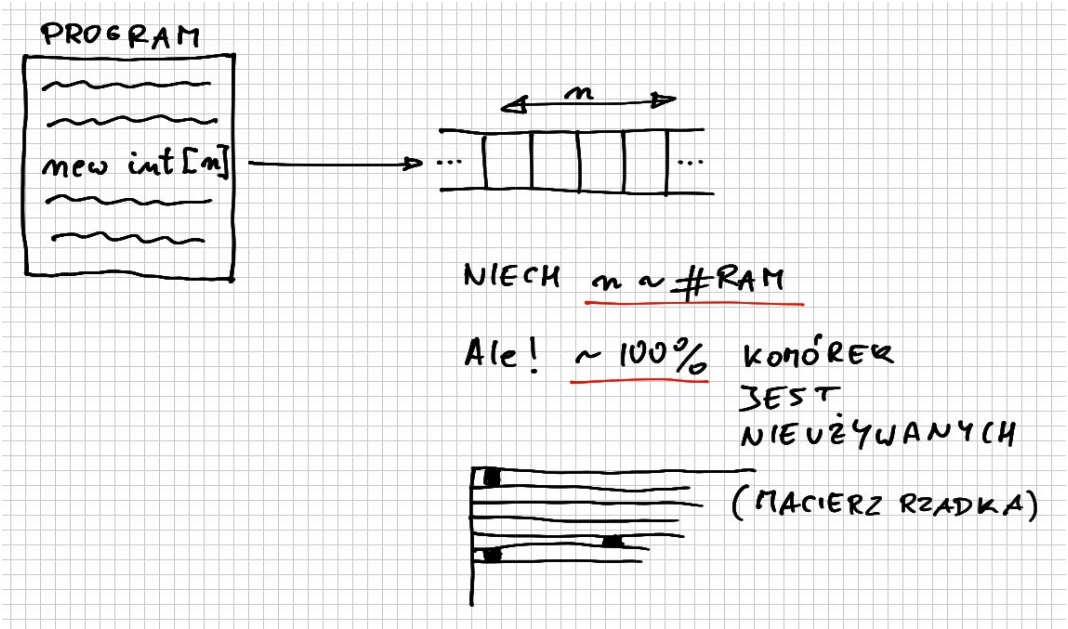
Systemy Operacyjne

Pamięć wirtualna

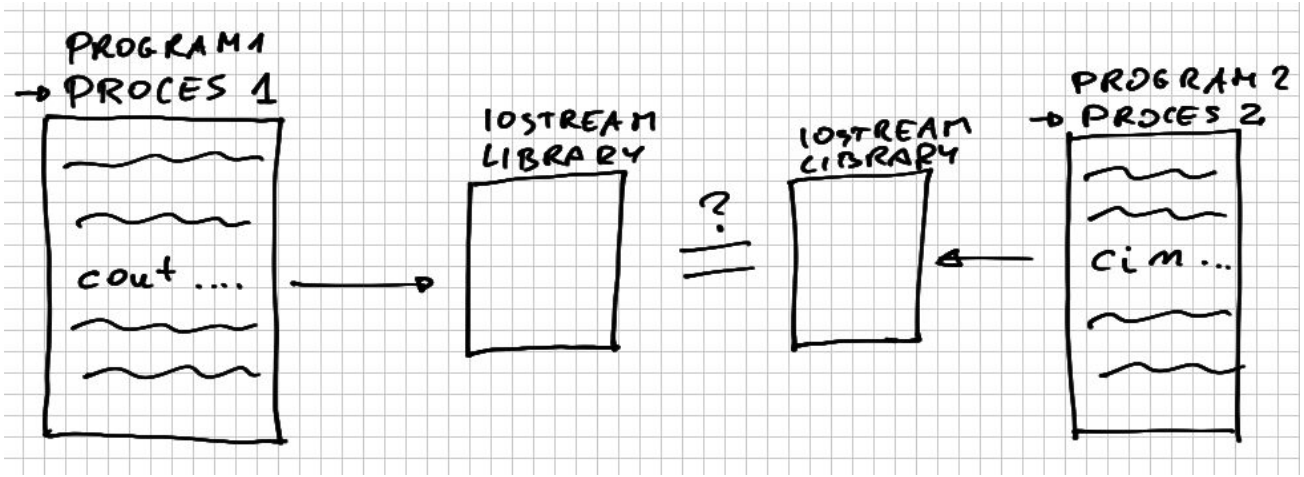
Dr hab. inż. Krzysztof Rzecki, prof. AGH

