,5
- (patrzymy 'wstecz', czyli która strona była ostatnio wstawiana np. jeśli chcemy wstawić 4, to wstawiamy na miejsce 1, ponieważ była wcześniej użyta, niż 2 i 3)



8. Symulowanie LRU:

- jest bardzo niewiele maszyn spełniających wymogi sprzętowe dla algorytmu LRU dlatego też w wielu systemach używa się ograniczone środki wspomagania, takie jak bit odniesienia R
- aproksymacja poprzez algorytm NFU – najrzadziej używany
- co pewien czas system przegląda tablicę stron i dla tej strony, której R==1 zwiększana jest zawartość licznika i R=0 ← w ten sposób strony z dużą ilością odwołań charakteryzują się dużą wartością licznika
- na ofiarę wybierana jest strona z największą wartością licznika

9. Stronicowanie na żądanie w systemie wieloprogramowym:

- zastępowanie stron w systemie wieloprogramowym:
  - lokalne: ofiara jest wybrana wyłącznie spośród stron procesu
  - globalne: ofiara ze stron wszystkich procesów
- przydzielanie liczby ramek proporcjonalnie do rozmiaru procesu

10. Buforowanie stron:

- zastępowanie niekoniecznie w momencie wystąpienia błędu braku strony
- odzyskane strony na liste stron zmodyfikowanych lub do bufora stron
- z list grupami na dysk i na liste niezmodyfikowanych
- w przypadku potrzeby nowej ramki przydzielana jest jedna ramka z bufora stron
- w przypadku błędu sprawdzane są najpierw obie listy

11. Szamotanie:

- zbyt mała liczba ramek ↔ liczba błędów bardzo duża
- **procesy ciągle oczekują na sprowadzenie stron z dysku i na dysk**
- małe wykorzystanie procesora

==========

Wykład 9

1. Struktura sektora dyskowego:

- Preambuła – zawiera numer sektora i cylindra
- dane
- ECC – nadmiarowe informacje pozwalające na wykrycie (prawie) wszystkich błędów i poprawienie niektórych z nich

2. Planowanie żądań do dysku:

- celem jest minimalizacja czasu spędzonego na przesuwaniu głowicy i czekaniu
- używanie algorytmów planowania do ustalania kolejności obsługi żądań:

a) FCFS (podobnie jak przy stronicowaniu:
kolejność żądań:
98, 183, 37, 122, 14, 124, 65, 67, głowica nad 53
kolejność obsługi:
53, 98, 183, 37, 122, 14, 124, 65, 67

b) SSTF (pierwsza z najkrótszym czasem oczekiwania, czyli ta, która jest najbliżej):
kolejność żądań:
98, 183, 37, 122, 14, 124, 65, 67, głowica nad 53
kolejność obsługi:
53, 65, 67, 35, 14, 98, 122, 124, 183

- **możliwość zagłodzenia**

c) SCAN (poruszanie się po dysku i realizowanie żądań napotkanych)
kolejność żądań:
98, 183, 37, 122, 14, 124, 65, 67, głowica nad 53
kolejność obsługi:
53, 37, 14, koniec - czyli 0 i powrót,65,67,98,122,124,183

d) C-SCAN (kiedy głowica dotrze do końca, natychmiast powraca na początek)
kolejność żądań:
98, 183, 37, 122, 14, 124, 65, 67, głowica nad 53
kolejność obsługi:
53,65,67,98,122,124,183,koniec i powrót do 0,14,37

e) C- LOOK (ramie przesuwa się do granicznego żądania):
kolejność żądań:
98, 183, 37, 122, 14, 124, 65, 67, głowica nad 53
kolejność obsługi:
53,65,67,98,122,124,183,przesunięcie,14,37

