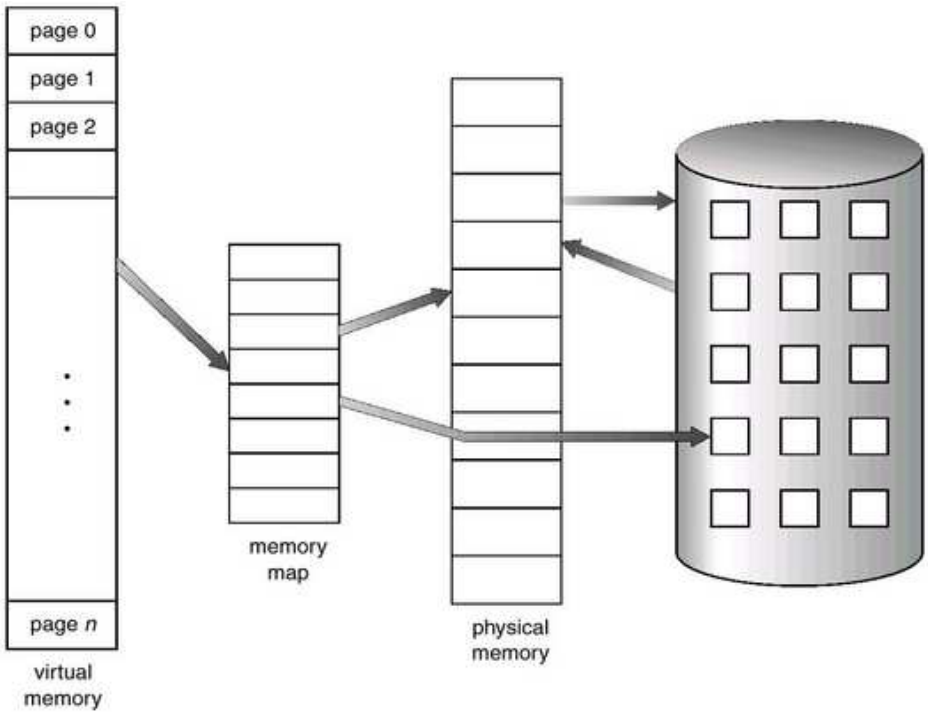
Pamięć wirtualna

Wstęp

- Rozdzielenie logicznej przestrzeni adresowej użytkownika i pamięci fizycznej.
- W trakcie wykonywania w pamięci może znajdować się tylko fragment programu.
- Logiczna przestrzeń adresowa może być większa niż rozmiar pamięci fizycznej.
- Pozwala na współdzielenie pamięci przez różne procesy.
- Umożliwia efektywniejsze tworzenie nowych procesów.

Implementacja możliwa jest przez:

- stronicowanie na żądanie,
- segmentację na żądanie.

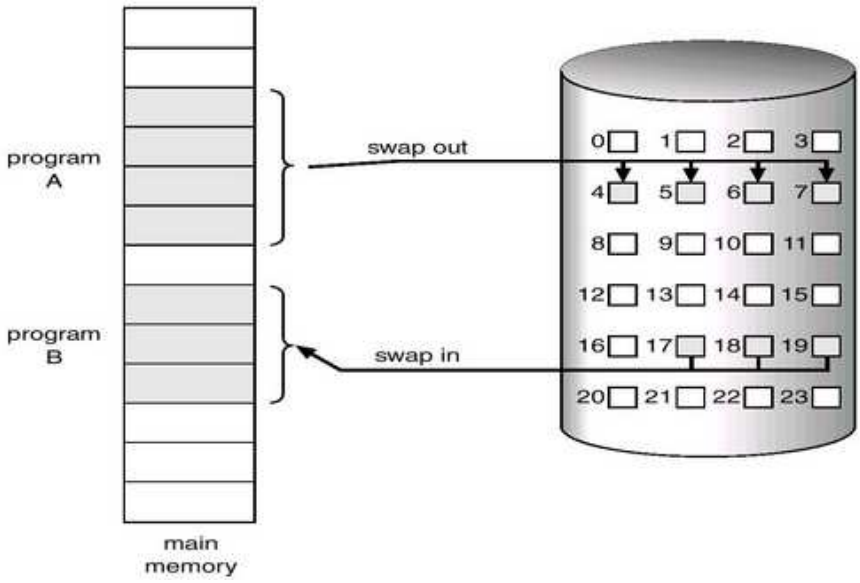


Stronicowanie na żądanie

Sprowadzamy stronę do pamięci wtedy, gdy jest potrzebna.

- Mniej we/wy na sprowadzanie programów.
- Szybsze wykonanie i szybszy czas reakcji.
- Mniejsze zapotrzebowanie na pamięć.
- Można obsłużyć większą liczbę użytkowników.

Mechanizm podobny do stronicowania z wymianą, ale sprowadzamy tylko część stron i tylko te, które są potrzebne.



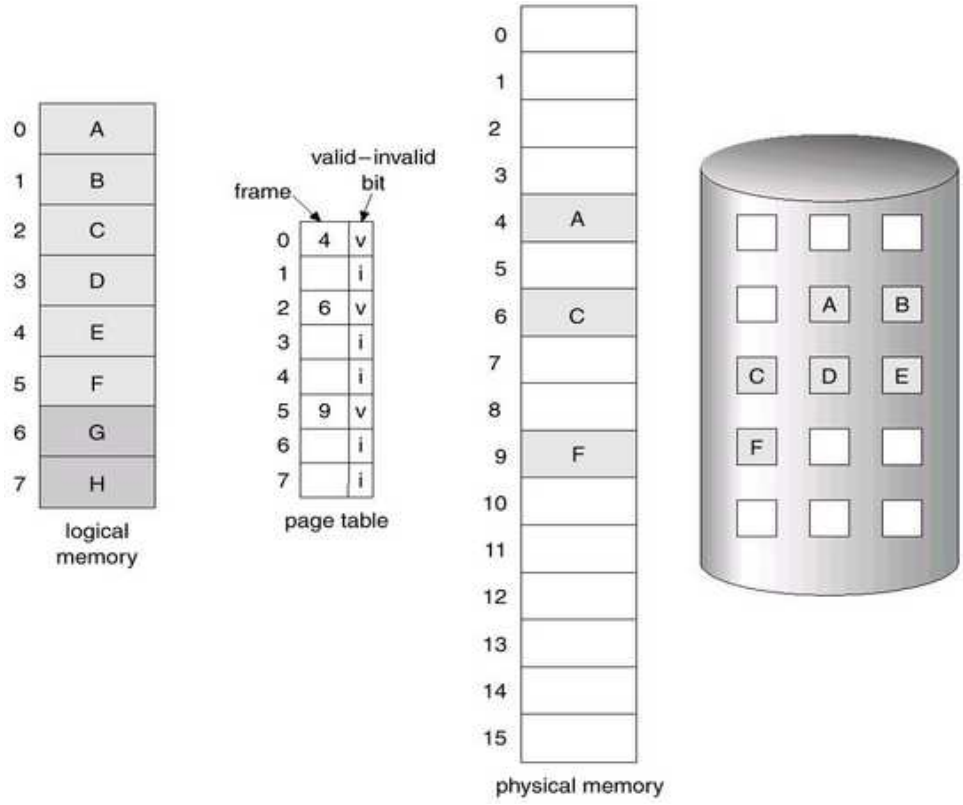
Strona jest potrzebna, tzn. było do niej odwołanie.