Na podstawie: Abraham Silberschatz, *Koncepcje systemów operacyjnych*

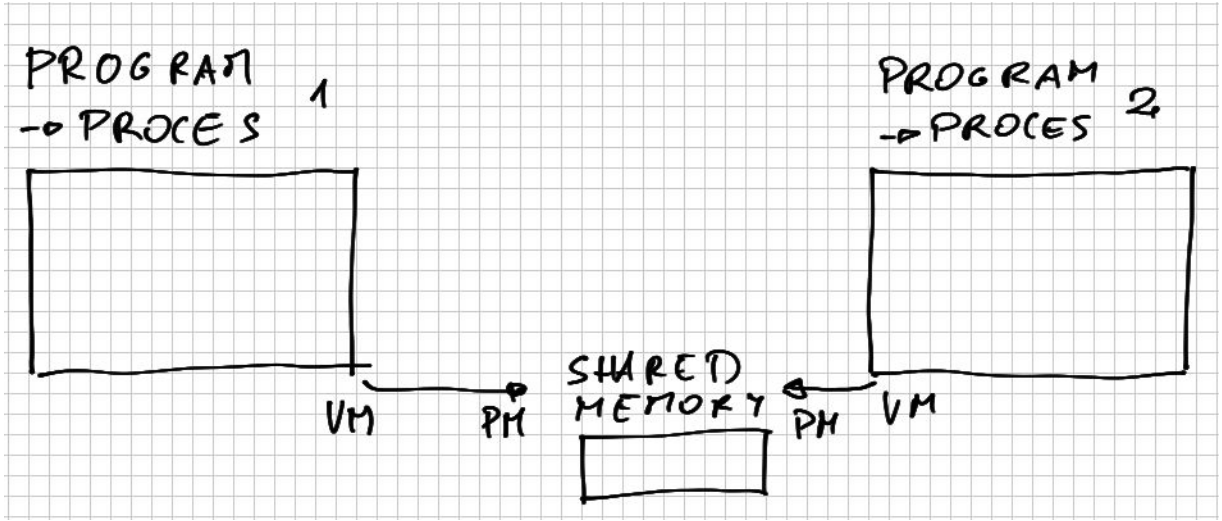
Przypadek 1


PROGRAM
new int[n]
n
NIECH n ~ #RAM
Ale! ~ 100% KOMÓREK JEST NIEUŻYWANYCH
(MACIERZ RZADKA)

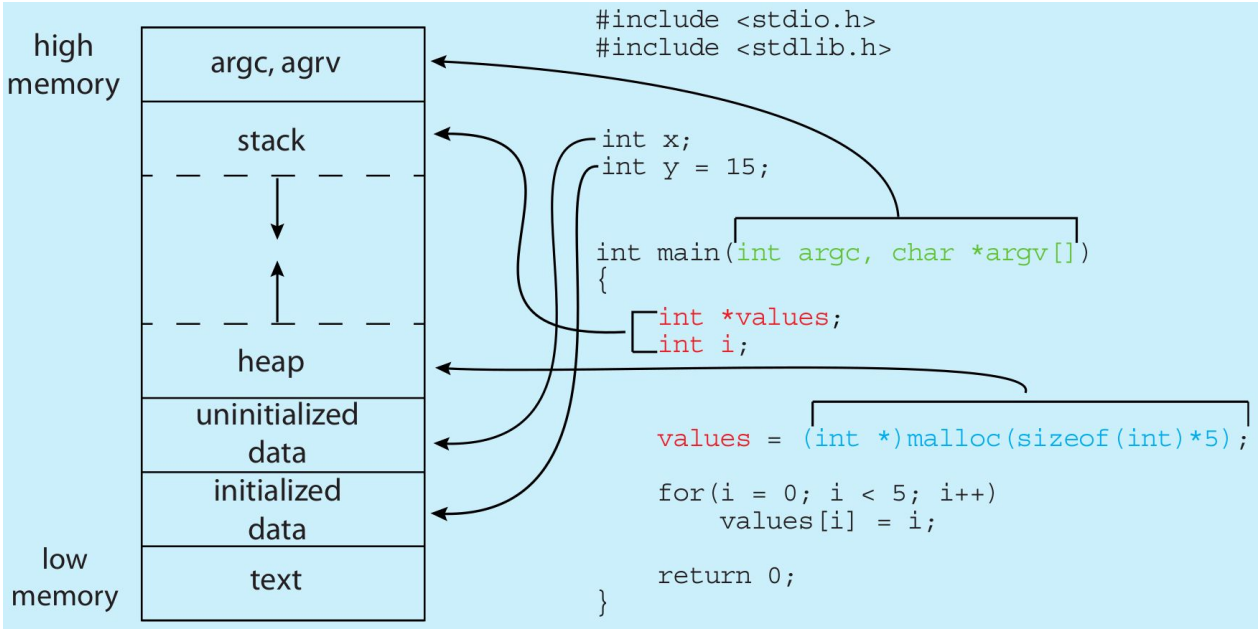
Przypadek 2


PROGRAM 1
PROCES 1
cout ....
IOSTREAM
LIBRARY
?
IOSTREAM
LIBRARY
PROGRAM 2
PROCES 2
cin ...

