**Sys oper** (cechy: wydajność, niezawodność, pielęgnowalność, mały rozmiar, ergonomia)

Przez system operacyjny będziemy rozumieli warstwę oprogramowania oddzielającą użytkownika od sprzętu.

System operacyjny udostępnia różnorodne usługi i ma na celu ułatwienie użytkownikowi w korzystaniu z komputera.

System operacyjny zwiększa wykorzystania takich zasobów jak procesory, pamięć i urządzenia we/wy. Tworzy środowisko dla (innych) programów

**Masz wykorz wsadowo**

Gromadzi się zadania wykorzystujące te same zasoby np. ten sam kompilator i wykonuje jedno podrugim. Zysk – nie trzeba za każdym razem ładować innego kompilatora. Wady - brak bezpośredniego nadzoru ze strony użytkownika.

Ważne pojęcia:

**język opisu zadania** – polecenia interpretowane przez monitor np.: załaduj kompilator, skompiluj program, uruchom program.

**monitor** – program rezydentny, który jest interpreterem języka opisu zadania i przekazuje sterowanie z jednego zadania do następnego

**przetwarz pośrednie**

Kopiowanie danych z karty na taśmę lub z taśmy na karty odbywało się na innej specjalnie do tego celu skonstruowanej maszynie, która zajmowała się tylko tym zadaniem. Czasami do tego celu wykorzystywano mały komputer (zwany satelitą).

Wcześniej: czytnik kart -> jednostka centralna -> drukarka wierszowa

Teraz: czytnik kart -> przewijaki taśm -> jednostka centralna -> przewijaki taśm -> drukarka wierszowa

**Buforowanie** – gromadzenie danych w zarezerwowanych obszarach pamięci.

wejście -> bufor -> jednostka centralna -> bufor -> wyjście  
**spooling** – ta sama zasada co w buforowaniu lecz jako bufor wykorzystuje się fragment twardego dysku – zysk bufor może być duży.

**Wieloprogramowość**

system operacyjny wybiera zadanie i zaczyna je wykonywać. W chwili gdy zadanie zaczyna oczekiwać na jakąś akcję, system przechodzi do wykonywania innego zadania itd. Kiedy pierwsze zadanie zakończy oczekiwanie to system wznawia wykonywanie zadania pierwszego lecz dopiero po zakończeniu aktualnie wykonywanego zadania.

**Trzy główne elementy komputera** z programem zintegrowanym to: jednostka centralna (*CPU*), pamięć (*memory*) oraz magistrala łącząca pamięć z jednostką centralną.

**Pamięć**- W modelu von Neumana pamięć jest rozumiana jako pamięć główna (operacyjna) – jest to ta część pamięci, którą procesor może zaadresować podczas wykonywania pojedynczego rozkazu. Jeżeli w systemie istnieje pamięć wtórna (masowa) to przed użyciem informacji przechowywanej w takiej pamięci musi ona zostać skopiowana do pamięci głównej.

**Organizacja pamięci** -informacja przechowywana jest w komórkach o jednakowym rozmiarze, każda komórka zawiera jednostkę informacji zwaną słowem,

sposób przechowywania danych i instrukcji jest taki sam (kod nie zawiera etykiety), struktura (kod) słowa nie pozwala odróżnić instrukcji od danych – interpretacja słowa zależy tylko od stanu maszyny w chwili pobierania słowa z pamięci,

komórki tworzą zbiór uporządkowany, a każdej komórce można przypisać unikatowy identyfikator tzw. adres,

zawartość komórki pamięci może być zmieniona tylko przez procesor jako przesłanie słowa do pamięci w wyniku wykonania rozkazu.

**Procesor -** Zadaniem procesora jest przetwarzanie danych przechowywanych w pamięci głównej. „Wykonanie rozkazu inicjuje pobranie z pamięci słowa interpretowanego jako kod instrukcji i obejmuje: pobranie danych z pamięci, ich przetworzenie i zapis wyniku do pamięci”. W celu odciążenia magistrali dopuszcza się używania rejestrów do chwilowego przechowywania informacji, najczęściej są to wyniki pośrednie.

**Rejestr** – pojedyncza komórka pamięci stanowiąca część procesora. Zapis i odczyt rejestru są operacjami bardzo szybkimi, nie obciążają magistrali procesora.