3. Raidy (nie będe opisywał, było na ak :])

==========

Wykład 10

1. Pliki odwzorowane w pamięci (mmap), katalogi:

- po otwarciu pliku wykonywana jest operacja mmap
- plik staje się częścią przestrzeni adresowej procesu (wygodna abstrakcja, unikamy podwójnego kopiowania danych, implementacja na ogół wykorzystuje mechanizm pamięci wirtualnej)
- problemy: plik współdzielony przez kilka procesów, próba dostępu 'za końcem' pliku
- katalog o strukturze jednopoziomowej – jeden katalog dla wszystkich użytkowników (nie pozwala na grupowania, konflikty nazw)
- katalog o strukturze dwupoziomowej – katalog główny + odrębny katalog dla każdego z użytkowników (nadal nie pozwala na grupowania plików)
- katalog o strukturze drzewa – umożliwia grupowanie, lokalizacja pliku podana przez ścieżkę (bezwzględna ścieżka – początek w korzeniu, względna – początek w katalogu aktualnym)
- katalog o strukturze grafu acyklicznego – **brak cykli** chroni przed ścieżkami o nieskończonej długości, plik może mieć 2 różne nazwy, zapewnienie acykliczności – twarde dowiązania (kolejna nazwa tego samego pliku), lub symboliczne – specjalny plik wskazujący na inny

2. Ciągła alokacja bloków danych:
- każdy plik zajmuje nioeprzerwany ciąg bloków:
  - bardzo szybki odczyt
  - bardzo proste zarządzanie informacją o blokach danego pliku, wystarczy tylko pamiętać numer pierwszego bloku i jego liczbę
  - bardzo szybka operacja seek
- **wady:**
  - może powstać fragmentacja, a kompacja w pamięci dyskowej jest bardzo wolna
  - musimy z góry znać max rozmiar pliku przy jego tworzeniu, problemy przy zwiększaniu rozmiaru pliku

3. Alokacja listowa:
- ostatnie 2,4 bajty bloku danych są zarezerwowane na numer nast. Bloku
- w katalogu przechowywany jest numer pierwszego bloku + nr ostatniego (rozrozt pliku), -1 oznacza ostatni blok

4. FAT:
- odmiana alokacji listowej, w której numery nast. Blokówprzechowywane są w odrębnej tablicy
- wolny blok: w
- ostatni blok: -1
- tablica w części lub w całości w RAM – zwiększa wydajność seek

5. Alokacja indeksowa:
- oddzielny blok indeksowy poświęcony na numery blokół z danymi
- dobra wydajność
- dla 1KB bloków i 32-bitowe nr w bloku indeksowym zmieści się 256 numerów, kiedy rozmiar plioku jest większy niż 256KB należy zastosować **indeksowanie pośrednie**

6. Indeksowanie pośrednie:

- katalog zawiera nr bloku pośredniego, który zawiera nr bloków indeksowych
- plik może mieć max długość 256*256=64MB
- potrzeba indeksowania podwójnie pośredniego (max 16GB) lub potrójnie pośredniego

7. Cache dysku:
- część pamięci RAM na przechowanie najczęściej używanych bloków w celu poprawienia wydajności
- mechanizm pozwalający szybko znaleźć blok o danym numerze, w razie braku wczytać z dysku do cache
- algorytm LRU do wybierania bloku do usunięcie z cache, na miejsce którego wstawiony będzie znaleziony na dysku blok

==========

//skoro piszecie, że nie było wykłady 11, to mniemam że nie będziemy realizowali egzaminu z materiału, który zawiera

Wykład 12

1. Tylne drzwi+bomby logiczne, wirusy robaki etc:
- tylne drzwi to mechanizm pozwalający osobie, która jest świadoma ich istnienia na uzyskanie szybkiego dostępu
- używane w celu ułatwienia debugowania systemu, zdarza się, że programista zapomni takie drzwi usunąć
- włamywacze instalują tylne drzwi aby ułatwić sobie i innym kolejne włamania
- **BOMBA LOGICZNA** – kod umieszczony w legalnym programie, który eksploduje, gdy spełnione są pewne warunki, np. programista zostanie skreślony z listy płac firmy, firma pisząc program nie otrzyma zapłaty
- **KONIE TROJAŃSKIE** – program,często aktrakcyjny, np. gra, dostępny publicznie w internecie, który wykonywuje złośliwe czynności (np. formatuje dysk,), także programy spyware
- **WIRUS** to program zdolny do powielenia przez dodanie swojego kodu do innego programu, może infekować np. sektor startowy, dokumnty etc
- **ATAKI TYPU DENIAL OF SERVICE** - złośliwe ataki wykorzystujące dziury w oprogramowaniu
- **ROBAKI** – przenoszą się używając połączenia sieciowego z jednego systemu na drugi, mogą wykonać zdalny kod dzięki możliwości przepełnienia bufora
- zasady projektowania bezpiecznego systemu:
  - budowa powszechnie znana
  - dostęp domyślnie zabroniony
  - sprawdza uprawnienia na bieżąco
  - jak najmniejsze uprawnienia procesom

