ГУАП

КАФЕДРА № 43

ОТЧЕТ
ЗАЩИЩЕН С ОЦЕНКОЙ

ПРЕПОДАВАТЕЛЬ

старший преподаватель		М. Д. Поляк
должность, уч. степень, звание	подпись, дата	инициалы, фамилия

ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ №4

Управление памятью

по курсу: Операционные системы

РАБОТУ ВЫПОЛНИЛ

СТУДЕНТ гр. № 4233К	13.02.2025	С. В. Голанова
	подпись, дата	инициалы, фамилия

Санкт-Петербург 2025

Цель работы

Знакомство с принципами организации виртуальной памяти.

Задание на лабораторную работу

В данной работе необходимо реализовать фрагмент диспетчера памяти и часть функционала операционной системы, отвечающего за замещение страниц при возникновении ошибок отсутствия страниц. Для упрощения работы предполагается использование линейной инвертированной таблицы страниц, работу с которой необходимо реализовать в виде программы. Также для простоты предполагается, что в системе имеется один единственный процесс, поэтому идентификатор процесса в инвертированной таблице страниц не хранится. Входные данные представляют собой последовательность операций обращения к памяти, выходные данные - состояние инвертированной таблицы страниц после каждой операции обращения к памяти.

1. Вычислить номер варианта по списку в журнале и сохранить его в файл TASKID.txt в репозитории.
2. Написать программу на языке C++ в соответствии со следующей спецификацией.
    i. Входные данные:
        a. Аргумент командной строки (число): номер алгоритма замещения страниц, который должна использовать программа. Принимает значения 1 или 2, соответствующие двум алгоритмам замещения страниц, заданным по варианту.
        b. Перечень инструкций обращения к памяти, считываемый программой из стандартного потока ввода. На каждой строке не более одной инструкции. Инструкция состоит из двух чисел, разделенных пробелом, например: 0 1. Первое число обозначает тип операции доступа к памяти: 0 - чтение и 1 - запись. Второе число является номером виртуальной страницы, к которой происходит обращение.
    ii. Выходные данные:
        a. Для каждой операции обращения к памяти, информация о которой поступила на вход программы, на выходе должна быть сгенерирована строка, содержащая содержимое инвертированной таблицы страниц в виде последовательности номеров виртуальных страниц, разделенных пробелом. Если какая-либо из записей в таблице страниц отсутствует (таблица страниц не заполнена до конца), вместо номера виртуальной страницы необходимо вывести символ #.

3. Весь код поместить в файле lab4.cpp. Код должен корректно компилироваться командой g++ lab4.cpp -o lab4 -std=c++11. Настоятельно рекомендуется использовать стандартную библиотеку STL. Полезными могут быть контейнеры list, vector, bitset и др.
4. Если в работе алгоритма замещения страниц используется бит R, то необходимо реализовать эмуляцию прерывания таймера. Для этого через каждые 5 операций обращения к памяти необходимо запускать обработчик данного прерывания. Значения битов R по прерыванию таймера сбрасываются.
5. Для алгоритмов, использующих счетчик (NFU, Aging): если несколько страниц имеют одинаковое значение счетчика, одна из них выбирается случайным образом. При повторной загрузке страницы в память ее счетчик обнуляется. В алгоритме старения счетчик имеет размер 1 байт. В алгоритме NFU счетчик имеет размер не меньше 4 байт.
6. Во всех алгоритмах, использующих датчик случайных чисел (Random, NRU, NFU, Aging, ...), разрешается использовать **только** функцию int uniform_rnd(int a, int b), объявленную в файле lab4.h. Данная функция генерирует случайное целое число с равномерным распределением из диапазона [a, b]. Использование других функций для работы со случайными числами запрещено!
7. В качестве системного времени в алгоритме рабочего набора следует использовать количество инструкций доступа к памяти, обработанных с момента запуска программы.
8. После успешного прохождения локальных тестов необходимо загрузить код в репозиторий на гитхабе.
9. Сделать выводы об эффективности реализованных алгоритмов замещения страниц. Сравнить количество ошибок отсутствия страниц, генерируемых на тестовых данных при использовании каждого алгоритма.
10. Подготовить отчет о выполнении лабораторной работы и загрузить его под именем report.pdf в репозиторий. В случае использования системы компьютерной верстки LaTeX также загрузить исходный файл report.tex.

Вариант задания

Таблица 1 – Вариант задания на лабораторную работу

Номер варианта	**Количество страничных блоков**	**Алгоритм 1**	**Алгоритм 2**
28	14	Random	Aging

Описание используемых алгоритмов замещения страниц

Случайный выбор (Random)

Выбор замещаемой страницы осуществляется с помощью датчика случайных чисел, имеющего равномерное распределение. Таким образом, при возникновении ошибки отсутствия страницы вероятность быть выгруженной для всех страниц одинакова.

