Systemy operacyjne wykład

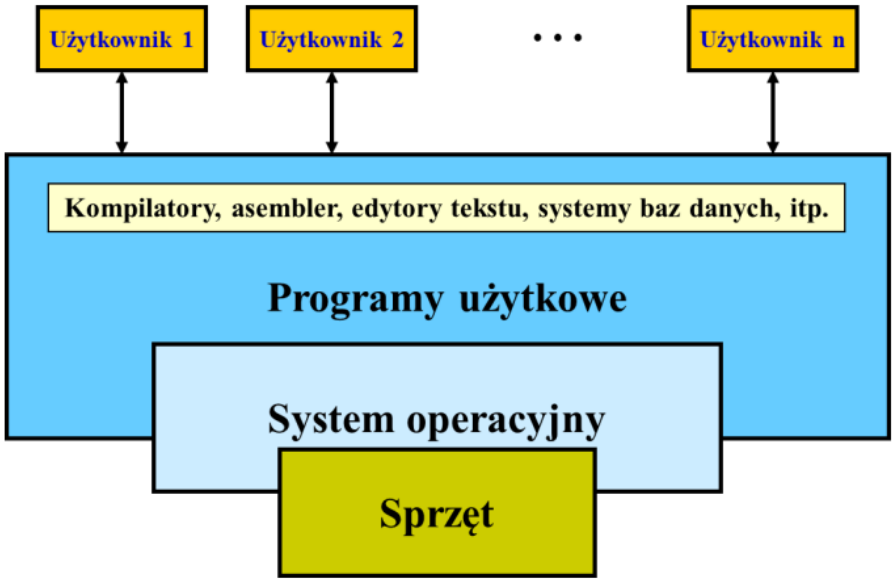
Zagadnienia do kolokwium

1. **System operacyjny** – definicja + zadania

Jest to program komputerowy:
- steruje zasobami systemu komputerowego (sprzęt + oprogramowanie) – dystrybutor zasobów, najbliżej powiązany ze sprzętem (zasoby to np. pamięć, czas CPU, we/wy, porty komunikacyjne; zarządzanie pamięci, procesorem, plikami itd.)
- jest pośrednikiem między użytkownikiem a systemem komputerowym
- zapewnia użytkownikowi interfejs (ukrywa szczegóły sprzętowe systemu komputerowego przez tworzenie maszyn wirtualnych, gdzie zbiory bloków dyskowych widziane są jako pliki o symbolicznych nazwach, pojawia się abstrakcja równoległości (współbieżne wykonanie programów) oraz sposób dostępu do urządzeń zewnętrznych jest jednolity

2. **Schemat blokowy systemu komputerowego**

Jądro systemu operacyjnego – podstawowa część systemu operacyjnego, która jest odpowiedzialna za wszystkie jego zadania.



3. **Systemy wsadowe**

Dla większej efektywności użycia komputera zatrudniano operatora, który z kart peforowanych przekazanych mu przez programistów przygotowywał wsady zadań o podobnych wymaganiach.
Rezydentny monitor – prosty program, który automatycznie przekazywał sterowanie od jednego zadania do drugiego – możliwość wykonywania jednego zadania na raz. Usprawnił on pracę operatora. Koncepcja AJS – automatyczne porządkowanie zadań.

Karta sterująca – Job Control Language – język opisu zadań.

Karta perforowana – karta, na której każda pozycja reprezentuje pojedynczy bit informacji; na kartach perforowanych programiści drukowali kod źródłowy programu.

W skład programu Monitor wchodziły: program ładujący, AJS, karty sterujące.

Praca pośrednia – sposób obsługi czytników kart i drukarek; wprowadzenie specjalnych urządzeń (przewijaki taśm), które przepisywały zawartość kart perforowanych na taśmę magnetyczną lub dane z taśmy magnetycznej drukowały na drukarce.

Buforowanie wejścia-wyjścia – proces, który ma zrównoważyć obciążenie procesora i urządzeń wejścia wyjścia. Bufor – miejsce w pamięci operacyjnej komputera, gdzie urządzenia wejściowe umieszczały dane potrzebne dalej programowi (lub urządzenia wyjściowe wyniki jego pracy). Zapełnianie buforów wejściowych i opróżnianie wyjściowych nadzorował system operacyjny.
Efektywność buforowania może być ograniczona z powodu uzależnienia zadań albo od procesora, albo od urządzeń wejścia wyjścia, co wpływa na proporcje operacji w obu odmianach.