==========

Wykład 13

1. Systemy wieloprocesorowe:

- z przesyłaniem komunikatów (N niezależnych procesorów, z których każdy wykonuje własny program, koordynują pracę wymieniając komunikaty przez sieć połączeń
- ze wspólną pamięcią (dowolny procesor ma dostęp do dowolnego modułu pamięci przez sieć połączeń)

a) ewolucja systemów operacyjnych dla maszyn wieloprocesorowych:

(1) każdy procesor ma swój własny system operacyjny
- pamięć na stałe podzielona na partycje, każdy OS ma jedną na wyłączność
- wady: jeżeli użytkownik zaloguje się do CPU1 wszystkie jego rpocesy wykonająsię na CPU1, jeżeli 1 procesorowi brak pamięci, to pozostałe nie mogą jej oddać

(2) model master-slave
- wydzielony (master) procesor wykonuje kod jądra i może wykonywać procesy użytkownika
- procesory slaves wykonują wyłącznie procesy użytkownika
- wady: master jest wąskim gardłem, w sytuacji gdy mamy wiele procesów zorientowanych na we/wy

(3) model SMP
- każdy procesor może wykonywać procesy użytkownika
- niezbędna poprawna synchronizacja, ponieważ w danej chwili w jądrze może przebywać wiele procesów
- potenc\jalnie prak wąskiego gardła
- wady: wymaga poprawnej synchronizacji w obrębie jądra

b) pamięć podręczna
- zastosowanie indywidualnej pamięci (typu write-back) komplikuje arch. systemu, utrudnia utrzymanie spójności danych
- system może się znaleźć w stanie niespójnym → oba procesory 'widzą' inne dane pod tym samym adresem
- do utrzymania spójności każdy procesor powinien śledzić magistralę i akcje podejmowane przez inne procesory, a próba zapisu do współdzielonego wiersza powinno powodować wysłanie sygnału do innych procesorów w celu unieważnienia tego wiersza w ich cahce

c) **fałszywe współdzielenie**
- jeśli procesor A cyklicznie odczytuje zmienną x, a procesor B cyklicznie zapisuje zmienną y formalnie wpsółdzielenia nie ma, ale pamięć podręczna operuje wierszami o typowym rozmiarze 64 bajtów. Dwie sąsiednie zmienne prawie na pewno znajdują się w tym samym wierszu, a jest on współdzielony
- aby tego uniknąć pomiędzy zmienną x a y należy wstawić odstęp równy długości wiersza pamięci podręcznej

==========

Wykład 14

1. Systemy rozproszone:
- system składający się z wielu maszyn połączaonych siecią
- współdzielenie zasobów: np. jednej drukarki, przetwarzanie informacji przez serwer baz danych
- przyspieszenie obliczeń – współdzielenie obciążenia
- zwiększenie niezawodności – w razie awarii inne maszyny mogą przejąć funkcje zepsutej maszyny

- migracja procesów do maszyn nieobciążonych
- migracja danych
- trudna synchronizacja
- wyposażone w centralny zegar i współdzieloną pamięć
- synchronizacja – przesyłanie komunikatów
- proces koordynatora – zmniejsza wydajność, wąskie gardło

2. Klaster:
- zbiór niezależnych komputerów, połączonych siecią sprawiających wrażenie jednej maszyny
- dzielimy na: obliczeniowe, zapewniające redundację (w przypadku awarii jednego z elementów zadanie przejmowane przez pozostałe elementy)

3. Hard real-time
- poprawność zależy nie tylko od sprawności, ale też od dostarczania wyników przed upływem nieprzekraczalnego terminu (**deadline**)
- przykłady: komputer sterujący rakietą,odtwarzacz DVD etc

4. Soft real-time:
- większość systemów PC
- nie gwarantują dotrzymania terminów
- procesy czasu rzeczywistego – otrzymują procesor zawsze przed zwykłymi procesami