Старение (Aging)

Дальнейшая модификация алгоритмов LRU и NFU. Для каждой страницы, загруженной в память, заводится счетчик. По прерыванию от таймера счетчики всех страниц сдвигаются вправо на 1 бит. Затем, самый старший (левый) бит счетчика устанавливается равным значению бита использования (R) этой страницы. В случае возникновения ошибки отсутствия страницы, удаляется та страница, значение счетчика которой минимально. Если имеется несколько страниц с одинаковым минимальным значением счетчика, необходимо выбрать одну из них случайным образом.

Результат выполнения работы

В результате выполнения работы была написана программа, имитирующая фрагмент диспетчера памяти. В данной программе реализованы два алгоритма Random и Aging, которые могут обработать ошибку отсутствия страницы.

Ниже приведены результаты прохождения тестирования данными алгоритмами рисунок 1.

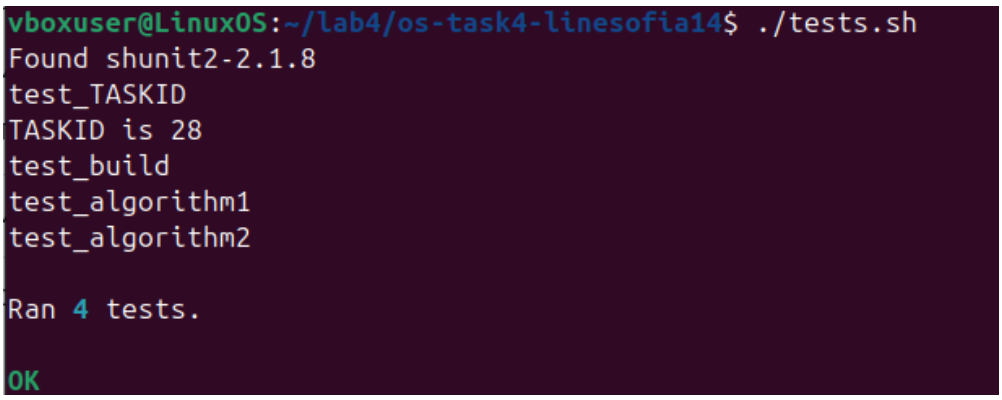

Рисунок 1 – Результаты тестирования программы lab4

После тестирования были созданы два файла, output1.txt содержит порядок заполнения таблицы страниц при использовании алгоритма Ramdom. Текст файла приведен ниже:

