Zadanie 3

Badanie algorytmów zastępowania stron.

1. Wygenerować losowy ciąg n odwołań do stron
2. Dla wygenerowanego ciągu podać liczbę braków strony dla różnych algorytmów zastępowania stron:

   - FIFO (usuwamy stronę najdłużej przebywającą w pamięci fizycznej)

   - OPT (optymalny - usuwamy stronę, która nie będzie najdłużej używana)

   - LRU (usuwamy stronę, do której najdłużej nie nastąpiło odwołanie)

   - aproksymowany LRU (Last Recently Used)

   - RAND (usuwamy losowo wybraną stronę)

Opracowanie teoretyczne

Pamięć wirtualna – koncepcja, w myśl której dane programu są pofragmentowane, i przechowywane w różnych pamięciach – operacyjnej i masowej. W pamięci operacyjnej przechowywane są tylko te dane, które w danej chwili są niezbędne do prawidłowego wykonania procesu. Wszystko jest zaplanowane w ten sposób, aby wymiana danych między pamięciami była ciągła i dostosowana do aktualnych potrzeb procesu. Dla procesów, całkowita pamięć wirtualna jest odbierana jako jedna, ciągła pamięć operacyjna.

Obszar wymiany - to pamięć dysku twardego przeznaczona do przechowywania danych programów, które w danej chwili nie są przetwarzane i nie muszą być w pamięcifizycznej.

W pamięci wirtualnej (logicznej) przechowywane są tzw. strony. Strony mogą mieć odwzorowanie w pamięci operacyjnej w postaci ramek, lub nie (wskazują bezpośrednio na dane dyskowe). Jeśli jakiś proces chciałby skorzystać ze strony nieposiadającej takiego odwzorowania, to musi ona zostać najpierw sprowadzona do pamięci operacyjnej, co wiąże się z zarezerwowaniem ramki.

Stronicowanie na żądanie – sprowadzenie strony do pamięci operacyjnej tylko gdy jest potrzebna. Nie sprowadza się niepotrzebnych w danej chwili stron.

Odwołanie procesu do pamięci może mieć trojakie skutki:
- odwołanie jest niepoprawne (bit poprawności = 0);
- odwołanie jest poprawne i strona jest już w pamięci (bit poprawności = 1);
- odwołanie jest poprawne, ale strona nie jest w pamięci (bit poprawności = 0).

W trzecim przypadku obsługa zlecenia wymaga sprowadzenia żądanej strony z dysku do pamięci.

Gdy w pamięci nie ma już wolnych ramek na sprowadzenie nowej strony, należy przepisać jakąś nieużywaną ramkę na dysk, a w jej miejsce sprowadzić żądaną stronę. Należy mieć na uwadze, że przepisanie ramki na dysk odbywa się tylko wtedy, gdy nie jest ona jeszcze zachowana w obecnej postaci w pamięci. W tym procesie aktualizowane są również bity poprawności.

Algorytmy zastępowania stron – algorytmy zarządzające przepisywaniem stron, opisanym powyżej. Im mniej przepisań zostanie wykonanych w trakcie pracy, tym efektywniejszy jest algorytm.

FIFO – tzw. ofiarą staje się strona, która najdłużej przebywa w pamięci. Jest ona usuwana i zastępowana nową stroną. To najprostszy algorytm, jednak wydajność działania tego algorytmu jest losowa i mocno zależy od względnej kolejności stron w ciągu odwołań.

OPT – algorytm optymalny. Jego działanie polega na wytypowaniu „ofiary” na podstawie cechy nieznanej w warunkach rzeczywistych – odległości kolejnego odwołania do tej strony. Strona, która najdłużej nie będzie potrzebna w pamięci fizycznej będzie usuwana.Właściwe działanie algorytmu można osiągnąć jedynie operując na statycznej kolejce odwołań do stron.W takich warunkach wydajność algorytmu będzie najwyższa spośród wszystkich.

LRU – (Least Recently Used)idea działania zbliżona do algorytmu OPT, jednak tutaj „ofiarą” staje się strona, która najdłużej była nieużywana – nie ta, która najdłużej będzie nieużywana.