- jeśli odwołanie niepoprawne, zatrzymanie programu,
- jeśli strony nie ma w pamięci, sprowadzamy ją.

Konieczny specjalny mechanizm sprzętowy, np. bit poprawności związany z każdą pozycją tablicy stron.

Znaczenie bitu nieco inne niż przy stronicowaniu.

- ustawiony na 1 — strona w pamięci,
- ustawiony na 0 — odwołanie niepoprawne albo strona poza pamięcią.



Dla stron poza pamięcią tablica zawiera adres dyskowy.

Dla odwołań niepoprawnych, odpowiedni znacznik.

Odwołanie do strony z bitem poprawności ustawionym na zero jest wykrywane przez sprzęt stronicujący i traktowane jako pułapka, tak zwany błąd braku strony (page fault).

Postępowanie w takim przypadku obejmuje:

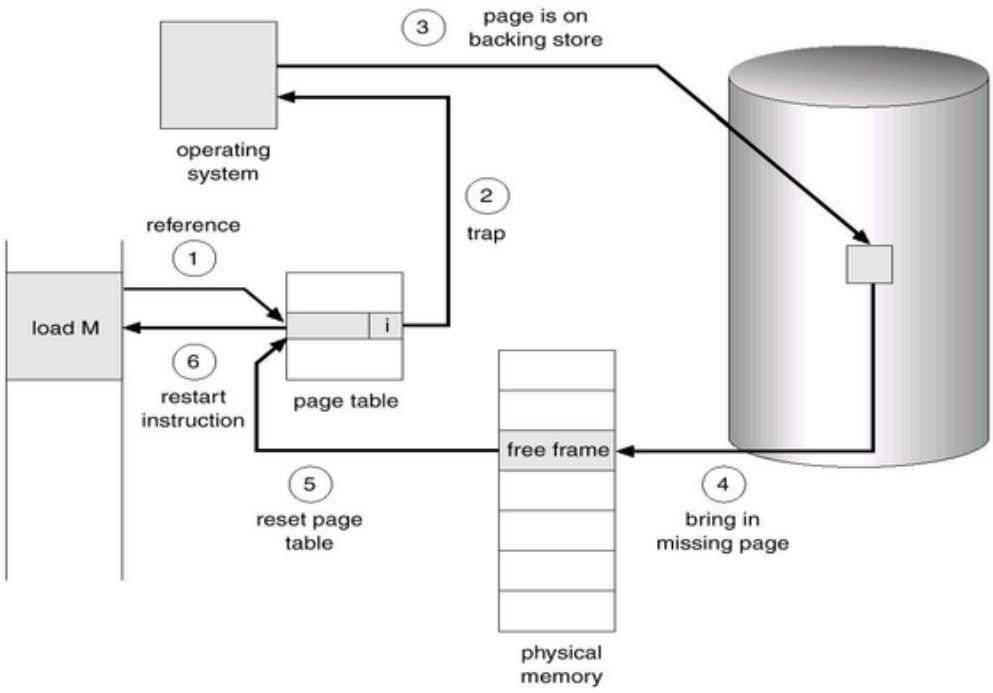
1. Sprawdzenie (tablica w PCB), czy odwołanie było nielegalne, czy dotyczyło strony poza pamięcią.
2. Jeśli odwołanie było nielegalne, kończymy proces. Wpp sprowadzamy stronę do pamięci.
3. Znajdujemy wolną ramkę.

4. Zlecamy przeczytanie strony z dysku do tej ramki.
5. Aktualizujemy zapis w tablicy stron.
6. Wznawiamy wykonanie przerwanej instrukcji.

Taki mechanizm pozwala na wykonywanie programu, którego żaden fragment początkowo nie znajduje się w pamięci.

Wznawianie przerwanej instrukcji może stwarzać poważne trudności:

- Np. instrukcja powodująca przesunięcie bloku bajtów tak, że obszar wyjściowy i docelowy zachodzą na siebie. Błąd może powstać po częściowym przesunięciu bloku a dane w bloku wyjściowym mogły zostać częściowo zniszczone.
- Pewne specjalne tryby adresowania z automatyczną zmianą zawartości rejestru.



A co się dzieje, gdy lista wolnych ramek jest pusta?

Pusta lista wolnych ramek $->$ zastępowanie

- Trzeba znaleźć w pamięci stronę nieużywaną i wykorzystać zajmowaną przez nią ramkę.
- Interesuje nas algorytm, przy którym liczba błędów braku strony będzie możliwie mała.
- W praktyce pewne strony mogą wędrować między pamięcią i dyskiem wielokrotnie.

Sprawność stronicowania na żądanie

- Współczynnik błędu braku strony: $0 <= p <= 1.0$.
  - $p = 0$ brak błędów,
  - $p = 1$ każde odwołanie powoduje błąd.

$$EAT = (1 - p) * czas\_dostpu\_do\_pamieci + p * czas\_obslugi\_bledu$$

Przy czym $czas\_obsugi\_bdu$ zawiera m.in. czas czytania strony z dysku.

oraz (ewentualnie) czas zapisu strony usuwanej.

Przykład:

- czas dostępu do pamięci $= 1\mu s$
- współczynnik błędu braku strony $= p$
- połowa wymienianych stron była modyfikowana
- czas odczytu/zapisu strony $= 10ms = 10000\mu s$

$$EAT = (1 - p) * 1 + p(0.5 * 10000 + 0.5 * 20000)$$

Copy on Write

Ta technika pozwala na współdzielenie stron przez rodzica i potomka.

Strony są kopiowane dopiero w momencie, gdy któryś z procesów je modyfikuje.