5. Planowanie w systemach czau rzeczywistego:
- proces: posiada okrez p – czas między kolejnymi zdarzeniami obsługi procesora, termin d – czas, w którym zdarzenie musi być obsłużone, czas t potrzebny procesorowi na obsługę tego zdarzenia
- stopień wykorzystania procesora: u=t/p
- proces przekazuje swoje parametry planiście, a on albo podejmuje się wykonania zadania gwarantując dotrzymania terminu albo odrzuca proces

6. Algorytm EDF - najwcześniejszy deadline najpierw:
- priorytet przypisany dynamicznie
- przyznany doa procesu o krótszym terminie realizacji

7. SCAN - EDF:
- algorytm sortuje żądania w porządku EDF, a żądania o takim samym terminie obsługuje w porządku SCAN
- grupowanie żądań dla kilku żądań niewiele różniących się w terminach
  przykład:

  Głowica jest nad cylindrem 50, porusza się w stronę cylindra 51, długość kwantu 100 ms. Grupujemy żądania w 4 grupy:
  - 0-99ms (D,F);
  - 100-199ms (A,G,H);
  - 200-299ms (B,E,J);
  - 300-399ms (C,I).

  Wewnątrz każdej grupy obsługa według algorytmu SCAN.
  Finalny porządek:

  (F, zmiana kierunku,D)

  (A, zmiana kierunku, H, G)

(E, zmiana kierunku B,J)

(I, zmiana kierunku C)

==========

Przykładowe zadanie na egzamin

Proces P1: 10ms CPU, 20 ms I/O, 20 ms CPU, 10 ms I/O

- **Proces P2: 40 ms I/O, 20 ms CPU.**
- **Proces P3: 50 ms CPU, 10 ms I/O, 20 ms CPU.**
- **Narysuj diagramy Gannt'a obrazujące planowanie procesora przy pomocy**
- **algorytmów: FCFS, SJF, SJF z wywłaszczaniem, i rotacyjnego z kwantem czasu 10ms.**

poniżej znajdują się updejtowane odpowiedzi :]

Odpowiedzi do

przykładowych zadań na egzamin

(uzupełniłem i zmieniłem tam, gdzie uznałem za stosowne, podziękowania dla Andruszy)

1. Różnice między systemami czasu rzeczywistego: soft real-time i hard real-time.

hard real-time - porawność zależy nie tylko od sprawności, ale też od dostarczania wyników przed upływem nieprzekraczalnego terminu (**deadline**)
przykłady: komputer sterujący rakietą,odtwarzacz DVD etc

soft real-time - większość systemów PC, nie gwarantują dotrzymania terminów, procesy czasu rzeczywistego – otrzymują procesor zawsze przed zwykłymi procesami

2. Różnice między synchroniczną, a asynchroniczną operacją wejścia/wyjścia.

synchroniczne wykonywanie operacji wejścia/wyjścia - procesor zleca wykonanie operacji i czeka, aż otrzyma potwierdzenie zakończenia operacji. Proces żądający operacji we/wy jest wstrzymywany na czas jej trwania.

- Metoda I: programowane we/wy (Programmed Intput-Output, PIO)
- Metoda II: we/wy sterowane przerwaniami (interrupt driven)
- Metoda III: bezpośredni dostęp do pamięci (Direct Memory Access, DMA)

asynchroniczne wykonywanie operacji wejścia/wyjścia - procesor zleca wykonanie operacji, przerywa wykonywanie aktualnie wykonywanego zadania i nie czekając na zakończenie się operacji wejścia/wyjścia zaczyna wykonywać inne zadanie; po zakończeniu operacji wejścia/wyjścia urządzenie zewnętrzne generuje przerwanie, a procedura obsługi przerwania odnotowuje ten fakt. Proces co jakiś czas sprawdza, czy operacja się zakończyła (w tym rozwiązaniu w czasie trwania operacji we/wy możliwe jest wykorzystanie procesora przez ten sam proces)

3. **Będą podane 3 urządzenia, np.: A- mysz B- klawiatura C- szybki dysk twardy i będzie trzeba wybrać urządzenie/a, które musi być obsługiwane metodą transmisji DMA.**

DMA używa się gdy szybkość urządzenia zewnętrznego jest zbliżona do szybkości pamięci

1. (pytanie, które zawsze pojawia się na egzaminie!) Czym zajmuje się planista krótkoterminowy, długoterminowy i średnioterminowy?

