Pamięć wirtualna

Sławomir Górawski

10.1. Idea

Oddzielenie pamięci fizycznej od logicznej, bazujące na założeniu, że w danej chwili tylko część pamięci procesu musi znajdować się w pamięci operacyjnej. Argumenty za tym przemawiające:

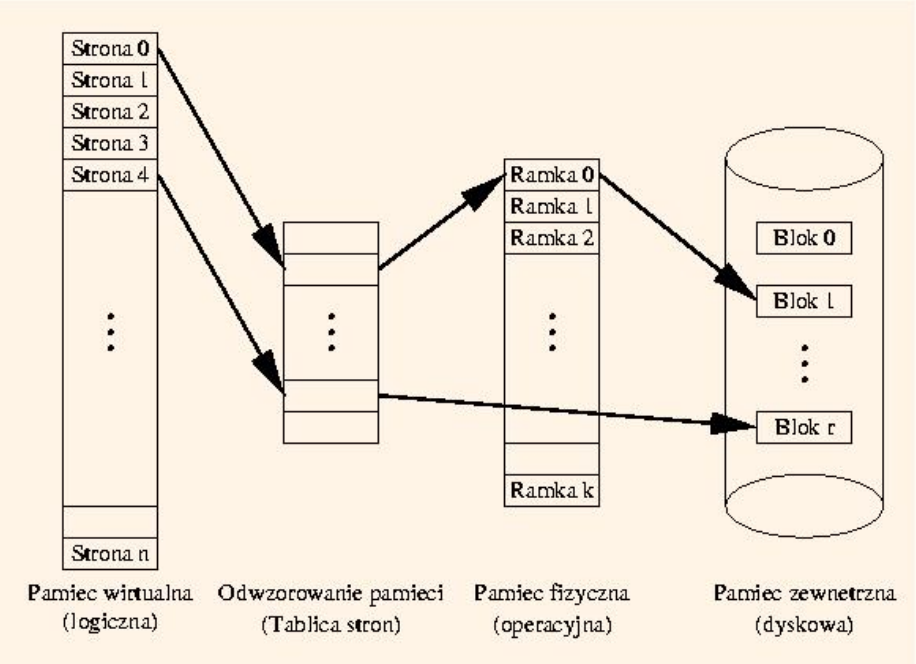
- W programach często istnieją rzadko wykonywane sekcje, np. przeznaczone do obsługi szczególnych błędów.
- Struktury danych często mają przydzielone więcej pamięci, niż rzeczywiście wykorzystują.
- Pewne możliwości programu mogą być rzadko wykorzystywane.
- Nawet jeśli cały program musi znaleźć się w pamięci operacyjnej, nie musi to mieć miejsca w tym samym czasie.

10.2. Zasada działania

Mechanizmy pamięci wirtualnej to rozszerzone mechanizmy **stronicowania** lub **segmentacji**. Nieużywane w danej chwili obszary pamięci logicznej procesu mogą być przeniesione na dysk. Ta pamięć drugorzędna jest nazywana **obszarem wymiany**. Zalety pamięci wirtualnej:

- Program nie jest ograniczony wielkością pamięci fizycznej, co upraszcza jego pisanie.
- Każdy program może zajmować mniej pamięci, a zatem można w tym samym czasie wykonywać więcej programów.
- Zmniejszenie liczby operacji wejścia-wyjścia wykorzystywanych do wymiany programów w pamięci.

Konstrukcja **MMU** (*Memory Management Unit*) musi odwzorowywać dany adres logiczny na adres fizyczny lub miejsce w obszarze wymiany. Służą do tego algorytmy **stronicowania na żądanie** lub segmentacji na żądanie (mniej elastyczny, wyparty przez poprzedni). W przypadku segmentacji na żądanie tablica segmentów tłumaczy numer segmentu na spójny obszar w pamięci fizycznej lub w obszarze wymiany. W stronicowaniu na żądanie numery stron są tłumaczone przez **tablicę stron** na numery ramek lub numery bloków dyskowych.



Rysunek 1. Schemat stronicowania na żądanie

10.3. Stronicowanie na żądanie

Strona jest sprowadzana do pamięci operacyjnej tylko wtedy, gdy jest potrzebna. Nazywa się to **procedurą leniwej wymiany** (ang. *lazy swapper*). Podczas ładowania procesu do pamięci procedura stronicująca zgaduje, które strony będą niezbędne. Podczas odwołania się procesu do pamięci mogą zajść trzy sytuacje:

- odwołanie jest niepoprawne,
- odwołanie jest poprawne i strona jest w pamięci,
- odwołanie jest poprawne i strony nie ma w pamięci.