**Program -** W modelu von Neumanna „program jest sekwencją instrukcji co oznacza, że w chwili wykonywania instrukcji znany jest adres lub sposób wyliczenia adresu następnej instrukcji”. W przypadku gdy kolejne instrukcje zostaną umieszczone w kolejnych lokacjach pamięci możliwe jest ich automatyczne adresowanie. Układ zajmujący się wytwarzaniem adresu następnego rozkazu nazywa się licznikiem rozkazów (*program counter*). Do zmiany domyślnej kolejności wykonywania rozkazów służą rozkazy rozgałęzień (*branch*) lub skoków (*jump*), które powodują wymuszenie nowej zawartości licznika rozkazów. Odpowiednikami w/w rozkazów w językach wysokiego poziomu są konstrukcje *if* warunek *then* oraz *goto*.

**Rozszerzenie archit von N** - stos programowy (*stack*) część pamięci o dostępie sekwencyjnym modelująca pracę bufora typu LIFO (*Last-in first-out*), której użycie upraszcza wywoływanie procedur i umożliwia rekursję.

**tryby adresowania** – metody wyznaczania adresu umożliwiające elastyczne i dynamiczne adresowanie, pośrednie i wieloelementowe

pamięć wirtualna

nowe typy danych np. liczby dziesiętne, liczby zmiennoprzecinkowe, łańcuchy danych

**Bariera von N** - Ograniczeniem mającym istotny wpływ na wydajność komputera jest szybkość i częstość komunikacji procesora z pamięcią. Efekt ten zwany jest barierą von Neumanna. Czas dostępu do danych nie zależy tylko do szybkości ich transmisji ale także od szybkości selekcji tych danych. Szybkość selekcji jest proporcjonalna do logarytmu liczby danych.

**Łagodzenie bariery**

implementacja rejestrów wewnętrznych,

implementacja pamięci podręcznej (*cache*),

przeplot pamięci.

Przeplot pamięci – podział pamięci głównej na banki tak aby był możliwy dostęp do banku X zanim zakończy się dostęp do banku Y.

**Archit procesora**

model programowy opisujący działanie podstawowych jednostek procesora. Mówiąc o architekturze procesora będziemy mieli na myśli uproszczony opis działania podstawowych jednostek procesora.

Podstawowymi elementami funkcjonalnymi architektury komputera są:

procesor zawierający:

jednostkę sterującą

jednostkę wykonawczą

jednostkę adresową

pamięć przechowująca dane

magistrala łącząca procesor z pamięcią

**Przerwanie** – przerwanie zewnętrzne – jest informacją o wystąpieniu asynchronicznego zdarzenia zewnętrznego w stosunku do procesora, jest niezależne od aktualnie wykonywanych przez procesor instrukcji.

**Wyjątek** – przerwanie wewnętrzne – przerwanie generowane wewnętrznie przez procesor, jest związany z aktualnie wykonywaną instrukcją, która powoduje wystąpienie wyjątku, np. próba dzielenia przez 0, błąd koprocesora, brak segmentu, brak strony.

**Hieriarchia pamięci**

Podział pamięci ze względu na czas dostępu:

rejestry procesora

pamięć podręczna,

pamięć operacyjna,

dysk elektroniczny,

dysk magnetyczny,

dysk optyczny,

taśmy magnetyczne.

**System przerwań**

Projektanci systemów operacyjnych dążąc do zwiększenia efektywności wykorzystania systemów komputerowych dążyli do równoległego wykorzystania urządzeń we/wy i procesora. Aby umożliwić pracę urządzeń we/wy współbieżnie z pracą procesor należy zapewnić mechanizmy umożliwiające poinformowanie procesora o rozpoczęciu lub zakończeniu operacji we/wy. Są dwie metody pozyskania takich informacji:

implementacja systemu przerwań

**pooling -** „odpytywanie” przez system operacyjny urządzeń co powoduje opóźnienia w wykonywaniu wszystkich procesów. Dodatkową trudnością jest określenie interwału czasowego pomiędzy kolejnymi pytaniami (nie za długi, nie za krótki). wykorzystywana w tych procesorach, które nie obsługują mechanizmu przerwań.

**Zarządzanie procesami**

tworzenie i usuwanie procesów użytkowych i systemowych,

wstrzymywanie i wznawianie procesów,

dostarczanie mechanizmów synchronizacji procesów,

dostarczanie mechanizmów komunikacji procesów,

dostarczanie mechanizmów obsługi zakleszczeń.

