**МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ**

**УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ**

**ГОМЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ**

**УНИВЕРСИТЕТ ИМЕНИ П. О. СУХОГО**

Факультет автоматизированных и информационных систем

Кафедра «Информационные технологии»

ОТЧЁТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №5

по дисциплине: «Операционные системы»

на тему: **«**Простейшие схемы управления памятью**»**

Выполнил: студент гр. ИТИ-12

Ёвженко Ю.Д.

Принял: преподаватель-стажёр

Дашкевич Д.А.

Гомель 2022

**Цель работы**: изучение алгоритмов управления памятью, разработка программы менеджера памяти.

**Задание:**

Разработать программу, реализующую заданный алгоритм (табл. 1) выделения памяти. Менеджер памяти должен:

1. По запросу процесса выделять память, согласно заданного алгоритма (таблица). На экран должна выводиться следующая информация о состоянии памяти: объем памяти, объём свободной памяти, размер наибольшего свободного блока, количество запросов на выделение памяти, количество удовлетворённых запросов (%).

2. Для выделения памяти указывается имя процесса и размер блока. После нажатия на кнопку «ДОБАВИТЬ» память выделяется или выдаётся сообщение о невозможности выделения.

3. Удалять из памяти заданный блок или все блоки заданного процесса (по нажатию кнопки «УДАЛИТЬ»). Указывается номер удаляемого блока и имя процесса.

4. Реализовать возможность последовательной записи/чтения информации в/из выделенную память по логическому адресу. Вывести физического адреса ячейки памяти, в которую была осуществлена запись.

5. Организовывать циклическое выделение и освобождение памяти. При этом случайным образом задается количество выделяемых блоков и их размер.

Таблица 1 – Вариант выполняемого задания

|  |  |
| --- | --- |
| Вариант | Задание |
| 4 | **Свопинг.** Выгружается процесс, занимающий наибольший объём памяти. |

**Ход работы**

На рисунке 1 показан ввод кол-ва байт в менеджере и меню программы.

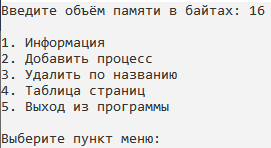


Рисунок 1 – Меню программы

На рисунке 2 показан процесс добавления объекта в память



Рисунок 2 – Добавление процесса

На рисунке 3 показано освобождение памяти по названию процесса



Рисунок 3 – Удаление процесса

На рисунке 4 показано расположение процессов в памяти

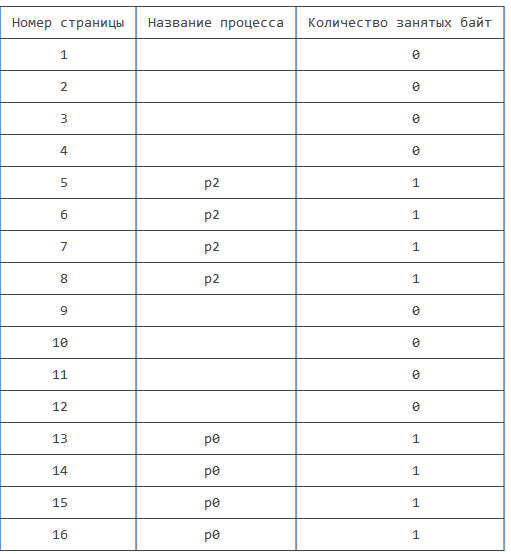


Рисунок 4 – Таблица страниц

На рисунке 5 показано состояние памяти и кол-во полученных/выполненных запросов.

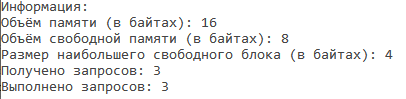


Рисунок 5 – Состояние памяти

**Вывод:** были изучены алгоритмы управления памятью и создан менеджер памяти.

**ПРИЛОЖЕНИЕ А**

(обязательное)

**Текст программы**

Файл *memman.cpp*:

#include <stdio.h>

#include <conio.h>

#include <string.h>

#include <stdlib.h>

#include <iostream>

#include <locale.h>

#include <iomanip>

#include <windows.h>

char bufRus[256];

char\* Rus(const char\* text)

{

CharToOem(text, bufRus);

return bufRus;

}

using namespace std;

void menu()

{

puts(Rus("\n1. Информация"));

puts(Rus("2. Добавить процесс"));

puts(Rus("3. Удалить по названию"));

puts(Rus("4. Таблица страниц"));

puts(Rus("5. Выход из программы"));

printf(Rus("\nВыберите пункт меню: "));

}

void add\_proc(string\* processes,int\* numbers,int &size1, int size, char\* memory[], int &v\_call) {

string m;

printf(Rus("Введите имя процесса: "));

fflush(stdin);

getline(cin, m);

int n;

printf(Rus("Введите количество байт, нужных процессу: "));

scanf("%d", &n);

int ind=0, q = 0;

int check = 0;

for(int i = size - 1; i >= 0; i--)

{

if(processes[i] == "" && numbers[i] == 0)

{

check++;

}else if(processes[i] != "" && numbers[i] != 0){

check = 0;

}

if(check==n){

ind=i;

break;

