ГУАП

КАФЕДРА № 43

ОТЧЕТ   
ЗАЩИЩЕН С ОЦЕНКОЙ

ПРЕПОДАВАТЕЛЬ

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ассистент |  |  |  | Д.А. Кочин |
| должность, уч. степень, звание |  | подпись, дата |  | инициалы, фамилия |

|  |
| --- |
| ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ №4 |
| Управление памятью |
| по курсу: ОПЕРАЦИОННЫЕ СИСТЕМЫ |
|  |
|  |

РАБОТУ ВЫПОЛНИЛ

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| СТУДЕНТ ГР. № | 4231 |  |  |  | К.А. Чистякова |
|  |  |  | подпись, дата |  | инициалы, фамилия |

Санкт-Петербург 2024

**Цель работы:** Знакомство с принципами организации виртуальной памяти

**Задание и последовательность выполнения работы:**

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](https://github.com/suai-os-2024f/os-task4-Krismin0/blob/master/TASKID.txt) в репозитории.
2. Написать программу на языке C++ в соответствии со следующей спецификацией.
   1. Входные данные:
      1. 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.
      2. 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.
   2. Выходные данные:
      1. 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](https://en.cppreference.com/w/cpp/container/list), [vector](https://en.cppreference.com/w/cpp/container/vector), [bitset](https://en.cppreference.com/w/cpp/utility/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](https://github.com/suai-os-2024f/os-task4-Krismin0/blob/master/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.

**Описание используемых алгоритмов замещения страниц**

В коде реализованы два алгоритма замещения страниц: Second Chance (второй шанс) и NRU (Not Recently Used, наименее используемые недавно).

Second Chance

Этот алгоритм работает по принципу FIFO, но дает страницам «второй шанс».

1. Если страница уже есть в таблице, её бит R устанавливается в 1.
2. Если таблица заполнена, выбирается страница из головы очереди:
   * Если R = 0, страница заменяется.
   * Если R = 1, R сбрасывается в 0, а страница перемещается в конец очереди.
3. Таймерное прерывание (timer\_interrupt()) периодически сбрасывает все биты R.

Реализация: do\_second\_chance()

* Новая страница добавляется в tab[] и очередь second\_chance\_queue.
* Если таблица заполнена, ищется страница с R = 0, иначе R сбрасывается, и страница уходит в конец очереди.

NRU (Not Recently Used)

NRU делит страницы на 4 класса по битам R (использование) и M (изменение):

* R=0, M=0 – наилучший кандидат для удаления.
* R=0, M=1
* R=1, M=0
* R=1, M=1 – наихудший кандидат.

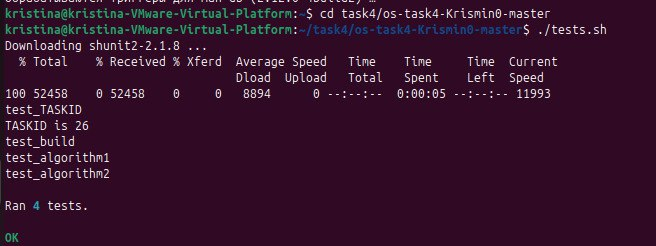
При замене выбирается страница с минимальным классом. Если таких несколько – выбирается случайная.

Реализация: do\_nru()

* Определяется класс каждой страницы.
* Ищется страница с наименьшим классом.
* Если несколько таких страниц – выбирается случайно (uniform\_rnd()).
* Новая страница заменяет найденную.