**Zarządzanie pamięcią oper**

utrzymywanie ewidencji obszarów zajętych wraz z informacją, który proces działa w danym obszarze,

decydowanie o tym, które procesy mają być załadowane do zwolnionych obszarów pamięci,

przydzielanie i zwalnianie obszarów pamięci stosownie do potrzeb.

**Zarządzanie plikami**

tworzenie i usuwanie plików

tworzenie i usuwanie katalogów

dostarczanie elementarnych operacji do manipulowania plikami i katalogami

odwzorowywanie plików na obszary pamięci pomocniczej

składowanie plików w trwałych nośnikach pamięci

**Zarządzanie pamięcią pomocniczą**

zarządza obszarami wolnymi,

przydziela pamięć,

realizuje zadania związane z planowaniem, dostępu do obszarów pamięci pomocniczej.

**Nadzorowanie procesów**

zakończenie, zaniechanie;

załadowanie, wykonanie;

utworzenie procesu, zakończenie procesu – Win32API - *TerminateProcess;*

pobranie atrybutów, określenie atrybutów;

oczekiwanie na zdarzenie, sygnalizowanie zdarzenia;

przydział i zwolnienie pamięci;

oczekiwanie czasowe - Win32API - *Sleep(100).*

**Stany procesu**

nowy – system utworzył nowy proces;

aktywny – są wykonywane instrukcje;

wstrzymany – proces czeka na wystąpienie jakiegoś zdarzenia;

gotowy – proces czeka na przydział procesora;

zakończony – proces zakończył działanie.

**Kontekst procesu**

Informację niezbędną do wstrzymania lub wznowienia procesu nazywamy **kontekstem procesu**. W skład kontekstu procesu wchodzi **blok sterujący** oraz **kontekst pamięci**. Kontekst pamięci to obszar pamięci, w którym wykonuje się proces.

**Przełączanie procesów**

Wykonywanie wielu procesów na jednym procesorze wymaga ich przełączania, ponieważ w danej chwili czasu tylko jeden proces może korzystać z unikatowego zasobu jakim jest czas procesora. Przełączanie procesów jest inicjowane przez wywołanie przełączania i wymaga przełączania kontekstów, przechowania kontekstu procesu zawieszanego i odtworzenie kontekstu procesu wznawianego.

**Tworzenie procesu**

Proces może tworzyć nowe procesy za pomocą odpowiednich funkcji systemowych. Proces tworzący nowe procesy nazywany jest **procesem macierzystym**, a utworzony przez niego nowy proces **procesem potomnym**. Proces macierzysty może rozdzielać własne zasoby między procesy potomne również tak, aby niektóre zasoby były współdzielone przez te procesy.

**Wątki**

proces wykonywany w tym samym kontekście pamięci procesu macierzystego. Środowisko w którym działa wątek nazywa się zadaniem (ang. task). Tradycyjny „ciężki” proces jest równoważny zadaniu z tylko jednym wątkiem. Pojedynczy wątek ma przynajmniej własny kontekst procesora i na ogół stos. Przełączanie pomiędzy równoprawnymi wątkami jest „tanie” (szybsze) w porównaniu z przełączaniem pomiędzy tradycyjnymi procesami (nie trzeba przełączać kontekstu pamięci). Wątki zyskują na popularności ponieważ – mając pewne cechy ciężkich procesów – są efektywniejsze w działaniu.

**Wieloprogramowość**

Celem wieloprogramowania jest maksymalizowanie wykorzystania jednostki centralnej przez stałe utrzymanie w działaniu pewnej liczby procesów. W systemie jednoprocesorowym nigdy nie jest wykonywany więcej niż jeden proces. Jeżeli istnieje więcej procesów, to pozostałe muszą czekać do chwili, gdy procesor zostanie zwolniony i będzie można ponownie przydzielić go jednemu z nich.”

**Planowanie długoterminowe** – planista długoterminowy podejmuje decyzję o tym, który z procesów ma być załadowany z pamięci masowej do pamięci głównej.