Output1.txt

```
10 # # # # # # # # # # # #
```

10 30 # # # # # # # # # # # # #

10 30 4 # # # # # # # # # # # #

10 30 4 27 # # # # # # # # # #

10 30 4 27 15 # # # # # # # # #

10 30 4 27 15 19 # # # # # # # #

10 30 4 27 15 19 14 # # # # # # #

10 30 4 27 15 19 14 43 # # # # # #

10 30 4 27 15 19 14 43 24 # # # # #

10 30 4 27 15 19 14 43 24 18 # # # #

10 30 4 27 15 19 14 43 24 18 5 # # #

10 30 4 27 15 19 14 43 24 18 5 # # #

10 30 4 27 15 19 14 43 24 18 5 # # #

10 30 4 27 15 19 14 43 24 18 5 2 # #

10 30 4 27 15 19 14 43 24 18 5 2 # #

10 30 4 27 15 19 14 43 24 18 5 2 34 #

10 30 4 27 15 19 14 43 24 18 5 2 34 7

10 30 4 27 15 19 14 43 24 18 5 2 34 7

10 30 4 27 15 19 14 43 55 18 5 2 34 7

10 30 4 27 15 19 14 43 55 18 5 2 28 7

10 30 4 27 15 19 14 43 55 18 5 2 28 7

40 30 4 27 15 19 14 43 55 18 5 2 28 7

40 30 4 27 15 19 14 8 55 18 5 2 28 7

44 30 4 27 15 19 14 8 55 18 5 2 28 7

44 30 4 27 15 19 14 8 55 18 5 2 49 7

44 30 4 27 15 19 1 8 55 18 5 2 49 7

44 30 4 27 21 19 1 8 55 18 5 2 49 7

44 30 4 27 21 19 1 8 55 18 5 2 49 7

44 30 0 27 21 19 1 8 55 18 5 2 49 7

44 30 0 27 21 19 1 8 55 18 5 2 49 7

44 30 0 27 21 19 1 8 55 18 5 2 49 54

44 35 0 27 21 19 1 8 55 18 5 2 49 54

15 35 0 27 21 19 1 8 55 18 5 2 49 54

15 35 0 27 21 19 1 8 55 18 5 2 49 54

15 35 0 27 21 19 1 8 55 18 5 2 49 54

15 35 0 27 21 19 1 8 55 18 5 2 49 54

15 46 0 27 21 19 1 8 55 18 5 2 49 54

15 24 0 27 21 19 1 8 55 18 5 2 49 54

15 24 0 27 21 19 1 8 55 25 5 2 49 54

15 24 0 27 21 19 1 8 55 25 5 2 49 54

15 24 3 27 21 19 1 8 55 25 5 2 49 54

15 24 3 27 21 19 1 38 55 25 5 2 49 54

8 24 3 27 21 19 1 38 55 25 5 2 49 54

8 24 3 27 21 19 1 38 55 25 5 40 49 54

8 24 3 27 21 19 1 38 55 25 22 40 49 54

8 24 3 27 21 19 17 38 55 25 22 40 49 54

8 24 3 27 21 19 17 38 36 25 22 40 49 54

8 24 3 27 21 44 17 38 36 25 22 40 49 54

8 24 3 27 21 31 17 38 36 25 22 40 49 54

8 24 3 27 21 31 17 38 36 25 22 40 49 42

8 24 3 27 21 31 17 4 36 25 22 40 49 42

8 24 44 27 21 31 17 4 36 25 22 40 49 42

8 24 44 27 21 38 17 4 36 25 22 40 49 42

8 24 44 27 21 11 17 4 36 25 22 40 49 42

8 24 44 27 21 11 17 4 36 25 22 40 6 42

8 24 44 27 21 7 17 4 36 25 22 40 6 42

8 24 44 27 21 7 17 4 36 25 22 40 5 42

8 24 44 27 21 7 17 4 36 25 6 40 5 42

8 24 10 27 21 7 17 4 36 25 6 40 5 42

8 24 10 27 21 7 17 4 36 25 6 40 5 42

Файл же output2.txt содержит порядок заполнения таблицы страниц при использовании алгоритма Aging. Текст файла приведен ниже.