Przypadek 3



Program w C - przypomnienie



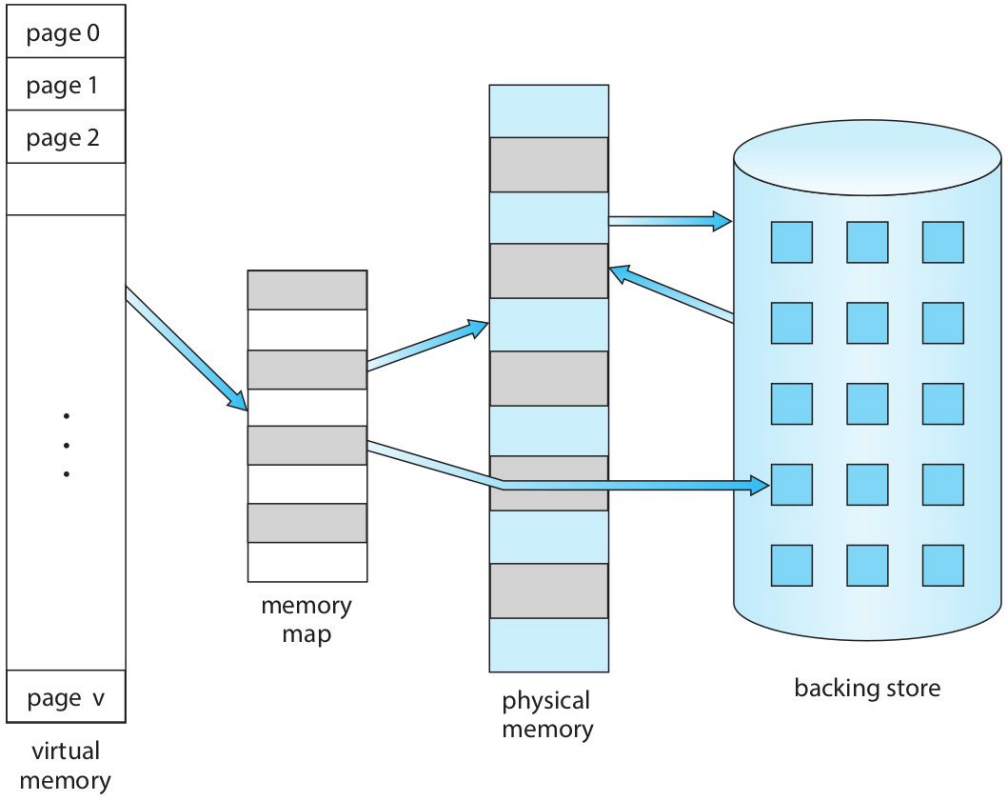
Źródło: A. Silberschatz, *Operating Systems Concepts Essentials*

Pamięć wirtualna

- Pamięć wirtualna to technika pozwalająca na wykonywanie procesów, które nie są całkowicie w pamięci (m.in. np. z powodu ich rozmiaru).
- Pamięć wirtualna jest abstraktem pamięci głównej jako bardzo dużej macierzy, oddzielając pamięć logiczną przeznaczoną dla programisty od pamięci fizycznej.
- Pamięć wirtualna pozwala procesom na współdzielenie plików, bibliotek oraz implementację pamięci dzielonej.
- Implementacja pamięci wirtualnej nie jest prosta, a nieostrożne korzystanie z niej wpływa znacząco na wydajność procesów.

Pamięć wirtualna

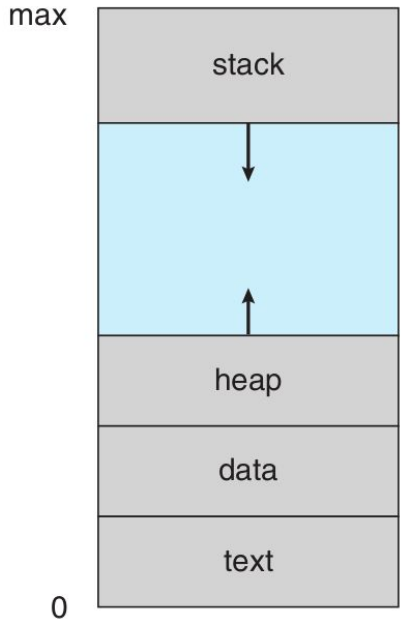
- Pamięć wirtualna oddzielona jest od pamięci fizycznej.
- Dzięki temu możliwe jest stosowanie olbrzymiej pamięci wirtualnej przy niewielkiej pamięci fizycznej.
- Programista nie musi zajmować się obsługą i ilością pamięci fizycznej.
- Pamięć zapasowa (ang. *backing store*, ang. *spaw space*) - przestrzeń poza pamięcią główną.