Technika pozwala na efektywniejsze tworzenie procesów.

Mapowanie plików do pamięci

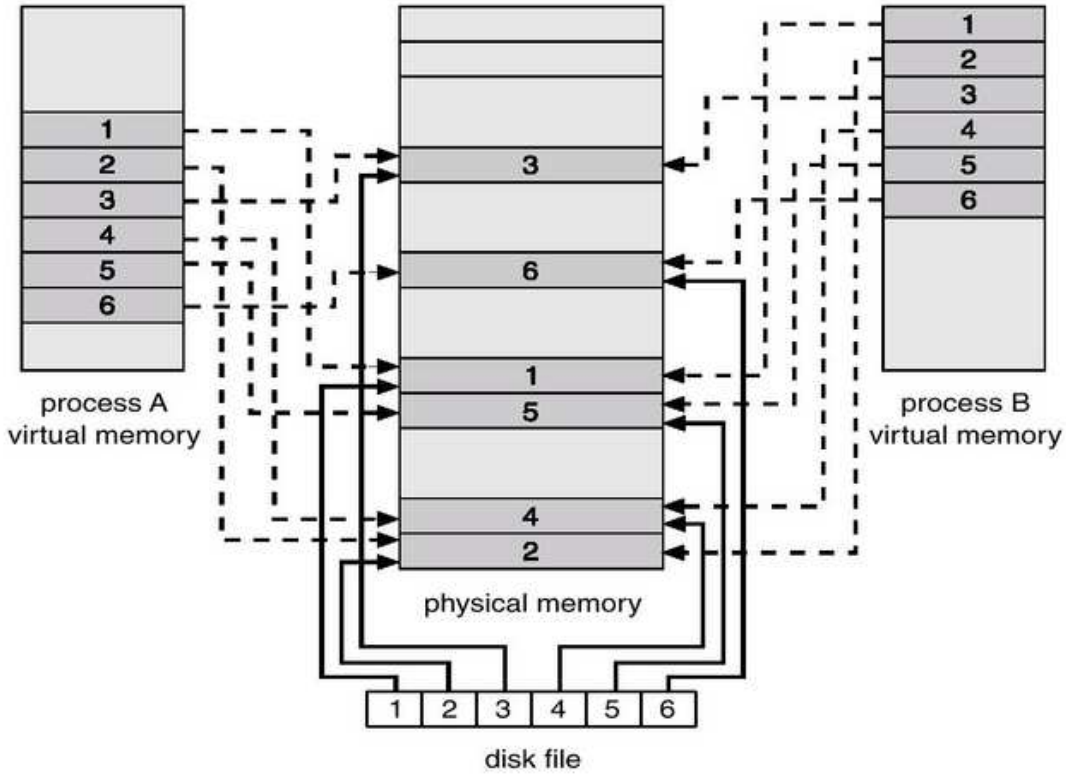
Bloki dyskowe są mapowane do stron pamięci.

Inicjalnie plik jest wczytywany do pamięci techniką stronicowania na żądanie.

Porcje wielkości strony umieszczane są w pamięci fizycznej.

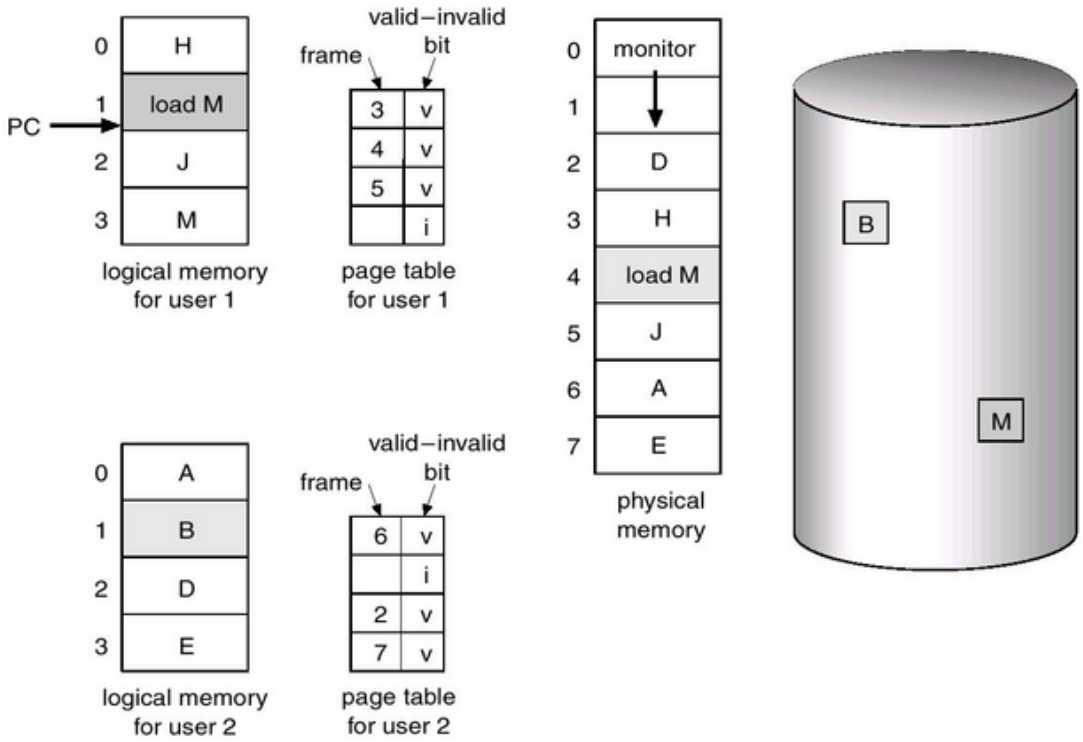
Operacje we/wy mogą być zastąpione zwykłymi odwołaniami do pamięci.

Wiele procesów może współdzielić mapowany plik.



Zastępowanie stron

W poniższym przykładzie cała pamięć fizyczna jest zajęta. W chwili odwołania do strony `M` system musi zapewnić miejsce dla tej strony.



Przez modyfikację algorytmu obsługi błędu braku strony zapobiegamy nadmiernym przydziałom ramek.