}

}

if(check == 0 || n > check) { ///////////////////////////12(4,2,2(),2,2())

int k, k\_max=0;

for(int i = size - 1; i >= 0; i--) {

k=0;

for(int j= size - 1; j >= 0; j--) {

if(processes[i] == processes[j] && processes[i] != "") {

k++;

}

}

if(k\_max < k) {

k\_max = k;

ind = i;

}

}

if(n <= k\_max) {

for(int i = ind; i < size, k\_max != 0; i--) {

processes[i] = "";

size1 = size1 + 1;

numbers[i] = 0;

free(memory[i]);

if(k\_max == 1)

ind = i;

k\_max--;

}

} else {

puts(Rus("\nНевозможно добавить процесс"));

return;

}

q = 1;

}

for(int i = ind; i >= 0, n > 0; i++)

{

if(processes[i] == "" && numbers[i] == 0)

{

processes[i] = m;

numbers[i] = 1;

size1 = size1 - 1;

memory[i] = (char\*) malloc (1);

n--;

}

}

if(q == 0) {

puts(Rus("Процесс добавлен"));

v\_call++;

}

else {

puts(Rus("Процесс добавлен с замещением"));

v\_call++;

}

}

void delete\_process(string\* processes,int\* numbers,int &size1, int size, char\* memory[]) {

string m;

printf(Rus("Введите имя процесса, который хотите удалить: "));

fflush(stdin);

getline(cin, m);

int k = 0;

for(int i = 0; i < size; i++) {

if(processes[i] == m && numbers[i] != 0)

{

processes[i] = "";

size1 = size1 + 1;

numbers[i] = 0;

free(memory[i]);

k++;

}

}

if(k == 0)

printf(Rus("\nНазвание не действительно!"));

else

printf(Rus("\nПамять освобождена по названию процесса."));

}

void table(string\* processes,int\* numbers,int size1, int size) {

int i,j;

//Шапка

const char \*S;

printf("%c", 218);

for(i=0;i<16;i++)

printf("%c", 196);

printf("%c", 194);

for(i=0;i<19;i++)

printf("%c", 196);

printf("%c", 194);

for(i=0;i<25;i++)

printf("%c", 196);

printf("%c\n", 191);

printf(Rus("%c Номер страницы %c Название процесса %c Количество занятых байт %c\n"), 179,179,179,179);

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

{

if(j!=size-1)

{

printf("%c", 195);

for(i=0;i<16;i++)

printf("%c", 196);

printf("%c", 197);

for(i=0;i<19;i++)

printf("%c", 196);

printf("%c", 197);

for(i=0;i<25;i++)

printf("%c", 196);

printf("%c\n", 180);

S = processes[j].c\_str();

printf(Rus("%c%8d %c%10s %c%15d %c\n"), 179, j+1, 179, S, 179, numbers[j], 179);

}

else

{

printf("%c", 195);

for(i=0;i<16;i++)

printf("%c", 196);

printf("%c", 197);

for(i=0;i<19;i++)

printf("%c", 196);

printf("%c", 197);

for(i=0;i<25;i++)

printf("%c", 196);

printf("%c\n", 180);

S = processes[j].c\_str();

printf(Rus("%c%8d %c%10s %c%15d %c\n"), 179, j+1, 179, S, 179, numbers[j], 179);

printf("%c", 192);

for(i=0;i<16;i++)

printf("%c", 196);

printf("%c", 193);

for(i=0;i<19;i++)

printf("%c", 196);

printf("%c", 193);

for(i=0;i<25;i++)

printf("%c", 196);

printf("%c", 217);

}

}

}

int main(){

int size=0;//Общий размер памяти

int k, i;//Вариант меню

int check = 0;

int call = 0, v\_call = 0;

printf(Rus("Введите объём памяти в байтах: "));

while(size <= 0)

scanf("%d",&size);

int size1 = size;

string processes[size]; //Имена процессов

int numbers[size]; // Кол-во байт процесса

char\* memory[size]; // Память

for(int i = 0; i < size; i++) {

processes[i] = "";

numbers[i] = 0;

}

while(true)

{

menu();

scanf("%d",&k);

system("cls");

switch(k)

{

case 1:

printf(Rus("Информация:\n"));

printf(Rus("Объём памяти (в байтах): %d\n"),size);

printf(Rus("Объём свободной памяти (в байтах): %d\n"),size1);

for(i = size - 1; i >= 0; i--)

{

if(processes[i] == "" && numbers[i] == 0)

{

check++;

}else if(processes[i] != "" && numbers[i] != 0){

check = 0;

}

}

printf(Rus("Размер наибольшего свободного блока (в байтах): %d\n"),check);

printf(Rus("Получено запросов: %d\n"),call);

printf(Rus("Выполнено запросов: %d\n"),v\_call);

break;

case 2:

add\_proc(processes, numbers, size1, size, memory, v\_call);

call++;

break;

case 3:

delete\_process(processes, numbers, size1, size, memory);

break;

case 4:

table(processes, numbers, size1, size);

break;

case 5:

exit(0);

break;

default:

puts(Rus("Выбран неправильный пункт меню!"));

}

printf("\n");

}

return(0);

}