Output2.txt

10 # # # # # # # # # # # # #

10 30 # # # # # # # # # # # #

10 30 4 # # # # # # # # # # #

10 30 4 27 # # # # # # # # # #

10 30 4 27 15 # # # # # # # # #

10 30 4 27 15 19 # # # # # # # # #

10 30 4 27 15 19 14 # # # # # # # #

10 30 4 27 15 19 14 43 # # # # # #

10 30 4 27 15 19 14 43 24 # # # # #

10 30 4 27 15 19 14 43 24 18 # # # #

10 30 4 27 15 19 14 43 24 18 5 # # #

10 30 4 27 15 19 14 43 24 18 5 # # #

10 30 4 27 15 19 14 43 24 18 5 # # #

10 30 4 27 15 19 14 43 24 18 5 2 # #

10 30 4 27 15 19 14 43 24 18 5 2 # #

10 30 4 27 15 19 14 43 24 18 5 2 34 #

10 30 4 27 15 19 14 43 24 18 5 2 34 7

10 30 4 27 15 19 14 43 24 18 5 2 34 7

10 30 4 27 15 19 14 43 24 18 5 2 55 7

10 30 4 27 15 19 14 43 24 18 5 2 28 7

10 30 4 27 15 19 14 43 24 18 5 2 28 7

10 30 4 40 15 19 14 43 24 18 5 2 28 7

10 30 4 8 15 19 14 43 24 18 5 2 28 7

10 30 4 44 15 19 14 43 24 18 5 2 28 7

10 30 4 49 15 19 14 43 24 18 5 2 28 7

10 30 4 49 1 19 14 43 24 18 5 2 28 7

10 30 4 49 21 19 14 43 24 18 5 2 28 7

10 30 4 49 8 19 14 43 24 18 5 2 28 7

10 30 4 49 0 19 14 43 24 18 5 2 28 7

10 30 4 49 1 19 14 43 24 18 5 2 28 7

10 30 4 49 1 19 14 54 24 18 5 2 28 7

10 30 4 49 1 19 14 35 24 18 5 2 28 7

10 30 4 49 1 19 14 15 24 18 5 2 28 7

10 30 4 49 1 19 14 27 24 18 5 2 28 7

10 30 4 49 1 19 14 21 24 18 5 2 28 7

10 30 4 49 1 19 14 21 0 18 5 2 28 7

10 30 4 49 1 19 14 21 46 18 5 2 28 7

10 30 4 49 1 19 14 21 24 18 5 2 28 7

10 30 4 49 1 19 14 21 25 18 5 2 28 7

```
10 30 4 49 1 19 14 21 8 18 5 2 28 7
10 30 4 49 1 3 14 21 8 18 5 2 28 7
10 30 4 49 1 38 14 21 8 18 5 2 28 7
10 30 4 49 1 38 14 21 8 18 5 2 28 7
10 30 4 49 1 40 14 21 8 18 5 2 28 7
10 30 4 49 1 22 14 21 8 18 5 2 28 7
10 30 4 49 1 22 14 21 8 17 5 2 28 7
10 30 4 49 1 22 14 21 8 36 5 2 28 7
10 30 4 49 1 22 14 21 8 44 5 2 28 7
10 30 4 49 1 22 14 21 8 31 5 2 28 7
10 30 4 49 1 22 14 21 8 42 5 2 28 7
10 30 4 49 1 22 14 21 8 42 5 2 28 7
44 30 4 49 1 22 14 21 8 42 5 2 28 7
38 30 4 49 1 22 14 21 8 42 5 2 28 7
11 30 4 49 1 22 14 21 8 42 5 2 28 7
6 30 4 49 1 22 14 21 8 42 5 2 28 7
6 30 4 49 1 22 14 21 8 42 5 2 28 7
6 30 4 49 1 22 14 21 8 42 5 2 28 7
6 30 4 49 1 22 14 21 8 42 5 2 28 7
6 10 4 49 1 22 14 21 8 42 5 2 28 7
6 10 4 49 1 22 14 21 8 42 5 2 28 7
```

Количество ошибок отсутствия страниц для алгоритмов:

Random: 34 ошибки отсутствия страниц.

Aging: 35 ошибок отсутствия страниц.

Текст файла тестирования:

```
1 10
1 30
1 4
0 27
0 15
0 19
0 14
1 43
0 24
```

0 18
0 5
0 30
0 10
0 2
1 4
0 34
0 7
0 14
0 55
0 28
0 28
0 40
0 8
1 44
0 49
0 1
0 21
1 8
0 0
0 1
0 54
1 35
0 15
0 27
0 21
0 0
0 46
1 24
0 25
0 8
0 3
0 38
1 8

```
0 40
0 22
1 17
0 36
0 44
0 31
0 42
0 4
1 44
0 38
0 11
0 6
1 7
0 5
0 6
0 10
0 42
```