Źródło: A. Silberschatz, *Operating Systems Concepts Essentials*

Wirtualna przestrzeń adresowa

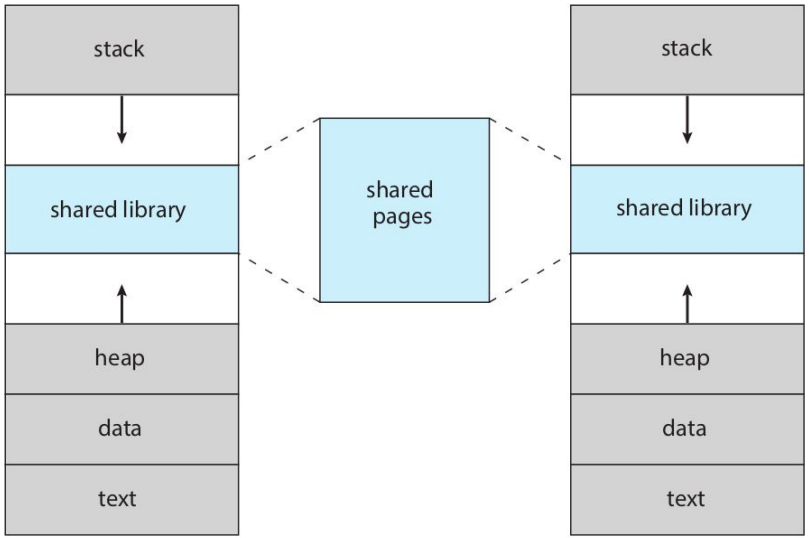
- Stos - ang. *Stack* - miejsce, w którym dane umieszczane są w sposób uporządkowany, w tym miejscu odkładane są dane dotyczące wywołań funkcji, zmiennych (~~statyczne w tym globalne~~, automatyczne w tym lokalne). Miejscem tym zarządza program.
- Sterta - ang. *Heap* - miejsce, w którym dane umieszczane są swobodnie wg zapotrzebowania, zwykle przez wywołanie malloc/new, są to zmienne (dynamiczne). Miejscem tym zarządza programista.
- Data - obszar o znanym w czasie kompilacji rozmiarze, w którym umieszczane są dane statyczne, w tym globalne.



Źródło: A. Silberschatz, *Operating Systems Concepts Essentials*

Współdzielenie przestrzeni pamięci fizycznej

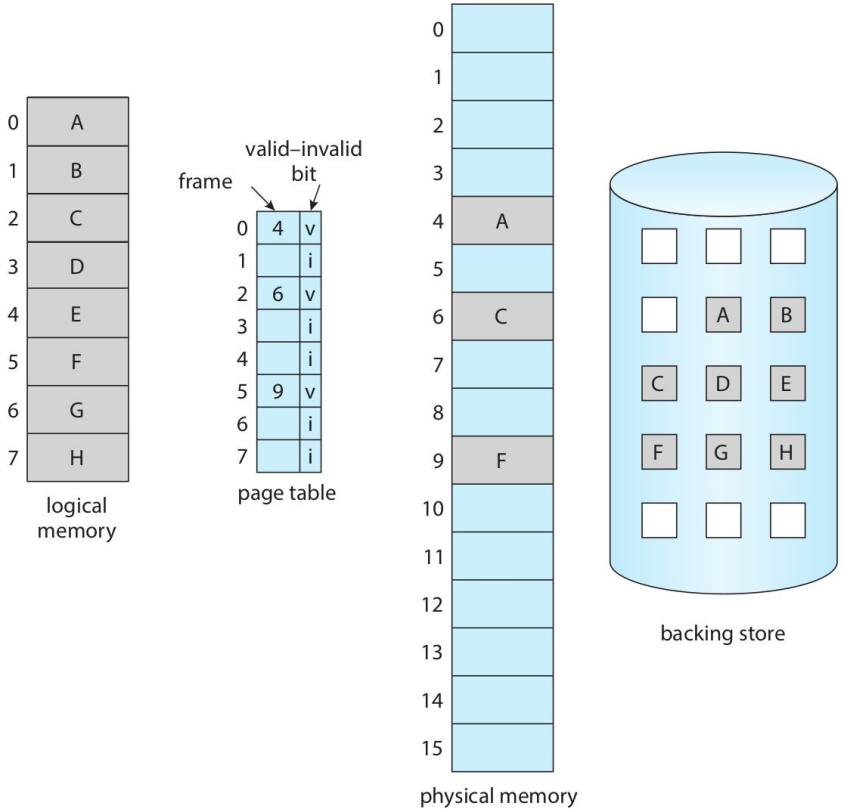
- Adresy wirtualne bibliotek systemowych mogą być mapowane z różnych procesów (w trybie tylko do odczytu) na wspólną część w przestrzeni adresów fizycznych.
- Procesy mogą współdzielić obszar pamięci (ang. *shared memory*) celem wymiany komunikatów i ją też muszą adresować w przestrzeni adresów wirtualnych.