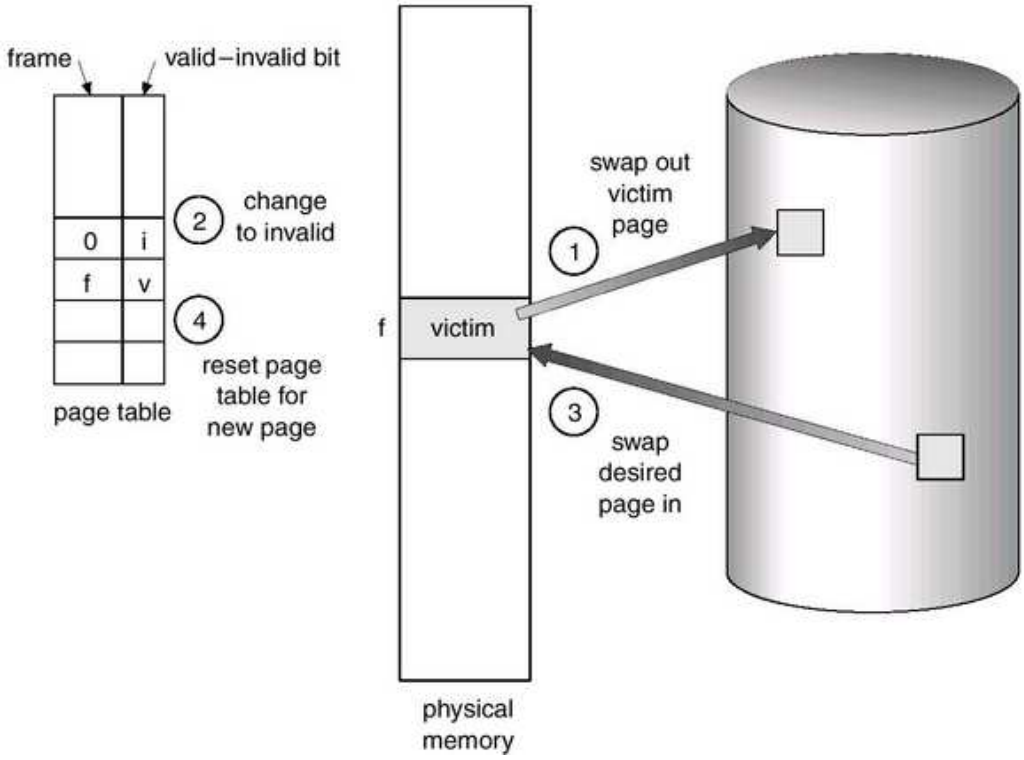
Użycie bitu zabrudzenia strony pozwala na uniknięcie niepotrzebnego zapisywania stron, które nie zostały zmienione.

Zastępowanie ostatecznie rozdziela logiczną i fizyczną przestrzeń adresową — pamięć fizyczna może być mniejsza od logicznej.

Podstawowy schemat zastępowania.

- Znajdź żądaną stronę na dysku.
- Znajdź wolną ramkę.
  - Jeśli są wolne ramki, użyj jednej z nich.

  - Jeśli nie ma wolnej ramki, użyj algorytmu zastępowania do wytypowania ’ofiary’, której ramka zostanie wykorzystana.
- Wczytaj z dysku stronę do znalezionej wolnej ramki. Zmodyfikuj tablicę stron i listę ramek.
- Wznów przerwany proces.



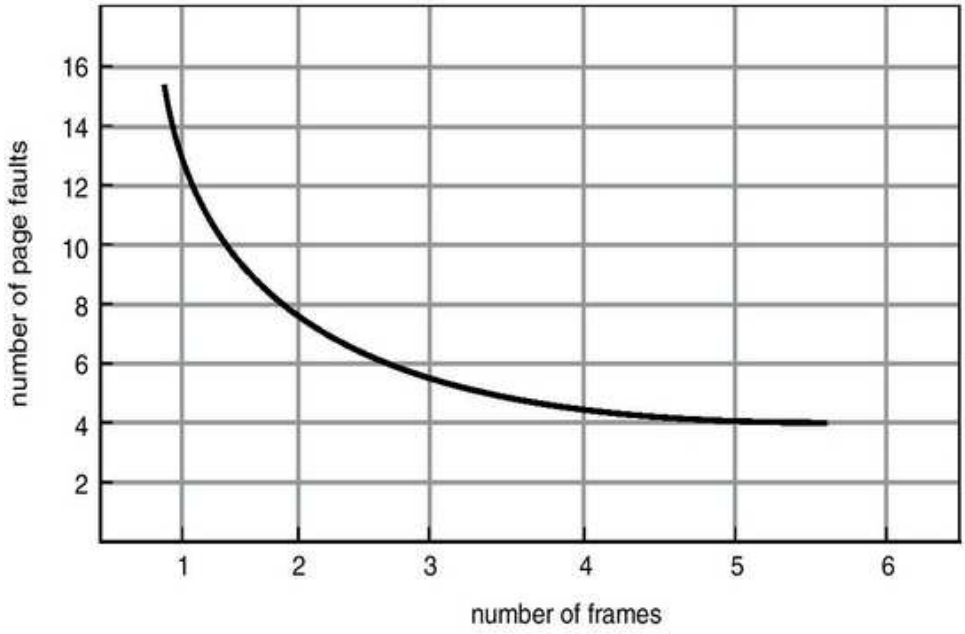
Algorytmy zastępowania stron

Istnieje ich wiele.

Podstawowy cel, minimalizacja liczby błędów braku strony.

Zwykle będziemy używać pamięci wielkości 3 ramek, inicjalnie pustej.

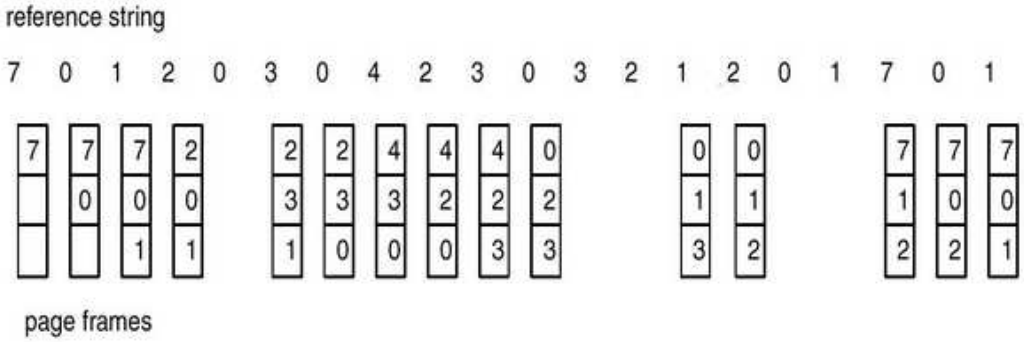
Poniższy diagram ilustruje ogólną zależność miedzy wielkością pamięci i liczbą błędów braku strony.