Spooling – ulepszenie buforowania – dysk stał się buforem, gdyż pojawiły się pamięci dyskowe. Możliwość gromadzenia danych w różnych miejscach dysku. System operacyjny stał się odpowiedzialny za obsługę pamięci dyskowej, nadzorowanie spoolingu i planowanie kolejności wykonania zadań.

4. Wieloprogramowość i wielozadaniowość

Wieloprogramowość – pojawienie się możliwości równoczesnego utrzymania do kilkudziesięciu procesów użytkownika. Procesor mógł je wykonywać w dowolnej kolejności. Komputer wykonuje zadanie umieszczone w pamięci; jeśli zadanie musi oczekiwać na jakiś zasób, wówczas procesor zawiesza ten proces i przechodzi do wykonania innego procesu.

Wielozadaniowość – przełączanie między aktywnymi zadaniami.
Podział na wielozadaniowość ze stałym podziałem czasu, oraz wielozadaniowość reagującą na zdarzenia.

5. Budowa przykładowych systemów operacyjnych

a) System THE
Jest to system dla indywidualnego użytkownika. Ma strukturę warstwową.

b) System DOS
MS-DOS to system dla indywidualnego użytkownika. Jest to system monolityczny. Maksymalna funkcjonalność przy oszczędności miejsca; brak wyraźnego podziału na moduły; interfejsy i poziomy funkcjonalności nie są jasno wydzielone; brak dualnego trybu pracy.

Dualny tryb pracy – tryb użytkownika i tryb systemowy.

c) System Unix
System dla indywidualnego użytkownika z możliwością wielodostępu. Składa się z dwóch odrębnych części: programy systemowe, jądro systemu. Jądro to wszystko co znajduje się poniżej interfejsu funkcji systemowych, a powyżej sprzętu. Udostępnia ono system plików, planowanie przydziału procesora, zarządzanie pamięcią i inne funkcje systemu.

Działanie struktury warstwowej oraz mikrojądra opisane jest w punkcie 9.

6. Usługi systemu operacyjnego

1. Realizacja programu
2. Obsługa operacji we/wy
3. Manipulowanie systemem plików
4. Komunikacja między procesami współbieżnymi
5. Wykrywanie i obsługa błędów
6. Gospodarka zasobami systemu
7. Ochrona zasobów systemu komputerowego
8. Ewidencja wykorzystania zasobów systemu komputerowego i rozliczanie użytkowników.

7. Funkcje systemu operacyjnego

Funkcje systemowe to interfejs między wykonywanym programem a systemem operacyjnym. Są dołączane do kompilowanego programu w procesie kompilacji i wywoływane w celu wykonania przez system operacyjny określonych zadań.

Funkcje nadzorowania procesów
- End (zakończenie)
- Abort (zaniechanie)
- Load (załadowanie)
- Execute (wykonanie)
- Wait for Event (oczekiwanie na zdarzenie)

Funkcje operacji na plikach
- Create file (utworzenie pliku)
- Delete File (usunięcie pliku)
- Open (otwarcie pliku)
- Close (zamknięcie)
- Read (czytanie)
- Get File Attributes (pobranie atrybutów pliku)

Funkcje operacji na urządzeniach we/wy
- Request Device (zamówienie urządzenia)
- Release Device (zwolnienie urządzenia)
- Read (czytanie)
- Write (pisanie)
- Get Device Attributes (pobranie atrybutów urządzenia)

Funkcje utrzymywania informacji
- Get Time or Date (pobranie daty lub czasu)
- Set Time or Data (ustawienie daty lub czasu)
- Get System Data (pobranie danych systemowych)
- Get Process Attributes (pobranie atrybutów procesu)

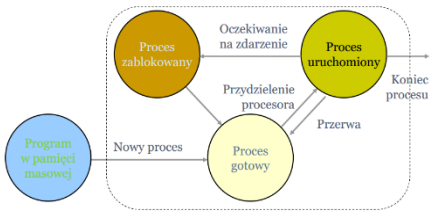
Funkcje utrzymywania komunikacji
- Create, Delete Communication Connection (utworzenie, usunięcie połączenia komunikacyjnego)
- Send, Receive Messages (nadawanie i odbieranie komunikatów)
- Transfer Status Information (przekazanie informacji o stanie).

8. Procesy i ich stany oraz blok kontrolny procesu

Proces – każde działanie systemu komputerowego. Jest obiektem aktywnym (w przeciwieństwie do pliku czy programu).
System komputerowy składa się ze zbioru procesów (np. procesy S.O., procesy użytkownika).

Programy do wykonywania procesów potrzebują niezbędnych zasobów, np. czas procesora, pamięć operacyjna, pliki, urządzenia we/wy.