**Planowanie średnioterminowe** – pośredni poziom planowania - wymiana procesów pomiędzy pamięcią główną a pamięcią masową:

**Kryteria planowania**

efektywność - wykorzystanie procesora, w rzeczywistych systemach powinno wahać się w granicach 40-90%;

przepustowość – liczba procesów kończonych w jednostce czasu;

czas cyklu przetwarzania (czas wykonania zadania) – czas pomiędzy chwilą nadejścia procesu do systemu a chwilą zakończenia procesu;

czas oczekiwania – czas który proces spędza w kolejce procesów gotowych do wykonania;

czas odpowiedzi – czas między upływający między przedłożeniem zamówienia a pojawieniem się pierwszej odpowiedzi.

**Algorytmy planowania**

FCFS (ang. *First Come First Served*) – z kolejki procesów gotowych do wykonania jest wybierany ten który najszybciej się znalazł w tej kolejce. Algorytm jest algorytmem niewywłaszczającym.

SJF (ang. *Shortest Job First*)– najkrótsze zadanie pierwsze, tak naprawdę to jest „najpierw najkrótsza następna faza procesora”. Ciężko przewidzieć jaką długość będzie miała następna faza procesora, więc musi ona zostać oszacowana. Szacuje się na podstawie poprzednich faz procesora danego procesu. Algorytm może mieć dwie odmiany (Niewywłaszczającą, wywłaszczającą)

Algorytm jest często używany w planowaniu długoterminowym.

Priorytetowe - Każdy proces ma nadany priorytet a planista krótkoterminowy wybiera z kolejki procesów gotowych do wykonania proces o najwyższym priorytecie. Priorytety mogą być nadawane:

wewnętrznie przez system operacyjny;

zewnętrznie przez użytkownika lub administratora.

Problem – głodzenie procesów niskopriorytetowych.

Możliwe dwie odmiany wywłaszczeniowa i niewywłaszczeniowa.

Rotacyjne - Z kolejki procesów gotowych do wykonania wybierany jest pierwszy z procesów i uruchamiany na kwant czasu. Po upłynięciu kwantu czasu, system zaczyna wykonywać następny proces, a proces, który został wstrzymany wędruje na koniec kolejki. Schemat jest powtarzany aż do wykonania wszystkich procesów lub zatrzymania systemu.

W tym algorytmie bardzo istotne jest odpowiednie dobranie kwantu czasu np. tak aby 80% procesów była blokowana (np.: czekając na we/wy) a nie wywłaszczana.

Wielopoziomowe planowanie kolejek - W systemie istnieje kilka kolejek procesów gotowych do wykonania z różnymi priorytetami i często z różnymi algorytmami przydziału w każdej kolejce.

Wielopoziomowe planowanie kolejek ze sprzężeniem zwrotnym - W systemie istnieje kilka kolejek procesów gotowych do wykonania a procesy mogą być przemieszczane między kolejkami. Przy wyborze tego rodzaju planowania trzeba określić:

liczbę kolejek;

algorytmy planowania dla każdej z kolejek;

zasadę mówiącą o przesunięciu procesu do kolejki o priorytecie niższym (lub wyższym);

regułę do której kolejki trafia nowy proces.

Planowanie wieloprocesorowe - W systemie homogenicznym ze względów praktycznych istnieje jedna kolejka procesów gotowych do wykonania i możliwe są dwa schematy przetwarzania:

każdy z procesorów samodzielnie przegląda i wybiera jeden z procesów do wykonania. Należy zadbać aby dwa procesory nie wybrały jednego procesu do przetwarzania.

jeden z procesorów jest poświęcany do realizacji zadań planowania i to on przydziela innym procesorom procesy do wykonania.

**Sekcja krytyczna** – fragment kodu w którym proces korzysta z zasobu współdzielonego np. zmiennych, plików itd.

Aby rozwiązać problem sekcji krytycznej należy zapewnić:

wzajemne wykluczanie – tylko jeden proces może przebywać w sekcji krytycznej,

postęp – każdy proces czekający na wejście do sekcji krytycznej uzyska do niej dostęp,

warunek ograniczonego czekania – zapewnić, że proces nie będzie czekał w nieskończoność.

**Semafor** S jest zmienną całkowitą, która niezależnie od inicjalizacji jest dostępna przy pomocy dwóch standardowych, niepodzielnych operacji: czekaj i sygnalizuj.

**Komunikacja między procesowa**

Trzy najbardziej popularne metody komunikacji między procesami występujące w najpopularniejszych systemach dla komputerów rodziny PC to:

pamięć współdzielona,

kolejki komunikatów,

łącza komunikacyjne.