Źródło: A. Silberschatz, *Operating Systems Concepts Essentials*

Stronicowanie na żądanie

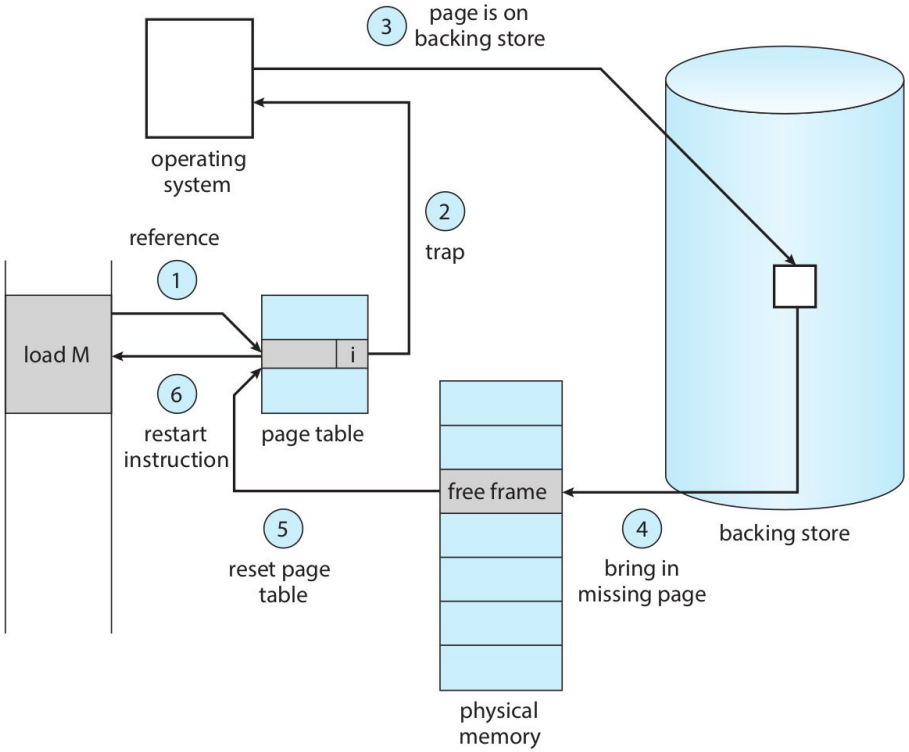
- Strona jest wprowadzana do pamięci wtedy, gdy jest potrzebna.
- Tablica stron zawiera dodatkowo bity poprawności odwołania:
  - 1 - strona znajduje się w pamięci głównej (ang. *valid*)
  - 0 - strona znajduje się poza pamięcią główną (ang. *invalid*)
- Odwołanie do strony z bitem == 0:
  - Błąd braku strony (ang. *page fault*)
  - Realizacja procedury z następnego slajdu.



Źródło: A. Silberschatz, *Operating Systems Concepts Essentials*

Obsługa błędu brak strony

1. Sprawdzenie tablicy stron (valid vs. invalid).
2. Jeśli referencja ma stan *invalid*, to:
   a. Brak strony w ogóle -> terminowanie procesu.
   b. Brak strony w pamięci głównej, ale jest w pamięci zapasowej, przejdź do (3).
3. Zgłoszenie do SO zapotrzebowania na wczytanie strony z pamięci zapasowej.
4. Odnajdywanie na liście wolnej ramki i wczytanie strony do ramki.
5. Aktualizacja tablicy stron.
6. Restart instrukcji, która wywołała błąd.



Źródło: A. Silberschatz, *Operating Systems Concepts Essentials*

Lista wolnych ramek (ang. *Free-Frame List*)

- W momencie wystąpienia błędu strony, system operacyjny musi dostarczyć daną stronę z pamięci zapasowej do pamięci głównej.
- W tym celu stosowana jest lista wolnych ramek:

- System operacyjny zwykle stosuje technikę ang. *zero-fill-on-demand*, która zeruje ramki przed użyciem (względy bezpieczeństwa).
- W momencie startu systemu, cała dostępna pamięć umieszczana jest na liście wolnych ramek.