Stany procesu:
Proces gotowy; proces uruchomiony; proces zablokowany.



Blok kontrolny procesu
Każdy proces ma w systemie operacyjnym przypisany mu blok kontrolny procesu, zawierający dane dotyczące procesu: stan, licznik rozkazów, rejestry procesora, informacje o planowaniu przydziału procesora, inf. o zarządzaniu pamięcią, inf do rozliczeń, inf o stanie urządzeń we/wy.

Zmiana bloku kontrolnego procesu to przełączanie kontekstu.

Przyczyny zakończenia procesów:
- proces zakończył działanie i poinformował system operacyjny o zakończeniu działań;
- proces potomny przekroczył zakres przydzielonych mu zasobów;
- dalsza realizacja procesu obliczeniowego potomka stała się zbędna;
- kończący się proces macierzysty wymusza zakończenie procesów potomnych (zakończenie kaskadowe).

9. Mikrojądro a struktura warstwowa – wady i zalety poszczególnych rozwiązań

Mikrojądro – jądro zredukowane do małego zbioru funkcji rdzeniowych
- większość operacji w przestrzeni użytkownika: obsługa urządzeń, plików, pamięci wirtualnej, grafiki; ochrona przy pomocy komunikatów
Np. Tru64 Unix – Digital Unix

»Zalety:
+Jednolity interfejs dla procesów
+Łatwość w kontrolowaniu kodu systemu operacyjnego
+Przenaszalność systemu (na inną architekturę)
+Niezawodność -małe jądro łatwiej przetestować (ok. 300 kB kodu, 140 funkcji systemowych)
»Wady:
-Wydajność -zbudowanie, zakolejkowanie, wysłanie, odebranie, potwierdzenie komunikatu
-Poprawianie wydajności prowadzi do rozbudowy mikrojądra.

Struktura warstwowa – system operacyjny podzielony na warstwy; warstwa najniższa to sprzęt, najwyższa – interfejs z użytkownikiem. Każda warstwa korzysta z funkcji i usług tylko niżej położonych warstw.

Warstwa API – Application Program Interface – pośrednik między systemem operacyjnym a programami użytkownika; polecenia, które aplikacje użytkownika mogą wydawać systemowi operacyjnemu jak „zamknij plik" itd.

10. Zalety współbieżności procesów, sposoby współpracy procesów

Proces sekwencyjny – ciąg takich instrukcji programu, że następna akcja nie rozpocznie się, zanim nie skończy się poprzednia.

Procesy są współbieżne, jeśli jeden z nich rozpocznie się przed zakończeniem drugiego. Współbieżność uzyskuje się dzięki zasadzie podziału czasu – czas pracy procesora jest dzielony między wszystkie wykonywane współbieżnie procesy.

Procesy współbieżne to takie procesy, które współdzielą moc obliczeniową procesora między wiele procesów naraz.

Korzyści ze współbieżności:
- podział zasobów fizycznych (procesor, we/wy)
- podział zasobów logicznych (pliki)
- multiplikacja obliczeń (w przypadku systemów wieloprocesorowych)
- modularność (łatwa dekompozycja procesów)
- wygoda w użytkowaniu

Proces macierzysty może być wykonywany współbieżnie z procesami potomnymi (w takim przypadku to on przydziela zasoby procesom potomnym), albo może być wstrzymany dopóki procesy potomne nie zakończą pracy (wówczas zasoby przydziela system operacyjny).

Sposoby współpracy procesów
Proces może być niezależny albo współpracujący.

Proces NIEZALEŻNY
1. Na stan procesu nie wpływa żaden inny proces.
2. Działanie procesu jest deterministyczne, czyli wynik zależy jedynie od stanu wejścia.
3. Wynik jest powtarzalny, czyli działanie da się powielać.
4. Działanie może być wstrzymywane i wznawiane dowolną ilość razy bez żadnych skutków

Proces WSPÓŁPRACUJĄCY
1. Oddziałuje na inne procesy.
2. Stan procesu jest dzielony z innymi procesami.
3. Wynik działania procesu jest zależny od stanu innych procesów.
4. Wynik działania jest niedeterministyczny i może b samych stanów wejściowych.

11. Algorytmy planowania przydziału procesora

Kolejki procesów:
- kolejka procesów gotowych
- kolejka procesów do stacji dysków
- kolejka procesów do drukarki

Planista – proces dokonujący wyboru procesów z kolejek do przetworzenia.
Rozpoczyna pracę w momencie, gdy proces przechodzi w stan bezczynności.