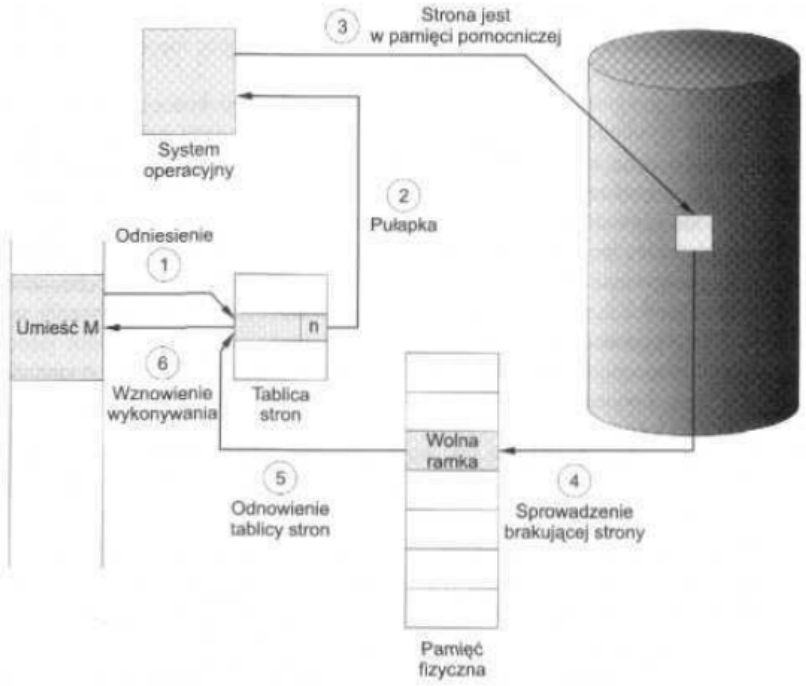
W pierwszym przypadku zlecenie jest odrzucane, w drugim jest po prostu obsługiwane, a w trzecim musi dodatkowo obejmować sprowadzanie żądanej strony z dysku do pamięci.

10.3.1. Bit poprawności

Mechanizm sprzętowy pozwalający stwierdzić, czy strona znajdujące się w pamięci operacyjnej. Znajduje się obok każdej pozycji w tablicy stron. Przy natrafieniu na 0 występuje przerwanie **braku strony** (ang. *page-fault*).

10.3.2. Obsługa braku strony

Może się zdarzyć, że nie ma wolnej ramki w pamięci. Należy wówczas znaleźć jakąś nieużywaną ramkę i przenieść ją na dysk. Tym zajmuje się algorytm zastępowania stron.



Rys. 9.4 Etapy obsługi braku strony

Rysunek 2. Etapy obsługi braku strony

10.4. Zastępowanie stron

W przypadku, gdy trzeba załadować stronę z dysku, a nie ma wolnej ramki, trzeba wybrać jedną z ramek z pamięci i zastąpić ją żądaną stroną. W tym celu trzeba stwierdzić, czy "ofiara" ma na dysku swoją wierną kopię – czy od czasu ostatniego sprowadzenia do pamięci była modyfikowana. Służy do tego sprzętowy **bit modyfikacji**. Jeżeli podczas zastępowania jest on równy 0, to "ofiary" nie trzeba zapisywać na dysku.

Do wyboru "ofiary" służą algorytmy, np.:

1. **FIFO** – ofiarą staje się strona, która najdłużej przebywa w pamięci.
2. **LRU** (ang. *Least Recently Used*) – ofiarą jest strona, która nie była używana przez najdłuższy okres.
3. **Algorytm drugiej szansy** – jak FIFO, tyle że każda ramka ma "dwa życia", tj. jeżeli po raz pierwszy została wybrana do wyrzucenia, to zamiast tego trafia na koniec kolejki.
4. **Algorytmy zliczające** – korzystają ze sprzętowych liczników odwołań do każdej strony; nie są popularne w praktyce:
   (a) **LFU** (ang. *Least Frequently Used*) – ofiarą staje się strona, do której było najmniej odwołań.
   (b) **MFU** (and. *Most Frequently Used*) – ofiarą staje się strona, do której było najwięcej odwołań.

10.5. Przydział ramek

Problem polegający na tym, ile ramek powinny dostać poszczególne procesy:

1. **Przydział lokalny** – każdy proces ma przydzieloną pulę ramek, z której wybierane są ramki do zastąpienia.
2. **Przydział globalny** – ofiara jest wybierana spośród wszystkich ramek w systemie.
3. **Przydział stały** – każdy proces ma na stałe przydzieloną liczbę ramek, o pozostałe procesy mogą dowolnie rywalizować.
4. **Przydział priorytetowy** – każdy proces ma przypisany priorytet, ramki do zastąpienia wybierane są z jego ramek oraz ramek procesów o niższym priorytecie.
5. **Zliczanie częstości braków stron** – dodatkowe ramki są przydzielane procesowi, u którego częstość braków stron jest wyższa niż pewna ustalona wielkość.

10.6. Szamotanie

Gdy proces ma mniej ramek niż liczba aktywnie używanych stron, dochodzi do **szamotania**. Ramek jest mniej niż trzeba, więc co chwilę należy sprowadzić nową stronę usuwając jedną z załadowanych stron. Ta usunięta strona jest za chwilę potrzebna i znów trzeba ją sprowadzić usuwając jakąś, która zaraz będzie znów potrzebna itd. System zaczyna się zajmować jedynie wymianą stron.

10.7. Zbiór roboczy

Jedna z metod zapobiegania szamotaniu; zbiór stron, do których nastąpiło odwołanie w ciągu ostatnich l instrukcji – ma przyblizać strefę, w której znajduje się program. Znając wielkości zbiorów roboczych, możemy przydzielić procesom pamięć proporcjonalnie do wielkości ich zbiorów roboczych. Jeśli suma rozmiarów zbiorów roboczych wszystkich procesów jest większa niż rozmiar dostępnej pamięci, prawdopodobnie mamy do czynienia z szamotaniem. Należy wówczas wstrzymać jeden z procesów, aby nie pogarszać sytuacji.