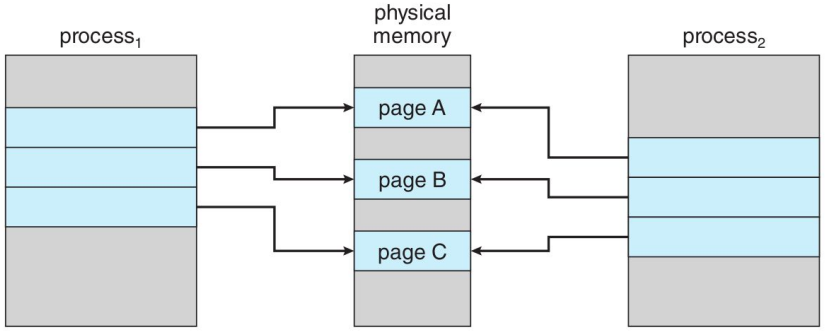
Przebieg stronicowania na żądanie

1. Odwołanie/przerwanie do systemu operacyjnego.
2. Zapisz stan rejestrów oraz procesu.
3. Stwierdź, że przerwanie wywołane było przez błąd strony.
4. Sprawdź, czy odwołanie jest właściwe i określ położenie strony w pamięci zapasowej.
5. Wywołaj odczyt z pamięci zapasowej do wolnej ramki:
    a. Czekaj w kolejce, aż żądanie odczytu zostanie obsłużone.
    b. Odczekaj czas działania urządzenia.
    c. Rozpocznij transfer strony do wolnej ramki.
6. Podczas oczekiwania, przekaż CPU innemu procesowi.
7. Przechwyć przerwanie z podsystemu I/O (zakończenie wczytywania).
8. Zapisz rejestry oraz stan innego procesu (jeśli krok 6 jest wykonany).
9. Określ, że przerwanie było z pamięci zapasowej.
10. Zaktualizuj tablicę stron, aby wskazać, że określona strona jest teraz w pamięci.
11. Czekaj, aż CPU zostanie przypisany znów temu procesowi.
12. Wznów rejestry, stan procesu oraz nową tablicę stron i wznów działanie procesu.

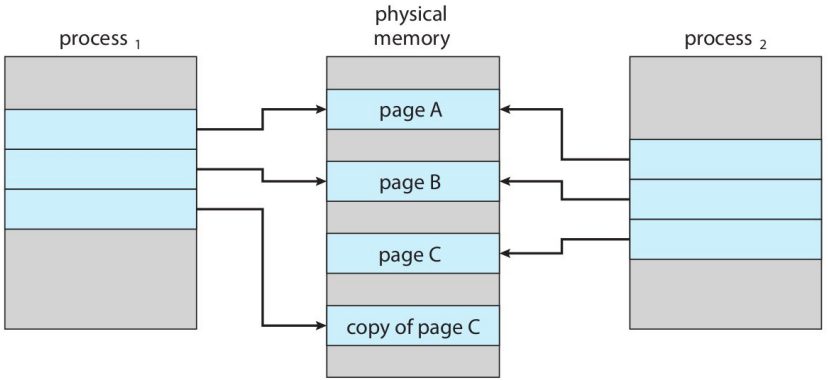
Czasochłonne!

Copy-on-Write

- COW w technice fork() pozwala na współdzielenie tych samych stron w pamięci.
- Tylko ta strona, która jest modyfikowana wymaga kopiowania.
- Często po fork() występuje exec() i wtedy okazuje się, że kopiowanie w ogóle nie jest potrzebne.



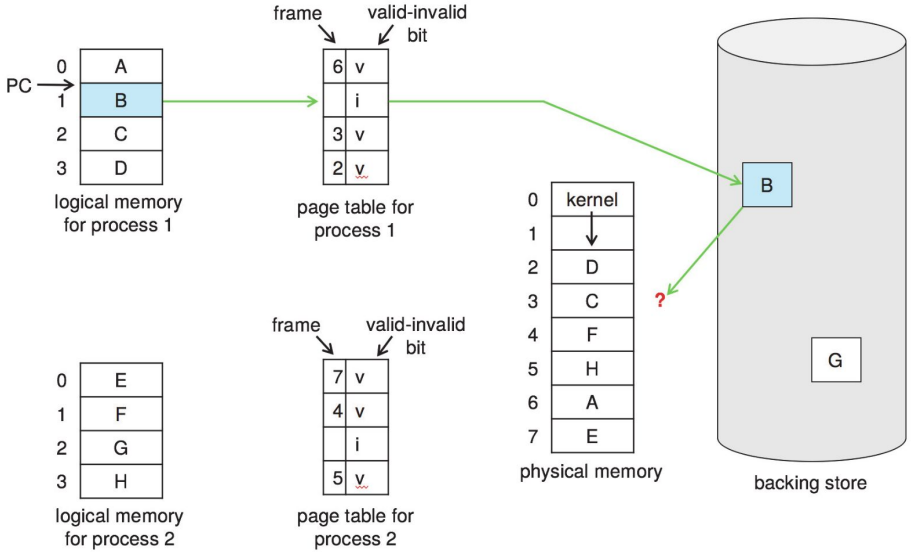
Modyfikacja strony C



Źródło: A. Silberschatz, *Operating Systems Concepts Essentials*

Over-allocation

- W trakcie wykonywania procesu, występuje błąd strony.
- System operacyjny odnajduje stronę w pamięci zapasowej, ale stwierdza, że nie ma wolnych ramek - cała pamięć jest zajęta.
- Jednym z rozwiązań jest terminowanie procesu… ale to nie jest rozwiązanie.