- Długoterminowy – wybiera procesy z pamięci masowej i umieszcza w pamięci operacyjnej, gdy wcześniejszy proces opuści system.
- Krótkoterminowy – planista przydziału procesora – przydziela procesor procesom z listy procesów gotowych.
- Średnioterminowy – np. w systemach z podziałem czasu, zarządza uwalnianiem pamięci.

Algorytmy planowania przydziału procesora – kryteria oceny:
- wykorzystanie procesora
- przepustowość, czyli liczbę procesów kończonych w jednostce czasu
- czas potrzebny na realizację procesu
- czas oczekiwania w kolejce procesów gotowych
- czas odpowiedzi (w systemach interakcyjnych lub systemach czasu rzeczywistego)

First Come First Serve – **FCFS** – modelowany kolejką FIFO; może dawać duże średnie czasy oczekiwania.

Shortest Job First – SJF – daje minimalny średni czas oczekiwania.

Algorytm priorytetowy – wymaga nadawania procesiom priorytetów z góry.

Round – Robin – algorytm rotacyjny – dla systemów z podziałem czasu; kolejka procesów jest cykliczna i każdemu procesowi przydziela się stały kwant czasu pracy procesora.

Algorytm planowania z kolejkami wielopoziomowymi – szereg kolejek dla różnych procesów, które mogą być obsługiwane różnymi algorytmami planowania.

Algorytm planowania z kolejkami wielopoziomowymi oraz sprzężeniem zwrotnym – istnieje możliwość zmiany kolejki.

12. Czym jest sekcja krytyczna

do{
sekcja wejściowa
    sekcja krytyczna
sekcja wyjściowa
reszta
}while(1);

Sekcja krytyczna to segment kodu, w którym dokonuje się modyfikacji wspólnych zmiennych. Jako, iż jest to modyfikacja zasobu dzielonego, to gdy jeden proces realizuje kod sekcji krytycznej, to żaden inny proces nie może działać w sekcji krytycznej.
Do sekcji krytycznej może wejść tylko taki proces, który nie wykonuje akurat swojej reszty.
Proces może oczekiwać na wejście do sekcji krytycznej skończony odcinek czasu.

13. Zadania i rodzaje semaforów (binarne, zliczające, standard POSIX)

Semafor – mechanizm synchronizacji procesu. Jest to zmienna całkowita, która przyjmuje wartości nieujemne lub dla semaforów binarnych – logiczne.
Jest ona dostępna za pomocą dwóch, niepodzielnych operacji:
- Wait (czekaj) / Proberen **P** – opuszczenie semafora, czyli zmniejszenie jego wartości;
- Signal (sygnalizuj) / Verhogen **V –** podniesienie semafora, czyli zwiększenie jego wartości.

Niepodzielność semafora oznacza, że gdy zmienna jest modyfikowana przez proces, żaden inny proces nie ma prawa zmieniać jej wartości.

Dla instrukcji „czekaj" nie może wystąpić przerwanie w czasie ani sprawdzania, ani modyfikacji zmiennej.

Semafor binarny – zmienna logiczna (true - podniesiony/false - opuszczony).

W przeciwieństwie do semafora binarnego semafor zliczający „pamięta" liczbę operacji podniesienia – przy wartości początkowej 0 można wykonać tyle operacji opuszczenia, ile razy został wcześniej podniesiony. Wartość semafora nie może osiągnąć liczby ujemnej.

14. Procesy jedno i wielowątkowe

Proces tradycyjny składa się z wątków.

Wątek – proces lekki – podstawowa jednostka wykorzystania procesora. Składa się z:
- identyfikatora
- licznika rozkazów
- zbioru rejestrów
- stosu.

Proces tradycyjny ma jeden wątek sterowania.

Różnica między zwykłym procesem a wątkiem polega na współdzieleniu przez wszystkie wątki działające w danym procesie wszystkich struktur systemowych tego procesu - z kolei procesy posiadają niezależne zasoby.

Korzyści wielowątkowości:
- łatwość komunikacji między wątkami jednego zadania
- dzielenie zasobów – wspólna pamięć i zasoby procesu do którego należą
- ekonomika – bardziej opłacalne przełączanie kontekstu niż przydzielanie osobnych zasobów dla procesów
- wykorzystanie architektury wieloprocesorowej – zwiększenie wykorzystania zalet pracy wielowątkowej.

Typy wątków:
Wątki użytkownika – realizowane na poziomie użytkowym. Jądro nie posiada informacji na temat wątków tworzonych przez użytkownika.
Wątki jądra – udostępniane bezpośrednio przez system operacyjny. Jądro zajmuje się tworzeniem i planowaniem wątków oraz administruje je we własnej przestrzeni.