Algorytm FIFO

Algorytm polega na wybieraniu do usunięcia strony, która najdłużej znajduje się w pamięci.

W poniższym przykładzie miało miejsce 15 błędów braku strony.

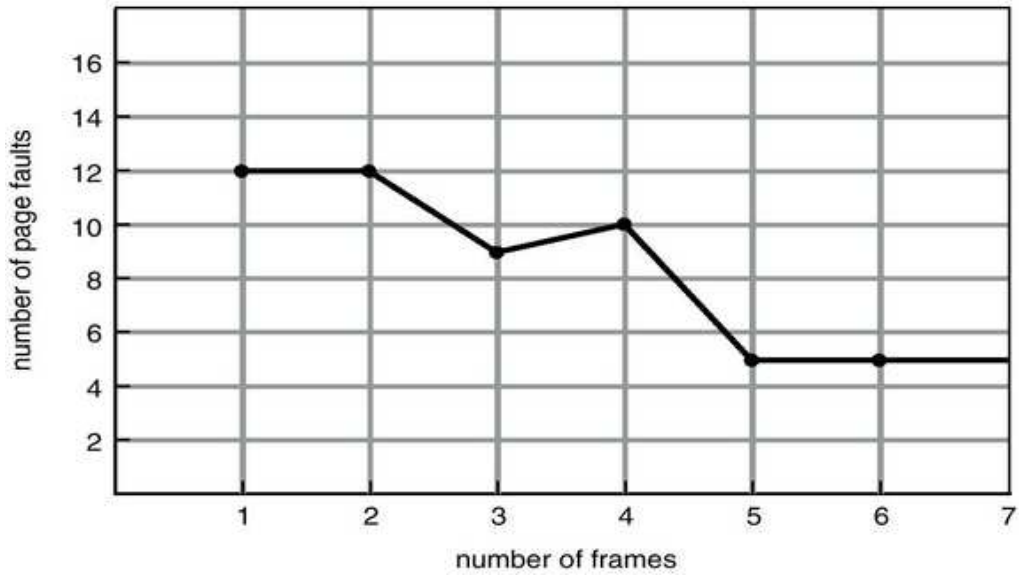


Rozważmy jeszcze jeden ciąg odwołań: `1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5`.

Nietrudno sprawdzić, że

- przy 3 ramkach mamy 9 błędów,

- przy 4 ramkach mamy 10 (!) błędów.

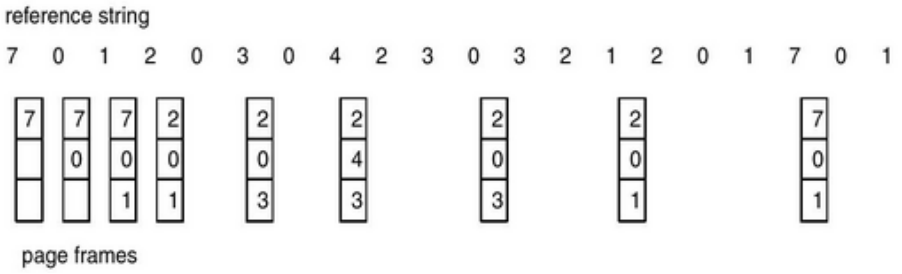


Własność ta nazywa się anomalią Belady'ego.

Algorytm optymalny

Algorytm optymalny polega na wybraniu do usunięcia strony, która najdłużej **nie będzie** używana.

W poniższym przykładzie wystąpiło 9 błędów braku strony.



Dla ciągu `1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5` i 4 ramek mamy 6 błędów (sprawdź!).

Niestety algorytm ten nie daje się zaimplementować, ponieważ wymaga wiedzy o przyszłych odwołaniach (czyli w praktyce wykonania programu).

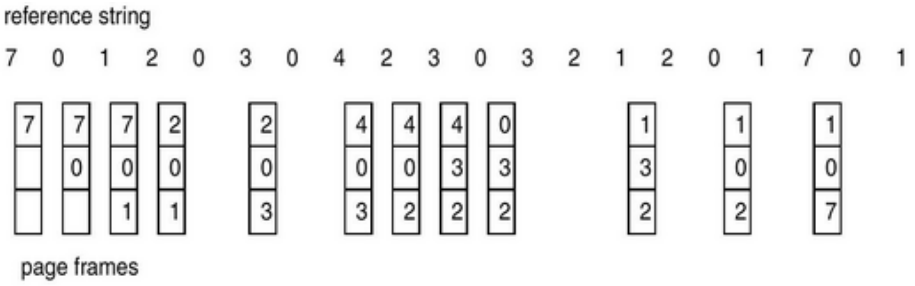
Używany jest głównie jako punkt odniesienia przy ocenie innych algorytmów.

Algorytm LRU

Algorytm używa do oszacowania zachowania w przyszłości wiedzy o przeszłości.

Polega on na wybraniu do usunięcia strony, do której odwołanie nastąpiło najdawniej (least recently used).

W poniższym przykładzie wystąpiło 12 błędów braku strony.



Dla ciągu `1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5` i 4 ramek mamy 8 błędów (znowu sprawdź!).

Ani algorytm optymalny ani LRU nie są obarczone anomalią Belady’ego.

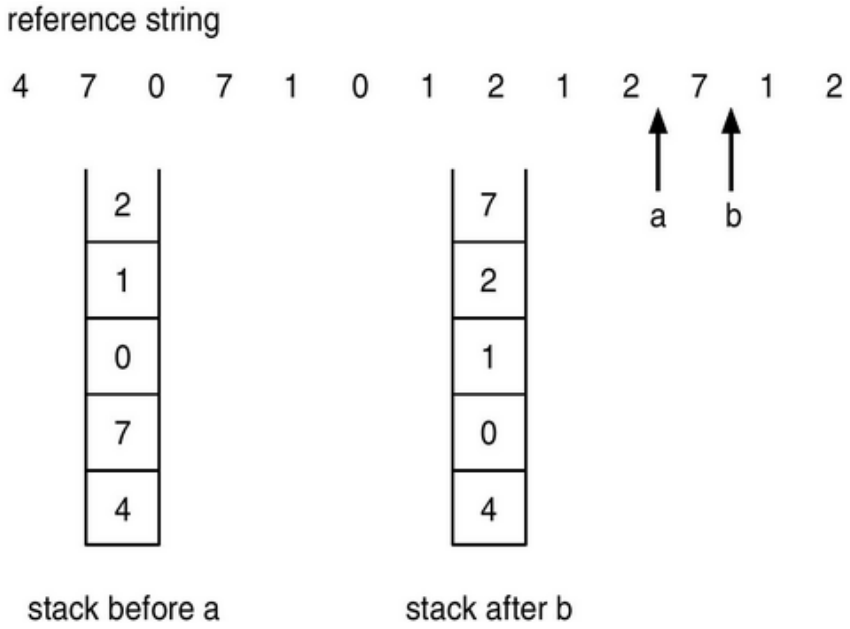
Anomalia ta nie dotyka tak zwanych algorytmów stosowych.