Inne formy komunikacji, które można zaliczyć do komunikacji między procesowej to:

gniazda,

zdalne wywoływanie procedur.

**Pamięć współdzielona** to najszybsza forma komunikacji pomiędzy dwoma procesami w obrębie danego systemu operacyjnego.

Dane wymieniane pomiędzy dwoma niezależnymi (niespokrewnionymi) procesami nie wymagają kopiowania danych typu „proces pierwszy->jądro systemu-> proces drugi”, a wymieniane są bezpośrednio pomiędzy zainteresowanymi procesami. Używanie pamięci współdzielonej wymaga synchronizacji dostępu do dzielonego obszaru pamięci.

**Kolejki FIFO**

Jedną z wad łącz komunikacyjnych jest możliwość komunikacji tylko procesów spokrewnionych. W celu wyeliminowania tego problemu powstały łącza nazwane – inaczej kolejki FIFO. W przeciwieństwie do łącz nienazwanych za ich pomocą można komunikować procesy niespokrewnione ponieważ są identyfikowane w systemie poprzez nazwę. Zarówno łącza jak i kolejki FIFO mogą być jedno i dwu kierunkowe.

**Zakleszczenie**

stan zakleszczenia występuje wtedy kiedy dwa lub więcej procesów czeka w nieskończoność na zdarzenie, które może być spowodowane tylko przez jeden z tych czekających procesów.

Zakleszczenie jest zjawiskiem niekorzystnym ponieważ procesy w nie uwikłane nie wykonują się.

Większość współczesnych systemów operacyjnych nie ma środków zapobiegającym zakleszczeniom.

**Zasoby**

System operacyjny składa się ze skończonej liczby zasobów. Zasoby mogą być reprezentowane wielokrotnie co oznacza, że istnieje kilka egzemplarzy danego zasobu. Zasoby są zamawiane i zwalniane poprzez użycie specjalnych funkcji systemowych. Użycie zasobu:

zamówienie – czasami związane z oczekiwaniem na zwolnienie się zasobu,

użycie zasobu,

zwolnienie zasobu.

Warunki konieczne do powstania zakleszczenia

wzajemne wykluczanie – w systemie musi istnieć pewien niepodzielny zasób

przetrzymywanie i oczekiwanie – w systemie musi istnieć proces, który ma przydzielony co najmniej jedne zasób i czeka na przydział innego zasobu przetrzymywanego przez inny proces

brak wywłaszczeń – zasoby nie podlegają wywłaszczeniu, co oznacza, że zasób może być zwolniony tylko z inicjatywy przetrzymującego go procesu

czekanie cykliczne – musi istnieć zbiór procesów takich, że proces P0 czeka na zasób przetrzymywany przez proces P1, P1 czeka na zasób przetrzymywany przez P2, ...,Pn czeka na zasób przetrzymywany przez P0.

**Postępowanie z zakleszczeniami**

Zasadniczo są trzy metody postępowania z zakleszczeniami:

nie pozwolić procesom nigdy wejść w stan zakleszczenia poprzez niespełnienie jednego lub więcej warunków koniecznych do powstania zakleszczenia,

wykrycie stanu zakleszczenia i usunięcie,

zlekceważenie problemu – system UNIX, Windows.

**Zapobieganie zakleszczeniom**

Wzajemne wykluczanie - w ogólnym przypadku nie jest możliwe unikanie zakleszczenia poprzez wykluczenie warunku wzajemnego wykluczania ponieważ niektóre zasoby są z natury niepodzielne.

Przetrzymywanie i oczekiwanie - aby wyeliminować warunek przetrzymywania i oczekiwania możemy:

przydzielać procesowi wszystkie zasoby zanim rozpocznie działanie,

pozwolić procesowi na zamawianie zasobów tylko wówczas gdy proces nie ma żadnych innych zasobów.

Brak wywłaszczeń – niespełnienie warunku braku wywłaszczeń można zrealizować w ten sposób, że w momencie kiedy proces zgłasza zapotrzebowanie na zasób, który nie może być natychmiast przydzielony, wówczas proces ten traci wszystkie dotychczasowe zasoby.