15. Modele wielowątkowości

Model „wiele na jeden" – wiele wątków użytkownika przypada na jeden wątek jądra; wątki przełączane są między dostępem do wątku jądra.

Model „jeden na jeden" – jeden wątek użytkownika przypada na jeden wątek jądra; „fajnie wygląda ale niebezpieczne" bo ilość zasobów ogranicza ilość wątków jądra też wątków użytkownika.

Model „wiele na wiele" – wiele wątków użytkownika przypada na wiele wątków jądra – za każdym razem ustalona (np. 3) ilość wątków jądra; najbezpieczniejsze rozwiązanie.

Pula wątków – utworzenie pewnej liczby wątków w czasie rozruchu procesu i zaliczenie ich do puli, w której oczekują na zadania. Po zakończeniu zadania wątek powraca do puli, gdzie znów oczekuje. Jeśli nie ma wolnych wątków – serwer czeka do czasu aż któryś się zwolni. Brak konieczności tracenia czasu na tworzenie wątku + ograniczenie liczby wątków istniejących (zaleta dla systemów nie mogących zapewnić dużej liczby wątków).

Standard POSIX – standard interfejsu systemu operacyjnego - umożliwienie pisania przenośnego kodu aplikacji, który miał w założeniu działać na wszystkich zgodnych z tym standardem systemach. Definiuje interfejs programów użytkowych API do tworzenia wątków i ich synchronizacji. Jest to specyfikacja, a nie implementacja.

16. Problemy synchronizacji procesów

Problem czytelników i pisarzy

Obiekt danych podlega współbieżnemu dzieleniu między kilka procesów.
Dwa typy procesów:

- Zainteresowane czytaniem danych – czytelnicy
- Zainteresowane aktualizacją danych – pisarze

W przypadku dostępu do obiektu przez pisarza, żaden inny proces nie może mieć do niego dostępu równocześnie – wyłączność procesów pisarzy do obiektów.

Opcje ustawienia priorytetów:
- żaden czytelnik nie powinien czekać, chyba że właśnie pisarz ma dostęp do obiektu;
- pisarz otrzymuje pierwszeństwo nad czytelnikami i żaden nie rozpocznie czytania gdy pisarz czeka.

Problem obiadujących filozofów

5 filozofów spędza czas na myśleniu i jedzeniu – proste odwzorowanie konieczności przydzielenia wielu zasobów do wielu procesów w sposób grożący zakleszczeniem lub głodzeniem.
Aby uniknąć zakleszczenia:
- pozwolić zasiadać do stołu najwyżej 4 filozofom;
- pozwolić na podnoszenie pałeczek tylko wtedy, gdy obie są wolne;
- rozwiązanie asymetryczne: filozof nieparzysty najpierw podnosi pałeczkę po lewej stronie, a później po prawej; filozof parzysty najpierw podnosi pałeczkę po prawej stronie, a potem po lewej.

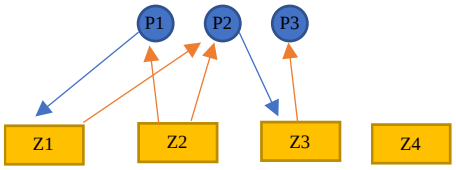
17. Graf przydziału zasobów (z cyklem, z blokadą, bez blokady)

Blokada – zakleszczenie; sytuacja, gdy procesy nie mogą uzyskać dostępu do zamówionych zasobów, gdyż oczekują na zasoby wzajemnie przetrzymywane.

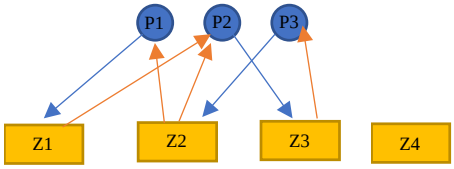
Warunki konieczne blokady:

- Wzajemne wyłączenie – co najmniej jeden zasób musi być niepodzielny
- Przetrzymywanie i oczekiwanie – musi istnieć proces przetrzymujący zasób i oczekujący na przydział innego zasobu, przetrzymywanego przez inny proces.
- Brak wywłaszczeń – zwolnienie zasobu może nastąpić jedynie z inicjatywy procesu przetrzymującego go.
- Czekanie cykliczne – istnieje zbiór procesów {P0, P1, .. , Pn} takich, że P0 czeka na zasób przetrzymywany przez P1, P1 na P2, a Pn na zasób przetrzymywany przez P0.