Algorytm nazywamy stosowym jeśli potrafimy pokazać, że zbiór stron w pamięci dla n ramek jest zawsze podzbiorem zbioru stron w pamięci przy n+1 ramkach.

Implementacja LRU

Implementacja tego algorytmu wymaga odpowiedniego sprzętu. Realizacja programowa wielokrotnie spowalnia dostęp do pamięci.

- Licznik czasu, rejestr czasu użycia przy każdym odwołaniu, przeszukiwanie tablicy stron.
- Stos z numerami stron do których były odwołania.
  - Wyjęcie numeru i umieszczenie na wierzchu stosu - modyfikacja 6 wskaźników.
  - Wybór ’ofiary’ nie wymaga przeszukiwania.



Przybliżanie metody LRU

Bity odniesienia

Z każdą stroną związany jest dodatkowy bit, początkowo ustawiony na 0.

W chwili odwołania do strony bit przyjmuje wartość 1.

Do zastąpienia wybierana jest strona z bitem o wartości 0.

Bardziej rozbudowany wariant — dla każdej strony dodatkowy rejestr do magazynowania wartości bitu odniesienia w ustalonych odstępach czasu.

Algorytm drugiej szansy

Wymaga bitu odniesienia.

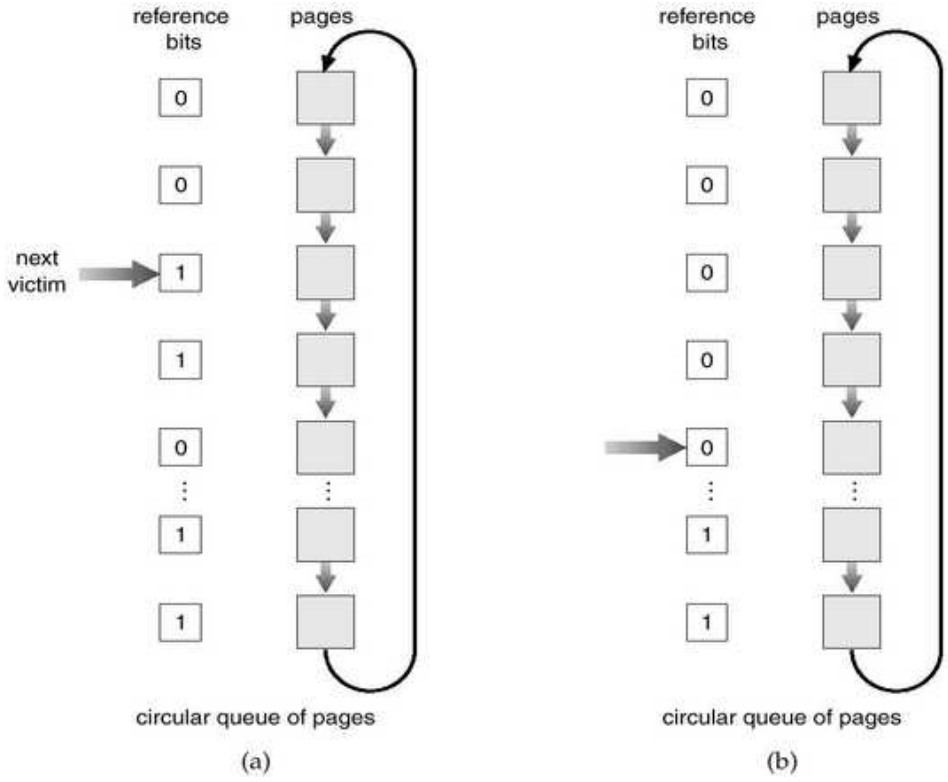
Strony wybierane są cyklicznie.

Jeśli bit odniesienia kandydatki ma wartość 1, jest ustawiany na 0 i strona zostaje w pamięci.

Do usunięcia wybierana jest strona z bitem odniesienia równym 0.

Możliwy wariant rozbudowany:

- poza bitem odniesienia, rozważamy bit modyfikacji,
- uzyskujemy 4 klasy stron: (0,0), (0,1), (1,0), (1,1),
- bit odniesienia traktujemy tak jak w algorytmie drugiej szansy,
- wybieramy stronę typu (0,0) albo ostatecznie (0,1).



Algorytmy zliczajace

Utrzymywany jest licznik zliczający odwołania do danej strony.

LFU — usuwamy stronę z najmniejszą liczbą odwołań.

MFU — usuwamy stronę z dużą wartością licznika (ta z małą pewnie będzie jeszcze używana).

Przydział ramek

Jak dzielić wolne ramki między różne procesy?

Czy utrzymywać pewną pulę wolnych ramek?

Co z ramkami dla nowych procesów

Minimalna liczba wolnych ramek

Określa ją architektura komputera.

IBM370 - rozkaz przesyłania ciągu znaków z pamięci do pamięci:

- sam rozkaz - 6 bajtów, czyli dwie ramki,
- argument skąd - 2 ramki,
- argument dokąd - 2 ramki

Faktycznie mogą dojść jeszcze 2 ramki: powyższy rozkaz może być argumentem rozkazu `execute` (2 ramki).