Planista krótkoterminowy - odpowiada za wybór procesów spośród gotowych do wykonania. Musi być on bardzo szybki, w przeciwieństwie do ***planisty długoterminowego***, który wybiera procesy z pamięci masowej i ładuje do pamięci operacyjnej.

Planista średnioterminowy - (ang. medium-term scheduler) zajmuje się wymianą procesów pomiędzy pamięcią główną, a pamięcią zewnętrzną (np. dyskiem)

1. Podaj definicję blokady (zakleszczenia).
Zakleszczenie - (*deadlock*) jest pojęciem opisującym sytuację, w której co najmniej dwie różne akcje czekają na siebie nawzajem, więc żadna nie może się zakończyć. Zbiór procesów jest w stanie blokady, kiedy każdy z nich czeka na zdarzeine, które może zostać spowodowane przez jakiś inny procez z tego zbioru (np. sytuacja na skrzyżowaniu, kiedy wszystkie drogi są zablokowane przez samochody i żaden nie może się cofnąć)

2. Podaj definicję zagłodzenia.
Zagłodzenie (starvation) - proces nie jest w stanie zakończyć działania, ponieważ nie ma dostępu do procesora lub innego współdzielonego zasobu. Proces czeka w nieskończoność, mimio iż zdarzenie, na które czeka występuje (reagują na nie inne procesy)
przykład: jednokierunkowe przejście dla pieszych (w danej chwili może przechodzić tylko jedna osoba) – osoby czekające tworzą kolejkę, z kolejki wybierana jest zawsze najwyższa osoba, bardzo niska osoba może czekać w nieskończoność

3. Jaka jest wada metody synchronizacji opartych na aktywnym czekaniu?
W architekturach jednoprocesorowych czekanie aktywne jest nazbyt kosztowne i na ogół stosuje się zamiast niego usypianie procesu. Marnowany jest czas procesora (można go wykorzystać na wykonanie innego proc) - uzasadnione, gdy czas oczekiwania jest stosunkowo krótki (najlepiej krótszy od czasu przełączania kontekstu) lub liczbie procesów odpowiada liczba procesorów. Alternatywą aktywnego czekania jest przejście procesu w stan zablokowany (semafory, monitory)

4. Podaj 2 z 3 warunków, które powinno spełniać rozwiązanie problemu sekcji krytycznej.

1. **Wzajemne wykluczanie** (***mutual exclusion***). Jeśli proces P_i wykonuje sekcję krytyczną, żaden inny proces nie może wykonywać sekcji krytycznych.
2. **Postęp**. Jeśli żaden proces nie działa w sekcji krytycznej oraz istnieją procesy, które chcą wejść do sekcji krytycznych, to tylko procesy nie wykonujące swoich reszt mogą kandydować do wejścia; wybór nie może być odwlekany w nieskończoność.
3. **Ograniczone czekanie**. Musi istnieć wartość graniczna liczby wejść innych procesów do ich sekcji krytycznych, po tym gdy dany proces zgłosił chęć do sekcji krytycznej i zanim dostał pozwolenie.

5. Podaj rozwiązanie problemu sekcji krytycznej wykorzystujące semafor.

Wystarczy dla każdej sekcji krytycznej utworzyć odrębny semafor, nadać mu na początku wartość 1, w sekcji wejściowej każdy proces wykonuje na semaforze operację P, a w sekcji wyjściowej operację V. Wymaga to jednak od programisty pewnej dyscypliny: Każdej operacji P musi odpowiadać operacja V i to na tym samym semaforze. Pomyłka może spowodować złe działanie sekcji krytycznej.