ALRU – (Last Recently Used) „ofiarą” jest strona, która była używana jako ostatnia. W założeniach, jeżeli odwołanie do strony nastąpiło przed chwilą, to kolejne odwołania powinny dotyczyć innych stron.

RAND – algorytm wybiera losową stronę na „ofiarę”.

Struktura programu

Main

- zawiera metodę main inicjującą generator i algorytmy;
- wywołuje funkcjedrukujące wniki.

Generator

- int QUEUE_LENGHT;
- int UNIQUE_PAGES;
- Int FRAMES;
- List<Page> pagesQueue;
- List<Page> framesList.

- generate() – w pętli wykonywanej QUEUE_LENGHT razy tworzy obiekty typu Page i dodaje je do listy pagesQueue;
- static List<Page> clonePagesQueue() – kopiuje i zwraca wygenerowaną kolejkę;
- static List<Page> cloneFramesList() – kopiuje i zwraca zadeklarowaną listę ramek.

Page

- int pageNumber;

- clone() – kopiuje i zwraca obiekt Page o takim samym pageNumber;

abstract Algorithm

- List<Page> pagesQueue = Generator.clonePagesQueue();
- List<Page> framesList = Generator.cloneFramesList();
- List<Page> referenceHistory;
- int replacements = 0

- containsPage() – kopiuje i zwraca obiekt Page o takim samym pageNumber;

FIFO

- process() – przenosi elementy z kolejki do listy ramek dopóki ta nie zostanie zapełniona.

  Gdy jest pełna, to kolejne zamówienia zastępują „najstarszą” ramkę, pod warunkiem, że danej strony nie ma jeszcze w liście ramek; Inkrementujemy licznik replacements.

OPT

- process() – przenosi elementy z kolejki do listy ramek dopóki ta nie zostanie zapełniona.

  Jeśli lista jest pełna: w zmiennej referenceHistory (o rozmiarze UNIQUE_PAGES) tworzymy listę numerów żądanych stron, w kolejności ich pierwszych wystąpieńw kolejce. Element ostatni jest stroną, do której najpóźniej wystąpi pierwsze odwołanie. Tę stronę zastępujemy we framesList.Inkrementujemy licznik replacements.

LRU

- process() – przenosi elementy z kolejki do listy ramek dopóki ta nie zostanie zapełniona. Jednocześnie dodajemy na koniec listy referenceHistory najnowsze wystąpienia danej strony. Jeśli element jest już w liście referenceHistory, przesuwamy go na ostatni indeks (indeks najnowszego wystąpienia).

  Jeśli lista ramek jest pełna: wybieramy pierwszy element z listy referenceHistory i odczytujemy wartość jego strony. Następnie usuwamy z listy framesList pozycję o tej samej wartości i zastępujemy ją nową stroną. Inkrementujemy licznik replacements.

ALRU

- process() – przenosi elementy z kolejki do listy ramek dopóki ta nie zostanie zapełniona. Jednocześnie dodajemy na koniec listy referenceHistory najnowsze wystąpienia danej strony. Jeśli element jest już w liście referenceHistory, przesuwamy go na ostatni indeks (indeks najnowszego wystąpienia).

  Jeśli lista ramek jest pełna: wybieramy ostatni element z listy referenceHistory i odczytujemy wartość jego strony. Następnie usuwamy z listy framesList pozycję o tej samej wartości i zastępujemy ją nową stroną. Inkrementujemy licznik replacements.

RAND

- process() – przenosi elementy z kolejki do listy ramek dopóki ta nie zostanie zapełniona. Jednocześnie dodajemy na koniec listy referenceHistory najnowsze wystąpienia danej strony. Jeśli element jest już w liście referenceHistory, przesuwamy go na ostatni indeks (indeks najnowszego wystąpienia).

  Jeśli lista ramek jest pełna: losujemy liczbę z przedziału indeks ramki którą usuniemy i zastępujemy ją nową stroną.Inkrementujemy licznik replacements.