Algorytmy przydziału

- stały przydział ramek - wszystkim po równo,

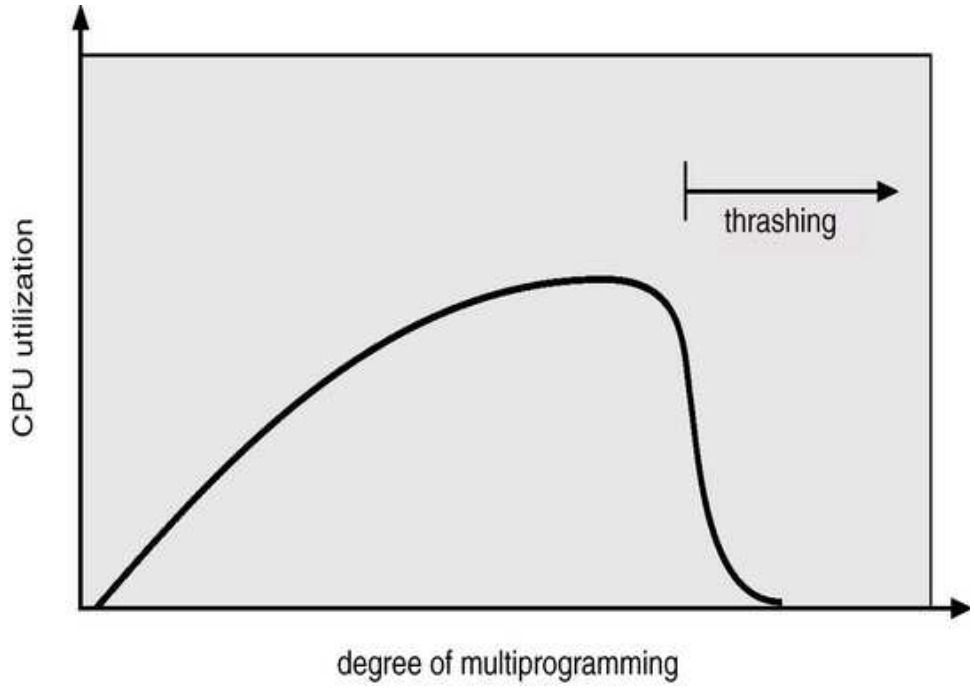
- przydział proporcjonalny, zależny od wielkości,
- przydział proporcjonalny, zależny od priorytetu.

Przydział lokalny vs. globalny

Lokalny — każdy korzysta tylko ze swojej puli ramek.

Globalny — strony do zastąpienia wybiera się z globalnej puli stron. Umożliwia to skorzystanie z ’cudzej’ ramki.

Szamotanie (migotanie)



Jeśli proces nie dysponuje ’wystarczającą’ liczbą ramek, współczynnik błędów braku strony staje się b. duży.

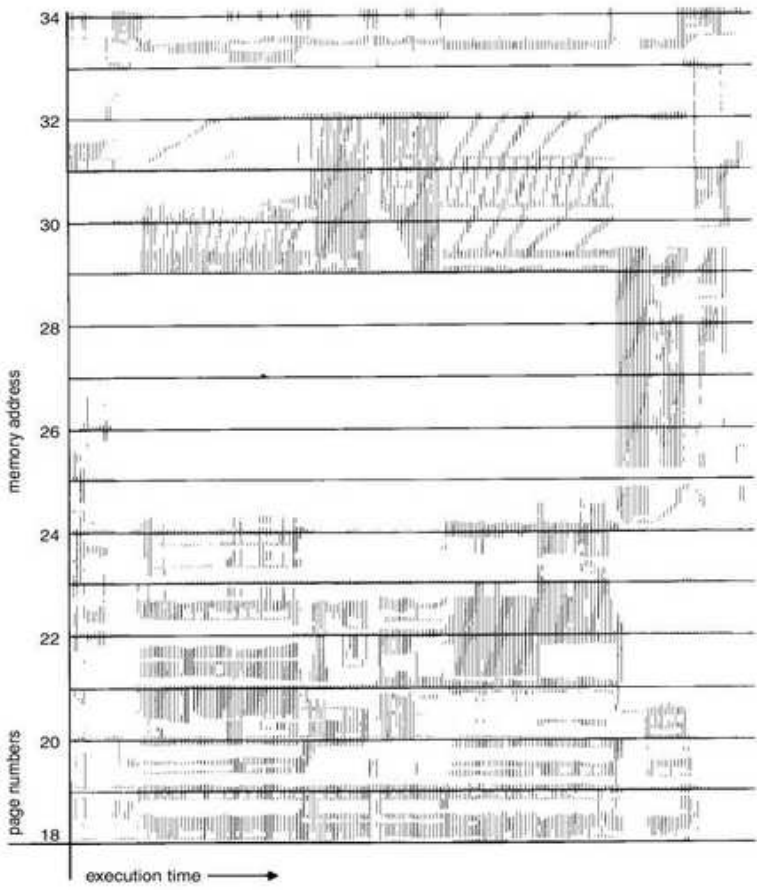
Prowadzi to do:

- słabego wykorzystania CPU,

- w konsekwencji może to spowodować zwiększenie stopnia wieloprogramowości,

Szamotanie — sytuacja, kiedy proces poświęca znaczącą część czasu na wymianę stron.

Model strefowy (locality model)



W różnych momentach wykonania proces aktywnie wykorzystuje tylko pewien podzbiór swoich stron. Taki podzbiór nosi nazwę strefy (locality).

W trakcie wykonywania proces przechodzi z jednej strefy do drugiej (mogą one zachodzić na siebie).

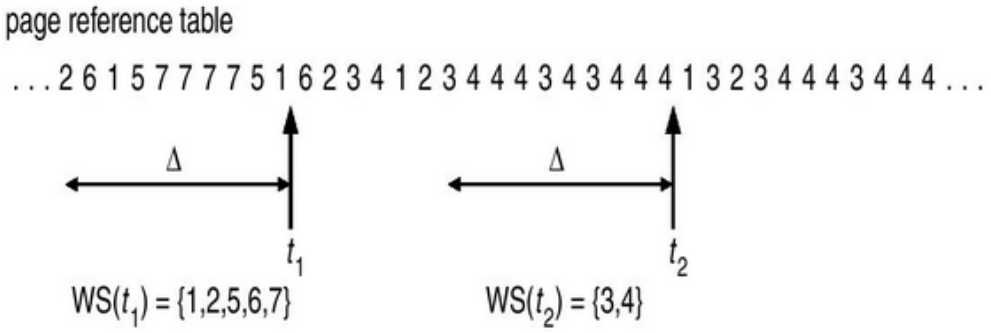
Jeśli liczba przydzielonych procesowi ramek jest mniejsza niż rozmiar strefy wzrasta liczba błędów braku strony.

Jeśli suma wymiarów stref procesów jest większa niż rozmiar pamięci, występuje szamotanie.

Model zbioru roboczego

- Δ — okno zbioru roboczego — ustalona liczba odwołań do stron.
- WSS_j — rozmiar okna zbioru roboczego procesu P_j — liczba stron, do których odwołał się proces w ciągu Δ ostatnich odwołań.
  - Δ za małe — zbiór roboczy nie obejmuje całej strefy,
  - Δ za duże — zbiór roboczy obejmuje więcej niż jedną strefę,
  - $\Delta = \infty$ — zbiór roboczy obejmuje cały program.
- $D = \sum WSS_j$ — łączna liczba potrzebnych ramek.
- Jeśli $D > m$ — szamotanie.
- Konieczne wstrzymanie (co najmniej jednego) procesu.