Źródło: A. Silberschatz, *Operating Systems Concepts Essentials*

Zastępowanie stron

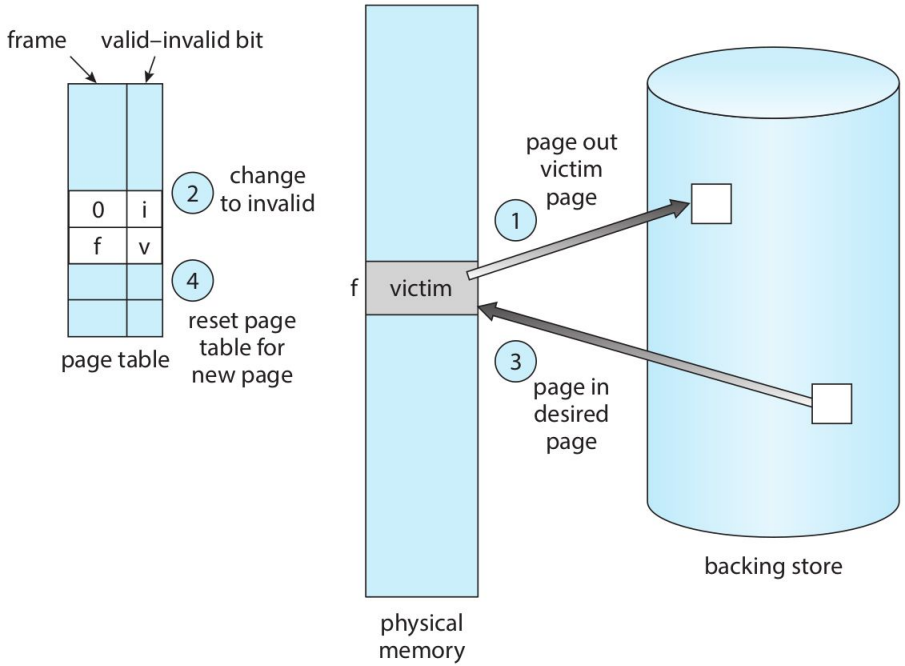
- Co kiedy wolne ramki się skończą ?
- Co w sytuacji, kiedy nie wszystkie strony są aktualnie w użyciu ?
- Jak zastępować strony nieużywane ?
- Czy ładować wszystkie strony, czy tylko przewidziane do użycia ?
- Co w sytuacji, kiedy odwołania do stron są w kilku miejscach programu ?
- Czy pozwalać na uruchomienie programów, dla których nie ma dostępnych ramek ?
- Jak wybierać ramki do zastąpienia ?
- Jak informować procesy o zastąpieniu ramki ?

Uwaga: adresacja logiczna = strony, adresacja fizyczna = ramki.

Zastępowanie stron

1. Znajdź stronę w pamięci zapasowej.
2. Szukaj wolnej ramki:
   - Jeśli jest, użyj jej.
   - Jeśli brak, użyj algorytmu wyboru ramki podlegającej wymianie (ang. *victim frame*).
   - Zapisz jej zawartość do pamięci zapasowej, zaktualizuj tablice ramek i stron.
3. Wczytaj oczekiwaną stronę do zwolnionej ramki. Zaktualizuj tablice ramek i stron.
4. Kontynuuj działanie procesu.

Celem optymalizacji stosuje się bit modyfikacji, tzn. brudny bit (ang. *dirty bit*). Np. strony z kodem programu zasadniczo są read-only.



Źródło: A. Silberschatz, *Operating Systems Concepts Essentials*

Algorytmy zastępowania stron

- Algorytm FIFO:
  - najstarsza z umieszczonych w pamięci stron (głowa) jest zastępowana, nowe strony umieszczane są na końcu kolejki (ogon),
  - wada: z jednej strony strona zastępowana może być czymś, co było dawno temu umieszczone w pamięci i jest już nieużywane, ale może to być też często używana zmienna istniejaca od początku procesu.
- Optymalne zastępowanie stron:
  - zastąp stronę, która nie będzie używana przez najdłuższy czas.
  - wada: trzeba znać przyszłość ;-)
- Algorytm LRU - ang. *least recently used*
  - zastąp stronę, która najdłużej nie była używana,
  - implementacje: liczniki (timestamp ostatniego użycia), stos (przekładanie na wierzch użytej strony),
  - dalsze modyfikacje i optymalizacje.