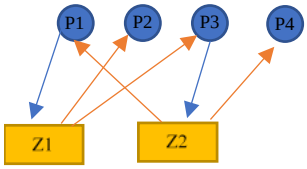
Graf zorientowany G = <W, K>. Zbiór wierzchołków **W** grafi składa się z podzbioru procesów **P** oraz podzbioru zasobów **Z**.
Krawędź skierowana od procesu P_i do zasobu Z_j oznacza że proces zamówił zasób i czeka.
Krawędź skierowana od zasobu Z_j do procesu P_i oznacza, że zasób został przydzielony procesowi.
Kiedy proces zamawia zasób, krawędź zamówienia umieszcza się w grafie. Gdy zasób zostanie przydzielony, krawędź zamówienia zamienia się na krawędź przydziału.
Po zwolnieniu zasobu krawędź przydziału zostaje usunięta z grafu.



Bez blokady



Z blokadą



Z cyklem ale bez blokady

18. Sposoby rozwiązywania problemu blokad

Wykrywanie i usuwanie blokad jest trudniejsze i kosztowniejsze niż zapobieganie im, dlatego przede wszystkim system operacyjny powinien unikać blokad.

Aby blokada nie wystąpiła wystarcza, aby nie był spełniony przynajmniej jeden z warunków koniecznych blokady.

Zapobieganie blokadom:

Niemożliwym jest zapobieganie blokadom poprzez wykluczenie wzajemnego wyłączenia, gdyż dotyczy to zasobu, który jest niepodzielny.

Aby wykluczyć przetrzymywanie i oczekiwanie:
- na etapie wprowadzania procesu zamówić wszystkie zasoby, które będą potrzebne
- proces może przedstawić zamówienie na zasób wtedy, gdy zwolni wszystkie dotychczas wykorzystywane.

Aby uniknąć blokady z powodu braku wywłaszczeń:
- gdy proces zgłasza zapotrzebowanie na kolejny zasób, traci w sposób niejawny wszystkie swoje zasoby; proces zostaje wznowiony, gdy odzyska wszystkie swoje zasoby + zamówiony.
- gdy proces zgłasza zapotrzebowanie na kolejny zasób, ale ten jest zajęty, z puli swoich zasobów oddaje te, na które jest zamówienie; proces zostaje wznowiony, gdy odzyska wszystkie swoje zasoby + zamówiony.

Aby uniknąć sytuacji cyklicznego czekania:
- należy uporządkować zasoby przydzielając im numery i przydzielać zasób Z_j tylko wtedy, gdy $F(Z_j) > F(Z_i)$.

Wykrywanie i usuwanie blokad:

Algorytm wykrywania blokady sprawdza, czy spełniony jest warunek cyklicznego czekania, po czym po stwierdzeniu blokady:

- usuwa się wszystkie procesy uczestniczące w blokadzie
- jeśli procesy mają punkty kontrolne, wówczas cofają się do nich i rozpoczynają realizację procesów od tych punktów
- z puli procesów oczekujących usuwa się procesy tak długo, aż zostanie usunięta blokada
- zawłaszcza się zasoby procesów uczestniczących w blokadzie.

19. Cele zarządzania pamięcią operacyjną (mechanizm nakładek, pamięć wirtualna, stronicowanie)

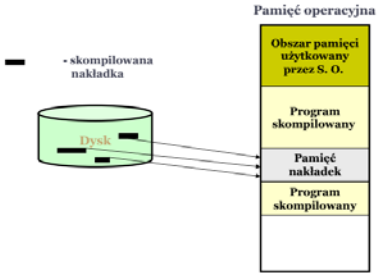
Adres fizyczny to rejestr bazowy + adres logiczny.

Procesy współdzielą między sobą pamięć operacyjną, a aktywny proces obliczeniowy musi być umieszczony w pamięci.
Zanim proces stanie się aktywny, jest on wprowadzany do pamięci operacyjnej z kolejki wejściowej.

Cele zarządzania pamięcią to:
- relokacja procesów w pamięci – liczba procesów może się zmieniać, dlatego nie są z góry znane ich adresy; system operacyjny musi przekształcać adresy użyte w programie na adresy rzeczywiste.
- ochrona zawartości pamięci – system operacyjny musi analizować wszystkie odwołania procesów do pamięci sprawdzając, czy odwołania dotyczą przydzielonego obszaru pamięci dla danego procesu.