Исходный код программы с комментариями

Ниже приведено содержание файла lab4.cpp.

```cpp
#include <iostream>
#include <vector>
#include <cstdlib>
#include "lab4.h"

using namespace std;

const int PAGE_TABLE_SIZE = 14;

// Inverted Page Table Structure
struct PageTableEntry {
    int vpn;       // Virtual Page Number
    bool r_bit;    // Reference Bit
    bool m_bit;    // Modified Bit
    unsigned char aging_counter; // 8-bit counter for Aging
};

// Function to initialize the page table
void initializePageTable(vector<PageTableEntry>& page_table) {
    for (int i = 0; i < PAGE_TABLE_SIZE; ++i) {
        page_table[i].vpn = -1; // -1 indicates an empty frame
```

```
        page_table[i].r_bit = false;
        page_table[i].m_bit = false;
        page_table[i].aging_counter = 0;
    }
}

// Function to perform random page replacement
int randomReplacement(const vector<PageTableEntry>& page_table) {
    return uniform_rnd(0, PAGE_TABLE_SIZE - 1);
}

// Function to perform aging page replacement
int agingReplacement(const vector<PageTableEntry>& page_table) {
    unsigned char min_counter = 255; // Initialize to max value
    vector<int> candidates;

    // Find pages with the smallest counter value
    for (int i = 0; i < PAGE_TABLE_SIZE; ++i) {
        if (page_table[i].aging_counter < min_counter) {
            min_counter = page_table[i].aging_counter;
            candidates.clear(); // New minimum found
            candidates.push_back(i);
        } else if (page_table[i].aging_counter == min_counter) {
            candidates.push_back(i); // Add to candidates
        }
    }

    // Randomly choose among the candidates
    if (candidates.size() > 1) {
        int rand_index = uniform_rnd(0, candidates.size() - 1);
        return candidates[rand_index];
    } else {
        return candidates[0];
    }
}

// Function to perform page replacement based on the selected algorithm
int pageReplacement(vector<PageTableEntry>& page_table, int algorithm) {
    if (algorithm == 1) {
        return randomReplacement(page_table);
    } else { // algorithm == 2
        return agingReplacement(page_table);
    }
}

// Function to simulate the timer interrupt
void timerInterrupt(vector<PageTableEntry>& page_table) {
    for (int i = 0; i < PAGE_TABLE_SIZE; ++i) {
        // Right shift counter and add R bit to MSB
```

```
        page_table[i].aging_counter = (page_table[i].aging_counter >> 1) |
(page_table[i].r_bit ? 0x80 : 0x00);
        page_table[i].r_bit = false; // Reset R bit
    }
}

// Function to print the page table state
void printPageTable(const vector<PageTableEntry>& page_table) {
    for (int i = 0; i < PAGE_TABLE_SIZE; ++i) {
        if (page_table[i].vpn == -1) {
            cout << "#";
        } else {
            cout << page_table[i].vpn;
        }
        if (i < PAGE_TABLE_SIZE - 1) {
            cout << " ";
        }
    }
    cout << endl;
}

int main(int argc, char *argv[]) {
    if (argc != 2) {
        cerr << "Usage: " << argv[0] << " <algorithm_number>" << endl;
        return 1;
    }

    int algorithm = atoi(argv[1]); // 1 for Random, 2 for Aging

    if (algorithm != 1 && algorithm != 2) {
        cerr << "Invalid algorithm number. Choose 1 or 2." << endl;
        return 1;
    }

    vector<PageTableEntry> page_table(PAGE_TABLE_SIZE);
    initializePageTable(page_table);

    int instruction_count = 0;
    int access_type, virtual_page;

    while (cin >> access_type >> virtual_page) {
        instruction_count++;

        bool page_hit = false;
        int frame_index = -1;

        // Search for the page in the page table
        for (int i = 0; i < PAGE_TABLE_SIZE; ++i) {
            if (page_table[i].vpn == virtual_page) {
```

```
                page_hit = true;
                frame_index = i;
                break;
            }
        }

        if (page_hit) {
            // Page Hit: Update R and M bits
            page_table[frame_index].r_bit = true;
            if (access_type == 1) {
                page_table[frame_index].m_bit = true;
            }
        } else {
            // Page Fault
            bool empty_frame_found = false;

            // Check for empty frames
            for (int i = 0; i < PAGE_TABLE_SIZE; ++i) {
                if (page_table[i].vpn == -1) {
                    // Found an empty frame, load the page
                    page_table[i].vpn = virtual_page;
                    page_table[i].r_bit = true;
                    page_table[i].m_bit = (access_type == 1); // Set M bit on write
                    page_table[i].aging_counter = 0;  // Reset aging counter
                    empty_frame_found = true;
                    break;
                }
            }

            // If no empty frames, perform page replacement
            if (!empty_frame_found) {
                int replace_index = -1;

                // Use the pageReplacement function to select the page to replace
                replace_index = pageReplacement(page_table, algorithm);

                // Replace the page
                page_table[replace_index].vpn = virtual_page;
                page_table[replace_index].r_bit = true;
                page_table[replace_index].m_bit = (access_type == 1); // Set M bit
on write
                page_table[replace_index].aging_counter = 0; // Reset aging counter
            }
        }

        // Timer Interrupt (every 5 instructions)
        if (instruction_count % 5 == 0 && algorithm == 2) {
            timerInterrupt(page_table);
        }
```

```
        printPageTable(page_table);
    }

    return 0;
}
```

Выводы

В ходе выполнения лабораторной работы был реализован фрагмент диспетчера памяти, эмулирующий работу подсистемы замещения страниц в операционной системе. Были реализованы два алгоритма замещения страниц: Random и Aging с использованием инвертированной таблицы страниц. Результаты работы показали, что количество ошибок отсутствия страниц различается в зависимости от используемого алгоритма и входных данных. Сравнение количества ошибок, сгенерированных каждым алгоритмом, позволяет сделать выводы об их относительной эффективности в различных сценариях нагрузки на память. При выполнении алгоритма случайного выбора ошибка отсутствия страницы возникла на один раз меньше, чем при выполнении алгоритма старения, но дальнейшего анализа целесообразно провести эксперименты с различными наборами входных данных. Так как такая разница не показательна. Сравнение алгоритмов замещения страниц Random и Aging показывает, что алгоритм Aging, как правило, более эффективен в контексте уменьшения количества ошибок отсутствия страниц. Алгоритм Aging использует биты обращения и счётчики, что позволяет ему учитывать частоту использования страниц и более точно определять, какие страницы следует выгружать из памяти. В отличие от алгоритма Random, который выбирает страницы для замещения случайным образом, алгоритм Aging учитывает историю обращений к страницам, что делает его более адаптивным к реальным условиям работы системы.