6. Kiedy mówimy o wyścigu?
wyścig – gdy wynik zależy od kolejności wykonywania instrukcji procesów. Poprawny program to taki, w którym nie ma wyścigów, występuje wtedy, gdy do jednego obiektu w danej chwili odnosi się więcej, niż jeden proces. Dostęp do tego obiektu powinien się znajdować w **sekcji krytycznej**

7. Podaj 2 wady rozwiązania problemu sekcji krytycznej przy użyciu wyłączenia przerwań.
1) przełączanie wszystkich procesów jest zablokowane
2) system nie reaguje na zewnętrzne zdarzenia, co może spowodować utratę danych

---

2. Jaka jest różnica między semantyką Hoare'a, a semantyką Mesa?
Semantyka Mesa
– Proces który wywołał operacje signal (Q) kontynuuje pierwszy.
– P może wznowić działanie, gdy Q opuści monitor.
– Wydaje się być zgodna z logiką, po co wstrzymywać proces który zgłosił zdarzenie.
Semantyka Hoare'a
– Proces odblokowany (P) kontynuuje jako pierwszy.
– Może ułatwiać pisanie poprawnych programów. W przypadku semantyki Mesa nie mamy gwarancji, że warunek, na jaki czekał P jest nadal spełniony (P powinien raz jeszcze sprawdzić warunek).

3. Jak działają operacje signal i wait monitora?
wait - powoduje wstrzymanie procesu i wstawienie go na koniec kolejki,
signal - powoduje wznowienie pierwszego procesu z kolejki, o ile kolejka nie jest pusta.

---

1. Diagramy Grant'a
diagramy realizujące i obrazujące algorytmy planowania procesów – patrz wykład 6

2. Co to jest wywłaszczenie?
Wywłaszczenie - to technika używana w środowiskach wielowątkowych, w której algorytm szeregujący (scheduler) może wstrzymać aktualnie wykonywane zadanie (np. proces lub wątek), aby umożliwić działanie innemu. Dzięki temu rozwiązaniu zawieszenie jednego procesu nie powoduje blokady całego systemu operacyjnego

//według mnie to po prostu zakończenie aktualnie wykonywanego procesu, tak jakby przerwanie mu, przed jego zxakończeniem na rzecz innego procesu. Stosowane w algorytmie SJF z wywłaszczaniem kiedyto algorytm o krótszym czasie pracy wywłaszczał aktualnie wykonywujący się proces o dłuższym czasie planowanej pracy :]

4. W PRL stosowano prawo na podstawie którego kobieta w ciąży miała bezwzględne pierwszeństwo nad pozostałymi osobami czekającymi w kolejce do lady w sklepie. Wykaż, że takie rozwiązanie mogło prowadzić do zagłodzenia zwykłych klientów i zaproponuj inne rozwiązanie, które preferowałoby kobiety w ciąży i jednocześnie nie prowadziło do zagłodzenia

Rozwiązaniem tego problemu może być, na przykład, automatyczne podwyższanie priorytetu procesu w miarę jego starzenia (upływu czasu oczekiwania). Wówczas żaden proces nie czekałby w nieskończoność, bo miałby wówczas nieskończony priorytet, co jest niemożliwe.

---

1. Fragmentacja wewnętrzna.
straty pamięci powodowane nieużytkami występującymi w ostatnich blokach plików lub w końcowych stronach programu
Zmniejszanie fragmentacji polega na stosowaniu mniejszych bloków, co jednak wydłuża czas przesyłania informacji między pamięcią operacyjną a pamięcią dyskową.

Fragmentacja:

- **wewnętrzna** ← obszar pamięci przydzielony dla procesu jest większy niż niezbędny, poważny problem przy partycjach o stałych rozmiarach
- **zewnętrzna** ← całkowita pojemność wolnej pamięci jest wystarczająca na zaspokojenie żądania procesu, ale wolna pamięć nie stanowi ciągłego bloku

2. Podaj przyczynę dla której przełączanie między procesami jest wolniejsze niż te między wątkami tego samego procesu?
Odp: Podczas przełączania zwalnia się bufor TLB, który zapamiętuje tablicę najczęściej używanych stron. <--true

---

1. Ciąg odwołań: umieć zasymulować algorytmy z wykładu: FIFO (First In First Out), Algorytm drugiej szansy, Algorytm optymalny, Algorytm LRU (Least recently used).
W pdfi'e bardzo ładnie opisałem, namęczyłem się nawet z obrazkami :]

---
1. Będzie podany ciąg numerów sektorów dysku i za pomocą algorytmów poznanych na wykładzie będzie trzeba je odczytać w odpowiedniej kolejności. (algorytmy: FCFS, SSTF, SCAN, C-SCAN, C-LOOK)
//algorytmy również opisałem w pdf'ie przy okazji wykładu 9
Strategia FCFS (ang. *first-come first-served*) to najprostsza z możliwych strategii. Polega ona na wykonywaniu odwołań do dysku w takiej kolejności, w jakiej się one pojawiają.