Czekanie cykliczne - jednym ze sposobów zagwarantowania, że czekanie cykliczne nigdy nie nastąpi, jest ponumerowanie wszystkich zasobów i wymaganie aby każdy proces zamawiał zasoby we wzrastającym porządku ich numeracji.

**Unikanie zakleszczeń** osiąga się poprzez zapamiętanie przez system pewnych dodatkowych informacji o działaniu procesów. Załóżmy, że proces deklaruje maksymalną ilość zasobów każdego typu jakich będzie używał. Informacja taka umożliwia zbudowanie algorytmu *unikania zakleszczenia* poprzez dynamiczne sprawdzanie stanu przydziału zasobów, tak żeby zagwarantować, że nie dojdzie do czekania cyklicznego.

**Wykrycie zakleszczenia**

W przypadku gdy system dysponuje zasobami zaprezentowanymi jednokrotnie zakleszczenie można wykryć poprzez zastosowanie grafu oczekiwania. Graf oczekiwania jest modyfikacją grafu przydziału zasobów i pokazuje tylko stany oczekiwania pomiędzy procesami. W przypadku gdy zasoby mogą być reprezentowane wielokrotnie stosuje się bardziej złożone algorytmy.

**usunięcie zakleszczenia** przez system operacyjny może polegać na:

zaniechaniu wszystkich procesów uwikłanych w zakleszczenie – metoda skuteczna lecz ponoszony koszt dość duży,

usuwanie procesów pojedynczo aż do zlikwidowania zakleszczenia – spory nakład na ponowne wykonywanie algorytmu wykrywania stanu zakleszczenia,

stopniowe wywłaszczanie zasobów – stopniowe zabieranie zasobów jednemu procesowi i przedzielanie innemu aż do przerwania pętli zakleszczenia.

**Wiązanie adresów** – proces ustalania adresów którymi się posługuje program.

Wiązanie adresów może zostać wykonane w następujących etapach przetwarzania programu użytkownika:

czas kompilacji – kod bezwzględny (np. pliki typu \*.com - 100 hex)

czas ładowania – kod przemieszczalny

czas wykonywania – specjalny sprzęt

**Ładowanie dynamiczne**

Program zostaje podzielony na podprogramy. Podprogramy nie są wprowadzane do pamięci dopóki nie są używane.

Wywołanie podprogramu polega na sprawdzeniu czy dany podprogram znajduje się już w pamięci i jeżeli go tam nie ma, to trzeba wywołać program łączący i ładujący moduły przemieszczalne.

Nie wymaga wsparcia ze strony systemu operacyjnego.

**Konsolidacja dynamiczna**

Koncepcja łączenia (konsolidacji) dynamicznego jest podobna do ładowania dynamicznego. Zamiast opóźnienia ładowania aż do czasu wykonania, opóźnia się łączenie.

W przypadku łączenia dynamicznego, w kodzie binarnym (wykonywalnym) programu w miejscach odwołań do podprogramów bibliotecznych znajdują się tylko zakładki. Zakładka jest małym fragmentem kodu, który mówi jak odnaleźć odpowiedni, rezydujący w pamięci, podprogram biblioteczny. Wykonanie zakładki powoduje zastąpienie jej przez adres podprogramu bibliotecznego. W ten sposób przy następnym napotkaniu danego fragmentu kodu, podprogram biblioteczny będzie wykonany bezpośrednio, bez ponoszenia kosztów na dynamiczne łączenie.

Metoda wymaga wsparcia przez system operacyjny.

**Biblioteki ładowane dynamicznie**

W systemie Windows biblioteki mogą być ładowane na dwa sposoby:

niejawne ładowanie biblioteki dll,

jawne ładowanie biblioteki dll.

Niejawne ładowanie biblioteki dll polega na użyciu w programie funkcji eksportowanych przez bibliotekę dll. Program ładujący (loader) ładując proces przegląda sekcję importu procesu i ładuje (oraz konsoliduje) użyte w programie biblioteki dll.

Jawne ładowanie biblioteki dll polega na wywołaniu funkcji systemu *LoadLibrary* i załadowaniu odpowiedniej biblioteki do pamięci procesu. Następnie aby określić adres funkcji lub procedury używamy funkcji *GetProcAdress*.

**Nakładki**

metoda, która umożliwia uruchomienie programów większych niż rozmiar aktualnej dostępnej pamięci operacyjnej. W metodzie tej Program podzielony zostaje na bloki. Programista jest odpowiedzialny za odpowiednią wymianę bloków pamięci.