- dostęp do obszarów współdzielonych – system operacyjny musi sprawować kontrolę oraz ochronę obszarów w sytuacji, kiedy kilku procesom umożliwia się dostęp do tych samych obszarów pamięci.
- organizacja logiczna pamięci – rzeczywista struktura pamięci jest jednowymiarowa, uporządkowana liniowo, dlatego stosuje się segmentację pamięci, gdzie poszczególne segmenty można kodować niezależnie oraz wyznaczać im różne poziomy ochrony czy tworzyć mechanizmy współdzielenia.
- organizacja fizyczna pamięci – aby uzyskać kompromis między szybkością pamięci a jej pojemnością zaczęto tworzyć pamięci dwupoziomowe: pamięć operacyjna i pamięć masowa (pomocnicza; np. dysk). Pojawił się problem wymiany danych między pamięciami – obecnie system operacyjny organizuje wymianę danych tworząc wirtualną organizację pamięci. (Najpierw próbowano rozwiązać to mechanizmem nakładek).

Mechanizm nakładek



Pamięć wirtualna – mechanizm umożliwiający zmianę logicznej przestrzeni adresowej widzianej przez programistę na fizyczną przestrzeń adresów w pamięci komputera.
Pozwala to programiście dysponować obszarem pamięci adresów nie związanym z rzeczywistą wielkością przestrzeni pamięci (czyli przestrzeń adresów nie musi być liniowa). Podstawą funkcjonowania pamięci wirtualnej jest mechanizm stronicowania na żądanie.

Stronicowanie – podział pamięci fizycznej na ramki stron oraz podział logicznej przestrzeni adresowej (przestrzeni adresów wirtualnych) na strony o takim samym rozmiarze jak ramki w pamięci fizycznej. Pozwala to na zwiększenie przestrzeni adresów – celem stronicowania jest fizyczny podział pamięci.

Strony procesu mogą być aktywne – ramki stron umieszczone w pamięci – lub nieaktywne – ramki stron umieszczone w pamięci pomocniczej.

Stronicowanie realizuje odwzorowywanie adresów procesu w numer strony oraz numer słowa na stronie; oraz przesyła strony między pamięcią główną i pomocniczą.
Przekroczenie zakresu numeracji bajtu powoduje zwiększenie numeru strony.

Rozmiar stron jest ustalony i wynikający z architektury systemu.

20. Strategie rozmieszczania danych w pamięci i zarządzanie wolnymi obszarami

Strategie rozmieszczania to jedna z kategorii strategii związanych z przydziałem obszarów fizycznej pamięci.
Strategie wymiany służą do określania tych danych w pamięci, które mogą być usunięte dla uwolnienia pamięci dla innych danych.
Strategie pobierania określają sytuacje, kiedy informacje przechowywane w pamięci pomocniczej mają być załadowane do pamięci głównej.

Strategie rozmieszczania służą do wyboru miejsc w pamięci głównej, do których ma być załadowana informacja pobierana z pamięci pomocniczej.

Dla systemów bez stronicowania

Segmentacja to logiczny podział przestrzeni adresów.

Strategie rozmieszczania danych wymagają aktualnej listy adresów oraz rozmiarów dziur pamięci.
Jeśli rozmiar dziury jest większy od rozmiaru segmentu, wówczas segment zajmuje pewien obszar dziury, zaś pozostała część staje się dziurą.

1. Strategia najlepszego dopasowania – gdy dziury uporządkowane są rosnąco pod względem rozmiaru; segment zostaje przydzielony do takiej dziury, że jego rozmiar jest mniejszy lub równy rozmiarowi dziury.
2. Strategia najgorszego dopasowania – gdy dziury uporządkowane są malejąco pod względem rozmiaru – mniejsze dziury, które pozostaną po umieszczeniu w największych dziurach segmentów, mogą nadal być na tyle duże by mieściły się w nich inne segmenty.
3. Strategia pierwszego dopasowania – gdy dziury uporządkowane są rosnąco pod względem ich adresów bazowych; algorytm grupuje adresy małych dziur na początku listy.
4. Algorytm bliźniaków (Buddy algotithm) – możliwość łączenia dziur w jedną dużą dziurę lub dzielenia jej na dwie mniejsze.

21. Metody przydziału miejsca na dysku

Pamięć główna jest medium ulotnym – informacje ulegają stracie po zaniku zasilania.
Dane jak i programy przechowuje się w pamięci pomocniczej – rolę pomocniczej pamięci masowej pełnią obecnie dyski.

Dysk składa się z napędu dysku oraz sterownika dysku.
Ogranizacja dysku: ścieżki, sektory.

Zarządzanie pamięcią dyskową związane jest z zarządzaniem wolnymi obszarami pamięci (wiążą się z tym lista powiązana oraz grupowanie), z zarządzaniem przydziałem miejsca na dysku oraz z planowaniem dostępu do dysku.