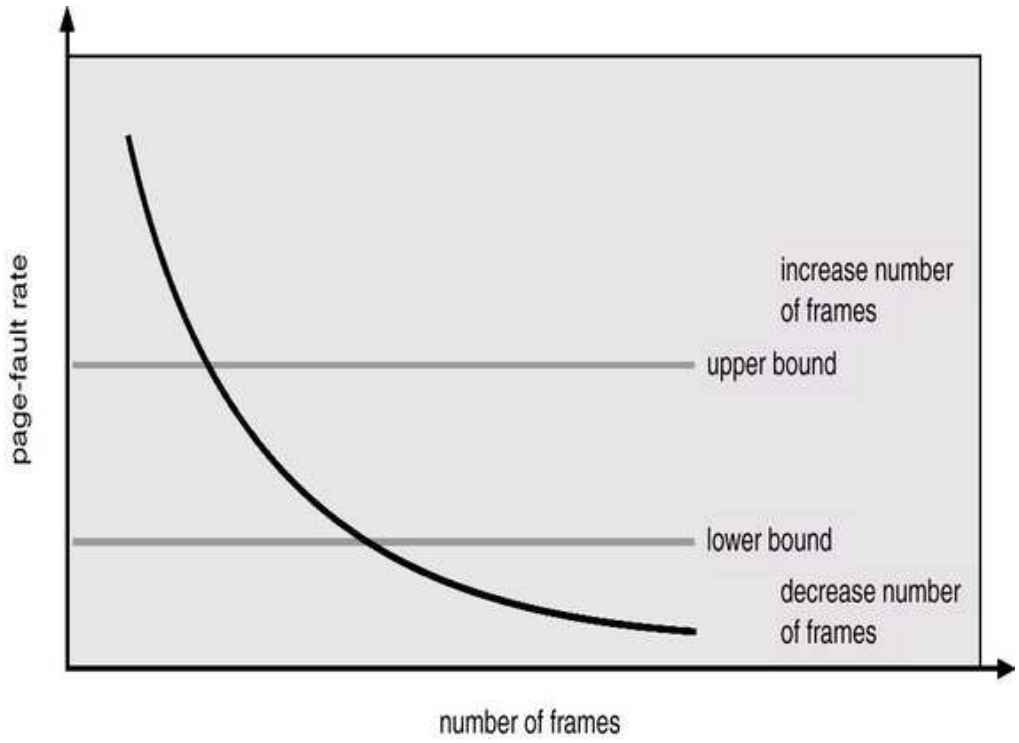
Model kłopotliwy do implementacji — np. poprzez śledzenie stanu bitów odniesienia w ustalonych odcinkach czasu.



Częstość błędów braku strony

Ustanawiamy ’akceptowalny’ współczynnik błędów braku strony.

- obserwowany za mały — odbieramy ramki,
- obserwowany za duży — przydzielamy ramki.



Zagadnienia dodatkowe

Stronicowanie wstępne

Czy sprowadzać do pamięci strony na początku pracy procesu i po jego powrocie z urządzenia wymiany.

Wybór wielkości strony

- fragmentacja,
- wielkość tablicy stron,
- koszty transmisji,

- strefy.

Generalnie zaznacza się tendencja do zwiększania stron.

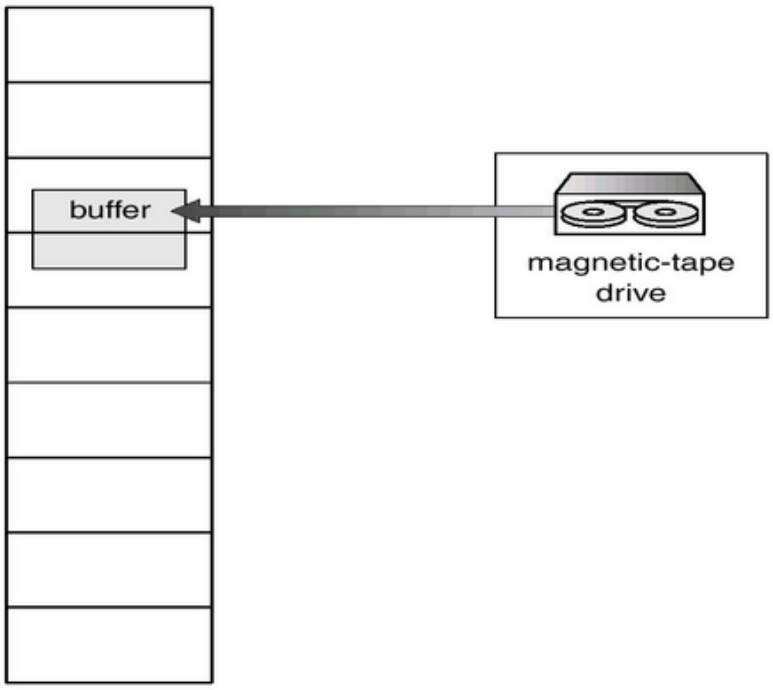
Struktura programu

W zasadzie stronicowanie na żądanie jest przezroczyste dla programisty. Wiedząc jednak jak w systemie jest ono zorganizowane, można wpłynąć na liczbę błędów braku strony.

Przyład — program zerujący duża tablicę kwadratową może to robić albo wierszami albo kolumnami.

Znając sposób rozkładu takiej tablicy na stronie, mamy wpływ na liczbę błędów braku strony.

Blokowanie stron a we/wy



Strony niekiedy muszą być zabezpieczone przed usunięciem z pamięci.

Dotyczy to np. operacji we/wy do bufora w pamięci użytkownika. Strona w oczekiwaniu na transmisję może zostać usunięta.

- we/wy przez bufory w pamięci systemu, albo
- blokowanie stron.

Inny przypadek, strony niskopriorytetowych procesów czekających na przydział procesora — zanim nowosprowadzona strona zostanie użyta, może stać się kandydatka do usunięcia.

Segmentacja na żądanie

Mechanizm znacznie mniej wydajny niż stronicowanie.

Stronicowanie wymaga znaczącego wsparcia sprzętowego.

Niekiedy, np Intel 80286, stronicowanie nie jest możliwe, ale segmentacja tak.

Konsekwencje niejednakowej wielkości segmentów.

Generalna zasada bardzo zbliżona do stronicowania.