Alokacja ramek

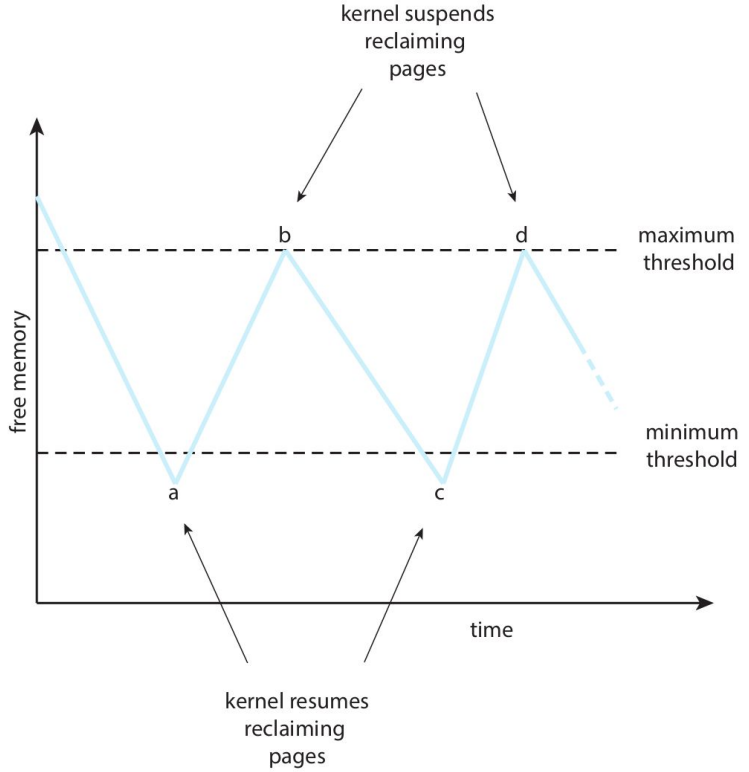
- **Przykład:** system ma 128 ramek pamięci fizycznej, OS zajmuje 35, a 93 ramki są dla procesów.
- W przypadku czystego stronicowania na żądanie (proces w całości ładowany jest do pamięci zapasowej), proces użytkownika wywoła 93 razy błąd strony, a przy żądaniu 94-tej strony zadziała algorytm zastępowania stron. Po zakończeniu procesu zwolnią się wszystkie ramki.
- **Strategie:**
- Minimalna liczba ramek - określona jest minimalna liczba ramek, które muszą zostać zaalokowane. Powód: im mniej zaalokowanych ramek, tym bardziej spada wydajność wykonywanych procesów.
- Równa alokacja - ramki po równo podzielone są między procesy, np. 93 ramki podzielić między 5 procesów oznacza, że każdy z nich otrzyma 18 ramek. Pozostałe 3 ramki trafią do puli wolnych.
- Proporcjonalna alokacja - ramki przydzielane są proporcjonalnie do wielkości procesów.

Globalna i lokalna alokacja

- Alokacja globalna / zastępowanie globalne - realizacja procesu wymaga zastępowania ramek pośród wszystkich ramek, także tych zaalokowanych do innego procesu.
- Alokacja lokalna / zastępowanie lokalne - realizacja procesu wymaga zastępowania ramek tylko pośród zaalokowanych do procesu.

Alokacja globalna ma szczególne zastosowanie w przypadku priorytetyzowania procesów.

Rysunek obok: strategia utrzymania wolnej pamięci w alokacji globalnej.



Źródło: A. Silberschatz, *Operating Systems Concepts Essentials*

Major and minor page faults

- Major page fault - występuje wtedy, gdy strona jest wywoływana, a nie znajduje się w pamięci. Obsługa tego błędu wymaga odczytania wskazanej strony z pamięci zapasowej do wolnej ramki i aktualizacji tablicy stron. Stronicowanie na żądanie zwykle generuje znaczną liczbę tych błędów.
- Minor page fault - występuje wtedy, gdy proces nie ma logicznego mapowania do strony, ale ta strona jest w pamięci. Błąd ten występuje w jednym z przypadków:
    - Proces odwołuje się do biblioteki dzielonej, która jest w pamięci, ale proces nie ma do niej mapowania. W tym przypadku wystarczy zaktualizować tablicę stron.
    - Proces utracił stronę, która trafiła na listę wolnych ramek, ale nie została jeszcze wyzerowana. W tym przypadku strona jest ponownie przypisana do procesu i usunięta z listy wolnych ramek.

Na laboratorium:

```
ps -eo min_flt, maj_flt, cmd
```

Wyspa Utklippan