Strategia SSTF (ang. *shortest seek time first*) polega na tym, że w pierwszej kolejności obsługiwane jest to odwołanie do dysku, które jest najbliżej aktualnej pozycji głowicy.
W strategii scan głowice poruszają się wahadłowo od skrajnie zewnętrznego do skrajnie wewnętrznego cylindra i z powrotem. Po drodze zatrzymują się wykonując odwołania do napotykanych cylindrów. Poniższy rysunek przedstawia przebieg głowicy dla przykładowych odwołań.

Strategia c-scan jest modyfikacją strategii scan. Głowice przesuwają się w jednym kierunku po dysku realizując odwołania do mijanych cylindrów. Gdy osiągną skrajny cylinder, to natychmiast przesuwają się na drugi koniec dysku, nie realizując po drodze żadnych odwołań, po czym cały cykl jest powtarzany. Strategia ta jest nieco wolniejsza niż scan, ale za to średni czas oczekiwania na wykonanie odwołania do dysku jest taki sam dla wszystkich cylindrów.

Strategia c-look stanowi modyfikację strategii look. Podobnie, jak w przypadku c-scan, odwołania do dysku są realizowane tylko gdy głowice przesuwają się w jedną stronę po dysku. Po zrealizowaniu skrajnego odwołania głowice zawracają i przesuwają się do

skrajnego odwołania po drugiej stronie dysku. Strategia ta jest trochę wolniejsza niż strategia look, ale za to średni czas oczekiwania na realizację odwołania jest taki sam dla wszystkich cylindrów.
---

1. Wiadomo, że wzrost długości rozmiaru bloków zwiększa wydajność. Dlaczego więc nie ma bloków o długości 128 MB ?
Ponieważ stosując:
a) alokację indeksową możemy uzyskać 256KB
b) indeksowanie pośrednie – 256KB*256KB = 64MB
c) indeksowanie pośrednie podwójne – 256KB*256KB*256KB = 16GB

2. Która metoda alokacji danych charakteryzuje się największą/najmniejszą wydajnością operacji seek?
//nie wiem :d
---
1. Co to jest fałszywe współdzielenie?
jeśli procesor A cyklicznie odczytuje zmienną x, a procesor B cyklicznie zapisuje zmienną y formalnie wpsółdzielenia nie ma, ale pamięć podręczna operuje wierszami o typowym rozmiarze 64 bajtów. Dwie sąsiednie zmienne prawie na pewno znajdują się w tym samym wierszu, a jest on współdzielony. Aby tego uniknąć pomiędzy zmienną x a y należy wstawić odstęp równy długości wiersza pamięci podręcznej

2.W którym systemie występuje planista długoterminowy?
W systemach wsadowych
---
1. Dwa algorytmy z wykładu:
a) Algorytm EDF – najwcześniejszy deadline najpierw:
- priorytet przypisany dynamicznie, przyznany dla procesu o krótszym terminie realizacji

b) SCAN – EDF:
- algorytm sortuje żądania w porządku EDF, a żądania o takim samym terminie obsługuje w porządku SCAN
- grupowanie żądań dla kilku żądań niewiele różniących się w terminach
  przykład:
  Głowica jest nad cylindrem 50, porusza się w stronę cylindra 51, długość kwantu 100 ms. Grupujemy żądania w 4 grupy:
  - 0-99ms (D,F);
  - 100-199ms (A,G,H);
  - 200-299ms (B,E,J);
  - 300-399ms (C,I).

  Wewnątrz każdej grupy obsługa według algorytmu SCAN.
  Finalny porządek:
  (F, zmiana kierunku,D)
  (A, zmiana kierunku, H, G)
  (E, zmiana kierunku B,J)
  (I, zmiana kierunku C)

życzę powodzenia i mam nadzieję, że się przyda ;)

~Greeg