**Результат выполнения работы**

**Прохождение тестов**



**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

55 30 4 27 15 19 14 43 24 18 5 2 34 7

55 28 4 27 15 19 14 43 24 18 5 2 34 7

55 28 4 27 15 19 14 43 24 18 5 2 34 7

55 28 40 27 15 19 14 43 24 18 5 2 34 7

55 28 40 8 15 19 14 43 24 18 5 2 34 7

55 28 40 8 44 19 14 43 24 18 5 2 34 7

55 28 40 8 44 49 14 43 24 18 5 2 34 7

55 28 40 8 44 49 1 43 24 18 5 2 34 7

55 28 40 8 44 49 1 21 24 18 5 2 34 7

55 28 40 8 44 49 1 21 24 18 5 2 34 7

55 28 40 8 44 49 1 21 0 18 5 2 34 7

55 28 40 8 44 49 1 21 0 18 5 2 34 7

55 28 40 8 44 49 1 21 0 54 5 2 34 7

55 28 40 8 44 49 1 21 0 54 35 2 34 7

55 28 40 8 44 49 1 21 0 54 35 15 34 7

55 28 40 8 44 49 1 21 0 54 35 15 27 7

55 28 40 8 44 49 1 21 0 54 35 15 27 7

55 28 40 8 44 49 1 21 0 54 35 15 27 7

55 28 40 8 44 49 1 21 0 54 35 15 27 46

24 28 40 8 44 49 1 21 0 54 35 15 27 46

24 25 40 8 44 49 1 21 0 54 35 15 27 46

24 25 40 8 44 49 1 21 0 54 35 15 27 46

24 25 3 8 44 49 1 21 0 54 35 15 27 46

24 25 3 38 44 49 1 21 0 54 35 15 27 46

24 25 3 38 8 49 1 21 0 54 35 15 27 46

24 25 3 38 8 40 1 21 0 54 35 15 27 46

24 25 3 38 8 40 22 21 0 54 35 15 27 46

24 25 3 38 8 40 22 17 0 54 35 15 27 46

24 25 3 38 8 40 22 17 36 54 35 15 27 46

24 25 3 38 8 40 22 17 36 44 35 15 27 46

24 25 3 38 8 40 22 17 36 44 31 15 27 46

24 25 3 38 8 40 22 17 36 44 31 42 27 46

24 25 3 38 8 40 22 17 36 44 31 42 4 46

24 25 3 38 8 40 22 17 36 44 31 42 4 46

24 25 3 38 8 40 22 17 36 44 31 42 4 46

24 25 3 38 8 40 22 17 36 44 31 42 4 11

6 25 3 38 8 40 22 17 36 44 31 42 4 11

6 7 3 38 8 40 22 17 36 44 31 42 4 11

6 7 5 38 8 40 22 17 36 44 31 42 4 11

6 7 5 38 8 40 22 17 36 44 31 42 4 11

6 7 5 10 8 40 22 17 36 44 31 42 4 11

6 7 5 10 8 40 22 17 36 44 31 42 4 11

**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 55 19 14 43 24 18 5 2 34 7

10 30 4 27 55 19 14 43 24 18 28 2 34 7

10 30 4 27 55 19 14 43 24 18 28 2 34 7

10 30 4 27 55 19 14 43 40 18 28 2 34 7

10 30 4 27 8 19 14 43 40 18 28 2 34 7

10 30 4 44 8 19 14 43 40 18 28 2 34 7

10 30 4 44 8 19 14 43 40 18 28 2 49 7

10 30 4 44 8 19 14 43 40 18 28 2 49 1

10 30 4 44 8 19 21 43 40 18 28 2 49 1

10 30 4 44 8 19 21 43 40 18 28 2 49 1

10 30 4 44 8 19 21 43 40 18 28 0 49 1

10 30 4 44 8 19 21 43 40 18 28 0 49 1

10 30 4 44 8 19 21 43 40 18 28 0 49 54

10 30 4 44 8 19 35 43 40 18 28 0 49 54

10 30 4 44 8 19 35 43 40 15 28 0 49 54

10 30 4 44 8 19 35 43 27 15 28 0 49 54

10 30 4 44 8 19 35 43 27 15 21 0 49 54

10 30 4 44 8 19 35 43 27 15 21 0 49 54

10 30 4 44 8 19 35 43 46 15 21 0 49 54

10 30 4 44 8 19 35 43 46 15 24 0 49 54

10 30 4 44 8 19 35 43 46 15 24 0 49 25

10 30 4 44 8 19 35 43 46 15 24 0 49 25

10 30 4 44 8 19 35 43 46 15 24 0 3 25

10 30 4 44 8 19 35 43 46 15 24 0 3 38

10 30 4 44 8 19 35 43 46 15 24 0 3 38

10 30 4 44 8 40 35 43 46 15 24 0 3 38

10 30 4 44 8 40 35 43 46 22 24 0 3 38

10 30 4 44 8 17 35 43 46 22 24 0 3 38

10 30 4 44 8 17 35 43 46 22 24 36 3 38

10 30 4 44 8 17 35 43 46 22 24 36 3 38

10 30 4 44 8 17 35 43 46 22 24 36 3 31

10 30 4 44 8 17 35 43 42 22 24 36 3 31

10 30 4 44 8 17 35 43 42 22 24 36 3 31

10 30 4 44 8 17 35 43 42 22 24 36 3 31

10 30 4 44 8 17 35 43 42 22 24 36 38 31

10 30 4 44 8 17 35 43 42 22 24 11 38 31

10 30 4 44 8 17 35 43 6 22 24 11 38 31

10 30 4 44 8 17 35 43 6 22 24 7 38 31

10 30 4 44 8 17 35 43 6 22 24 7 5 31

10 30 4 44 8 17 35 43 6 22 24 7 5 31

10 30 4 44 8 17 35 43 6 22 24 7 5 31

10 30 4 44 8 17 35 43 6 22 24 7 5 42

**Исходный код программы с комментариями**

#include "lab4.h"