Metoda wymaga dużego zaangażowania programisty i jest stosowana tam gdzie nie ma możliwości na stosowanie bardziej zaawansowanych mechanizmów.

**Pamięć wirtualna**

jest techniką umożliwiającą wykonywanie procesów, pomimo że nie są one w całości przechowywane w pamięci operacyjnej. Umożliwia to uruchamianie procesów, których zapotrzebowanie na pamięć jest większe niż aktualnie dostępna w systemie pamięć, a w szczególnych przypadkach uruchamianie procesów, których zapotrzebowanie na pamięć jest większe niż cała dostępna w systemie pamięć.

**Adres logiczny** – jest to adres wytworzony przez procesor, często nazywa się go adresem wirtualnym.

**Adres fizyczny** - jest to adres, który jest przekazywany do układów pamięci.

Odwzorowanie adresów wirtualnych na fizyczne dokonywane jest przez jednostkę zarządzania pamięcią MMU.

Mechanizmy translacji adresu wirtualnego

stronicowanie – podział pamięci i przydzielanie pamięci blokami o jednakowym rozmiarze (Strony mają stały rozmiar, co wpływa na łatwość ich wymiany. Fragmentacja wewnętrzna – marnowanie pamięci wskutek przydziału pamięci stałymi blokami. Duży rozmiar tablicy stron)

segmentacja – podział i przydzielanie pamięci blokami o różnych rozmiarach (Zmienny rozmiar segmentu powoduje trudności podczas wymiany – bardziej złożone mechanizmy wymiany Fragmentacja zewnętrzna – powstające obszary pamięci „dziury” za małe żeby w nie wpasować większość segmentów)

segmentacja stronicowana – połączenie wyżej wymienionych metod (Przydział następuje segmentami, które są zbiorami stron. Przydział pamięci do procesu następuje segmentami)

**Strategie zarządzania pamięcią**

strategie pobrania – decyzje kiedy powinno nastąpić załadowanie informacji do pamięci głównej

strategie przydziału – reguły i algorytmy wpasowania bloków informacji w wolne obszary pamięci głównej

strategie wymiany – reguły i algorytmy usuwania informacji z pamięci głównej

**Przydział pamięci**

przydział partycją stałą - rozmiar pamięci przydzielonej procesowi nie zmienia się podczas „życia” procesu,

przydział partycją zmienną - ilość przydzielonej procesowi pamięci może się zmieniać podczas „życia” procesu

Zauważmy, że przydział partycją stałą nie jest możliwy przy segmentacji – rozmiary segmentów mogą być (zazwyczaj są) różne.

Algorytmy wymiany dla stałych partycji

Losowy używana wyłącznie w środowiskach gdzie lokalność jest niewielka, np.: bazy danych

FIFO – First In First Out – kolejka bloków do wymiany jest umieszczana zgodnie z kolejnością umieszczania w pamięci, nie uwzględnia się częstości użycia bloków najdawniej alokowany może być najczęściej wykorzystywany

FINUFO – First In Not Used First Out – każda pozycja w kolejce ma dodatkowo znacznik używalności, kolejka przesuwa się cyklicznie początkiem jest wskaźnik ostatnio wymienionego bloku

LRU Last Recently Used – wymienia się blok najdawniej używany, niezbędna jest indeksacja kolejności referencji.

LFU Last Frequently Used – wymienia się blok najrzadziej używany.

Strategia MIN jako strategia odniesienia

**Plik**

logiczna jednostka magazynowania informacji lub inaczej nazwany zbiór powiązanych sobą informacji, zapisanych w pamięci pomocniczej.

Atrybut - „zewnętrzna cecha danej rzeczy lub osoby wyróżniająca ją spośród innych; istotna cecha, właściwość, przymiot czegoś”

Atrybuty pliku - mi. nazwa, typ, położenie, rozmiar, ochrona, czas i data, właściciel.

**Struktura katalogowa**

„Katalog można uważać za tablicę symboli tłumaczącą nazwy plików na ich wpisy katalogowe”

Przykładowe informacje o pliku przechowywane w katalogu:

nazwa pliku, typ pliku, lokalizacja, rozmiar, bieżące położenie, ochrona – kontrola dostępu, czas, data i identyfikator użytkownika