Przydział miejsca na dysku
Powinien uwzględniać dwa aspekty:
- efektywne wykorzystanie dysku
- szybki dostęp do plików.

Metody:
- metoda ciągła – każdy plik musi zajmować ciąg adresów na dysku, przy czym adresy definiują uporządkowanie liniowe. Może realizować dostęp bezpośredni i sekwencyjny ale wymaga deklaracji rozmiaru pliku.
- metoda listowa – pliki tworzone są na zasadzie listu co pozwala uniknąć fragmentacji zewnętrznej i nie wymaga deklaracji wielkości pliku, ale zapewnia jedynie dostęp sekwencyjny.
- metoda indeksowa – dla każdego pliku umieszcza się na dysku tablicę adresów bloków dyskowych.

22. Planowanie dostępu do dysku

Odpowiednie metody planowania dostępu do dysku pozwalają na zmniejszenie czasu przeszukiwania dysku.

Metody planowania:
- metoda FCFS – First Come First Serve (jak przy planowaniu przydziału procesora) – przeszukiwanie dysku realizowane według kolejki ścieżek bez ingerencji w kolejność kolejnych punktów;
- metoda SSFT - Shortest Seek Time First – przeszukiwanie dysku realizowane jest poprzez poruszanie się na zasadzie punkt najmniej oddalony od obecnego najpierw;
- metoda SCAN – przeszukiwanie dysku realizowane jest poprzez powrót do punktu 0 przez wszystkie punkty mniejsze niż obecny i następnie przejście przez punkty większe niż punkt z którego zaczęto;
- metoda C-SCAN – Cyclic SCAN – na odwrót niż wyżej, z punktu startowego do punktu największego i dalej do 0, skąd do punktów mniejszych
- metoda LOOK oraz C-LOOK

23. RAID – zadania, zalety, sposoby zapisu danych na macierzach dyskowych

Redundant Array of Independent Disc
Jest to sposób połączenia dysków fizycznych w jedną macierz, którą system operacyjny traktuje jako jeden napęd logiczny.

Macierze RAID oferują:
- odporność na awarie;
- zwiększenie prędkości transmisji w porównaniu z pojedynczym dyskiem;
- powiększenie przestrzeni dostępnej jako jedna całość (obecnie nie chodzi już o dużą pojemność, tylko o zabezpieczenie danych)

Potrafią one pracować zarówno równolegle jak i niezależnie.

RAID 0 (Paskowanie)
Dane przeplatane są między dyskami – brak danych nadmiarowych, dlatego w przypadku awarii któregoś z dysków nie ma możliwości odzyskania ich. Szybki transfer danych – używane np. jako pamięć PROXY, gdzie nieistotna stabilność danych, ale szybkość.

RAID 1
Całkowita pojemność jest taka jak pojemność najmniejszego dysku, ale zwiększone bezpieczeństwo danych – zapisywanie danych równolegle na dwóch dyskach.

RAID 2
Całkowita pojemność to suma pojemności dysków. Kod Hamminga pozwala niwelować błędy jednobitowe oraz wykrywać (ale już nie niwelować) błędy podwójne. Zredukowanie liczby nadmiarowych danych. Każdy dysk może w razie uszkodzenia zostać odbudowany przez pozostałe dyski. Konieczność synchronizacji wszystkich dysków z kodem Hamminga. Stworzony głównie do badań bo zawiłe obliczenia.

RAID 3 (Parzystość na poziomie bitu)
Rezygnacja z kodu Hamminga i wprowadzenie prostego bitu parzystości. Brak asynchronicznej pracy macierzy.

RAID 4 (Parzystość na poziomie paska)
Rozszczepia dane na poziomie paska, a nadmiarowe dane z bitu parzystości idą na inny dysk.(?) Dane zapisywane w dużych blokach, ale spowolniony zapis. Każdy zapis na dysku -> drugi zapis na dysku nadmiarowym.

RAID 5 (Parzystość rozproszona na poziomie paska)
Dane nadmiarowe rozproszone po wszystkich dyskach – zapisywane rotacyjnie na wszystkich dyskach. Duży wzrost prędkości zapisu, dane odtwarzane za pomocą kodu parzystości w przypadku uszkodzenia dysku.

RAID 6 (Podwójna parzystość rozproszona na poziomie paska)
Dodatkowo zwiększa bezpieczeństwo danych. Zmniejszenie minimalnej liczby dysków do 4, a do odzyskania danych mogą uszkodzić się na raz maksymalnie dwa dyski.