#include <cstring>

#include <iostream>

#include <queue>

#include <vector>

// 26

class VirtualPage {

public:

int number;

bool R;

bool M;

VirtualPage(int n, bool R, bool M) : number(n), R(R), M(M) {}

};

int operations\_counter = 0;

const int pages\_count = 14;

VirtualPage \*tab[pages\_count];

std::queue<int> second\_chance\_queue;

void timer\_interrupt() {

for (int i = 0; i < pages\_count; ++i) {

if (tab[i])

tab[i]->R = 0;

}

}

void log\_current\_table\_state() {

for (int i = 0; i < pages\_count - 1; ++i) {

if (tab[i])

std::cout << tab[i]->number;

else

std::cout << '#';

std::cout << " ";

}

if (tab[pages\_count - 1])

std::cout << tab[pages\_count - 1]->number;

else

std::cout << "#";

std::cout << std::endl;

}

void do\_second\_chance(int is\_write, int n) {

for (int i = 0; i < pages\_count; ++i) {

if (!tab[i]) {

// вставка страницы на свободное место

tab[i] = new VirtualPage(n, 1, is\_write);

second\_chance\_queue.push(i);

return;

} else if (tab[i]->number == n) {

// страница уже есть в таблице

tab[i]->R = true;

return;

}

}

while (true) {

// берём страницу из головы очереди

int indexToReplace = second\_chance\_queue.front();

second\_chance\_queue.pop();

if (!tab[indexToReplace]->R) {

// бит использования не установлен - страница заменяется

delete tab[indexToReplace];

tab[indexToReplace] = new VirtualPage(n, 1, is\_write);

second\_chance\_queue.push(indexToReplace);

return;

} else {

// Если при возникновении ошибки отсутствия страницы в памяти у страницы,

// находящейся в голове очереди, бит использования R установлен (т.е.

// равен 1), то этот бит сбрасывается в ноль, а сама страница перемещается

// в конец очереди.

tab[indexToReplace]->R = 0;

second\_chance\_queue.push(indexToReplace);

}

}

}

void do\_nru(int is\_write, int n) {

for (int i = 0; i < pages\_count; ++i) {

if (!tab[i]) {

// вставка страницы на свободное место

tab[i] = new VirtualPage(n, 1, is\_write);

return;

} else if (tab[i]->number == n) {

// страница уже есть в таблице

tab[i]->R = true;

if (!tab[i]->M)

tab[i]->M = is\_write;

return;

}

}

std::vector<int> replace\_candidates;

int min\_page\_class = 5, page\_class;

for (int i = 0; i < pages\_count; ++i) {

// подсчёт класса страницы

if (tab[i]->M && !tab[i]->R)

page\_class = 1;

else if (tab[i]->R && !tab[i]->M)

page\_class = 2;

else if (tab[i]->R && tab[i]->M)

page\_class = 3;

else

page\_class = 0;

// обнаружена страница с меньшим классом - предыдущие страницы можно удилить

// из списка кандидатов на замену

if (page\_class < min\_page\_class) {

replace\_candidates.clear();

min\_page\_class = page\_class;

}

if (min\_page\_class == page\_class) {

replace\_candidates.push\_back(i);

}

}

// Удаляют страницу, относящуюся к классу с наименьшим номером.

// Если таких страниц несколько, одну из них выбирают случайным образом.

int replaceIndex =

replace\_candidates[uniform\_rnd(0, replace\_candidates.size() - 1)];

delete tab[replaceIndex];

tab[replaceIndex] = new VirtualPage(n, 1, is\_write);

}

int main(int argc, char \*argv[]) {

int is\_write, virtual\_page\_number;

while (std::cin >> is\_write >> virtual\_page\_number) {

if (argv[1][0] == '1') {

// second chance

do\_second\_chance(is\_write, virtual\_page\_number);

} else {

// nru

do\_nru(is\_write, virtual\_page\_number);

}

// прерывание таймера каждые 5 операций

if (++operations\_counter % 5 == 0)

timer\_interrupt();

log\_current\_table\_state();

}

for (int i = 0; i < pages\_count; ++i) {

delete tab[i];

}

}

**Вывод:**

В данной работе были реализованы и протестированы два алгоритма замещения страниц: Second Chance и NRU.

Алгоритм Second Chance использует принцип FIFO, но учитывает бит R, давая страницам «второй шанс» перед удалением. Это снижает вероятность замены активно используемых страниц, но может привести к увеличению времени поиска замены.

Алгоритм NRU классифицирует страницы по битам R и M, выбирая для замены наименее используемые и изменяемые страницы. Это делает его более эффективным при управлении памятью, но требует периодического сброса битов.

Оба алгоритма демонстрируют разные подходы к управлению страницами: Second Chance упрощает реализацию замещения, а NRU оптимизирует выбор жертвы, снижая частоту обращений к диску.