ОТЧЕТ

по лабораторной работе № 5

«Исследование алгоритмов замещения страниц»

по дисциплине «ОПЕРАЦИОННЫЕ СИСТЕМЫ»

Выполнил студент группы ИС/б-22о

Горбенко К.Н.

Проверила:

Шалимова Е.М.

* 1. ЦЕЛЬ РАБОТЫ

Исследовать стратегии и методы управления виртуальной памятью в современных ОС.

* 1. ЗАДАНИЕ НА ЛАБОРАТОРНУЮ РАБОТУ

Разработать программу, моделирующую работу следующих алгоритмов замещения страниц: **FIFO** и **NUR** (вариант № 4). Реализации алгоритмов должны быть оформлены в виде подпрограмм. Считать, что виртуальная память имеет объем 1000 страниц. В оперативной памяти имеется 10 страниц, о которых известна вся необходимая для работы алгоритмов информация. Главная программа должна вводить исходные данные, вызывать подпрограммы, реализующие алгоритмы, и печатать результаты.

* 1. АЛГОРИТМЫ ЗАМЕЩЕНИЯ СТРАНИЦ
     1. Алгоритм FIFO

Словесное описание алгоритма:

1. Ввод массива обращений appeals[], его размера n, размера массива виртуальной памяти capacity. Создание экземпляра очереди memory, счетчика counter.
2. I = 0.
3. если I < N, то перейти к п. 4, иначе – п. 6.
4. Если memory содержит appealsi, то п. 3., иначе – п. 5.
5. Извлечь первый элемент из memory, вставить в memory appealsi, counter = counter + 1; перейти к п. 3.
6. Вернуть counter.
7. Конец.
   * 1. Алгоритм NRU
8. Ввод массива обращений appeals, его размера n, размера массива виртуальной памяти capacity. Создание массива памяти memory, счетчика counter.
9. I = 0.
10. Если I < N, то перейти к п. 4, иначе – п. 6.
11. Если memory содержит appealsi, то п. 3, иначе – п. 5.
12. position = найти позицию первого элемента memory, для которого бит обращений не установлен. memoryposition.index= appealsi; memoryposition.appeals = memoryposition.appeals + 1; counter = counter + 1; перейти к п. 3.
13. Вернуть counter.
14. Конец.
    1. ОПИСАНИЕ ИСПОЛЬЗУЕМЫХ СТРУКТУР ДАННЫХ

template<class T> class queue;

Структура представляет из себя очередь. Для добавления элемента в конец очереди используется метод void push(T item), для извлечения первого элемента очереди используется метод void pop(). Используется для реализации алгоритмов FIFO.

struct NURNode

{

int page;

int appealed;

};

Структура используется для представления виртуальной памяти, использующей алгоритм NUR для замещения страниц. Она содержит целочисленное поле page, представляющее номер страницы и целочисленное поле appealed, представляющее количество обращений к странице.

* 1. ТЕКСТ ПРОГРАММЫ

#include "pch.h"

#include <iostream>

#include <fstream>

#include <queue>

#include <unordered\_set>

using namespace std;

struct NURNode

{

int page;

int appealed;

};

int contains(NURNode array[], int n, int item)

{

for (int i = 0; i < n; i++)

{

if (array[i].page == item)

return i;

}

return -1;

}

int nur\_page\_faults(int pages[], int n, int capacity)

{

NURNode memory[10];

int pageFaults = 0;

for (int i = 0; i < capacity; i++)

{

memory[i].page = pages[i];

memory[i].appealed = 0;

}

for (int i = 10; i < n; i++)

{

int position = contains(memory, capacity, pages[i]);

if (position != -1)

memory[position].appealed += 1;

else

{

pageFaults += 1;

for (int j = 0; j < capacity; j++)

{

if (memory[j].appealed == 0)

{

memory[j].appealed = 1;

memory[j].page = pages[i];

break;

}

}

}

}

return pageFaults;

}

int fifo\_page\_faults(int pages[], int n, int capacity)

{

unordered\_set<int> s;

queue<int> indexes;

int page\_faults = 0;

for (int i = 0; i < n; i++)

{

if (s.size() < capacity)

{

if (s.find(pages[i]) == s.end())

{

s.insert(pages[i]);

page\_faults++;

indexes.push(pages[i]);

}

}

else

{

if (s.find(pages[i]) == s.end())

{

int val = indexes.front();

indexes.pop();

s.erase(val);

s.insert(pages[i]);

indexes.push(pages[i]);

page\_faults++;

}

}

}

return page\_faults;

}

int main()

{

char path[] = "file.txt";

ifstream file(path);

int pages[1000];

for (int i = 0; i < 1000; i++)

file >> pages[i];

int n = sizeof(pages) / sizeof(pages[0]);

int capacity = 10;

//cout << fifo\_page\_faults(pages, n, capacity);

cout << nur\_page\_faults(pages, n, capacity);

return 0;

}

* 1. РЕЗУЛЬТАТ РАБОТЫ ПРОГРАММЫ

Для одинаковых исходных данных алгоритмы показывают следующие результаты (рисунки № 1 и № 2):

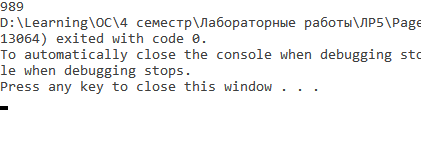


Рисунок № 1 – Результат работы программы, использующей алгоритм FIFO

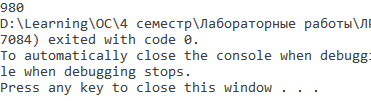


Рисунок № 2 – Результат работы программы, использующей алгоритм NRU

* 1. ВЫВОД

В ходе лабораторной работы были изучены следующие алгоритмы замещения страниц: выталкивание случайной страницы, выталкивание первой пришедшей страницы, выталкивание дольше всего не использовавшейся страницы, выталкивание реже всего используемой страницы, выталкивание не использовавшейся в последнее время страниц, BIFO, рабочие множества.

Экспериментально были реализованы алгоритмы выталкивания первой пришедшей страницы (FIFO) и выталкивания реже всего используемой страницы (NUR). В результате оказалось, что количество страничных прерываний очень велико для данного соотношения виртуальной памяти к оперативной памяти и сравнимо с общим количеством обращений. Разница между реализованными алгоритмами оказалась несущественной в данных условиях.