**МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ**

**УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ**

**ГОМЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ИМЕНИ П. О. СУХОГО**

Факультет автоматизированных и информационных систем

Кафедра «Информационные технологии»

ОТЧЕТ

ПО ЛАБОРАТОРНОЙ РАБОТЕ № 9

по дисциплине «Операционные системы»

Простейшие схемы управления памятью

Выполнил студент

группы ИТИ-11

*Громыко И. В.*

Проверил преподаватель

*Карась О.В.*

Гомель 2022

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

**Задание.**

Разработать программу, реализующую заданный алгоритм 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, объѐм свободной памяти, размер наибольшего свободного блока, количество запросов на выделение памяти, количество удовлетворѐнных запросов (%).

2. Для выделения памяти указывается имя процесса и размер блока. После нажатия на кнопку «ДОБАВИТЬ» память выделяется или выдаѐтся 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 | Задание |
| 6 | **Сегментная схема организации памяти.** Для каждого процесса  создавать 3 сегмента: сегмент стека (1 кб), сегмент кода (1 кб) и сегмент данных. |

**Ход работы**

**Порядок выполнения задания.**

На рисунке 1 показан запуск программы.

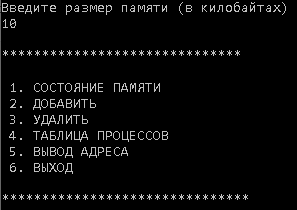


Рисунок 1 – Запуск программы

На рисунке 2 показано выполнения пункта 2 меню.

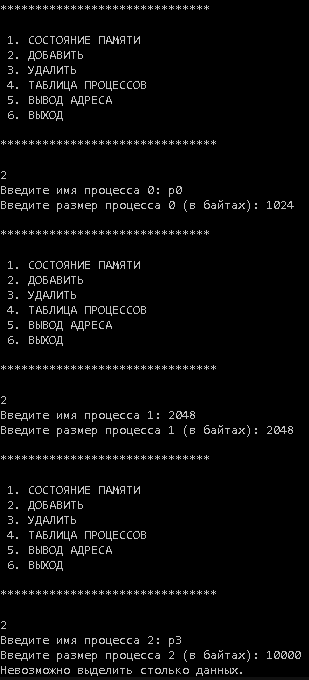


Рисунок 2 – Выполнение 2-го пункта меню

На рисунке 3 показано выполнение пункта 1 меню.

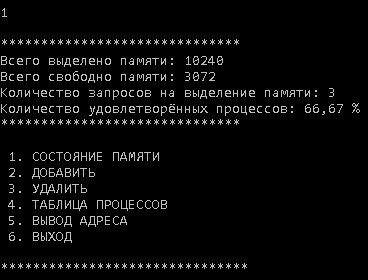


Рисунок 3 – Выполнение 1-го пункта меню

На рисунке 4 показано выполнение пункта 3 меню.

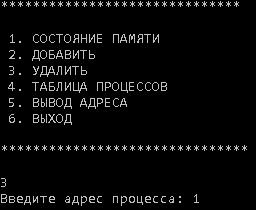


Рисунок 4 – Выполнение 3-го пункта меню

На рисунке 5 показано выполнение пункта 4 меню.

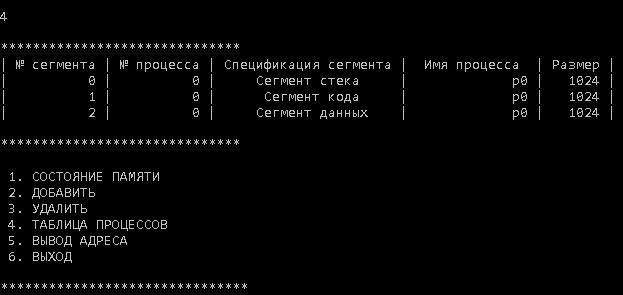


Рисунок 5 – Выполнение 4-го пункта меню

На рисунке 6 показано выполнение пункта 5 меню.

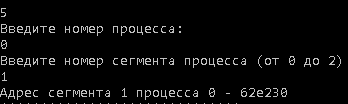


Рисунок 6 – Выполнение 5-го пункта меню

На рисунке 7 показано выполнение пункта 6 меню.

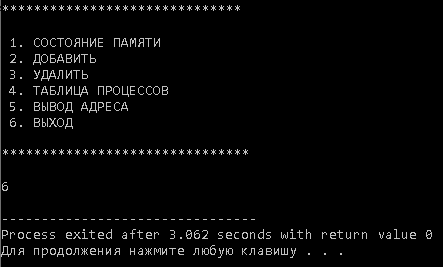


Рисунок 7 – Выполнение 6-го пункта меню

**Вывод:** в процессе работы ознакомились с различными алгоритмами управления памятью, а также разработали программу менеджера памяти.

**ПРИЛОЖЕНИЕ А**

**Листинг программы**

**Листинг программы laba:**

#include <stdio.h>

#include <conio.h>

#include <string.h>

#include <locale.h>

#include <stdlib.h>

int freeSpace=0,fullMemory,processNumber=0;

float cw=0,ca=0;

struct processes

{

int size;

char name[50];

int address;

int segments[3];

};

void addProcess(struct processes process[100])

{

#include <stdio.h>

#include <conio.h>

#include <string.h>

#include <locale.h>

#include <stdlib.h>

int freeSpace=0,fullMemory,processNumber=0;

float cw=0,ca=0;

struct processes

{

int size;

char name[50];

int address;

int segments[3];

};

void addProcess(struct processes process[100])

{

ca++;

int size;

printf("Введите имя процесса %d: ",processNumber);

fflush(stdin);

gets(process[processNumber].name);

printf("Введите размер процесса %d (в байтах): ",processNumber);

scanf("%d",&size);

if (size>freeSpace-2048)

{

puts("Невозможно выделить столько данных.");

}

else

{

cw++;

process[processNumber].size=size;

freeSpace=freeSpace-size-2048;

processNumber++;

}

}

void deleteProcess(struct processes process[100])

{

int adress;

printf("Введите адрес процесса: ");

scanf("%d",&adress);

freeSpace=freeSpace+2048+process[adress].size;

for (int i=adress+1;i<processNumber;i++)

{

process[i-1].size=process[i].size;

strcpy(process[i-1].name,process[i].name);

}

processNumber--;

}

void showTable(struct processes process[100])

{

int segments=0;

printf("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

printf("| № сегмента | № процесса | Спецификация сегмента | Имя процесса | Размер | \n");

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

{

for (int k=0;k<3;k++)

{

printf("| %10d |",segments);

printf(" %10d |",i);

if (k==0)

{

printf(" Сегмент стека |");

printf(" %14s |",process[i].name);

printf(" 1024 |");

}

else if (k==1)

{

printf(" Сегмент кода |");

printf(" %14s |",process[i].name);

printf(" 1024 |");

}

else

{

printf(" Сегмент данных |");

printf(" %14s |",process[i].name);

printf(" %6d |",process[i].size);

}

printf("\n");

segments++;

}

}

}

void showData()

{

float percent = (cw/ca)\*100;

printf("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

printf("Всего выделено памяти: %d",fullMemory);

printf("\nВсего свободно памяти: %d",freeSpace);

printf("\nКоличество запросов на выделение памяти: %.0f",ca);

printf("\nКоличество удовлетворённых процессов: %.2f %%",percent);

}

void showAddress(struct processes process[100])

{

int segment;

int number;

puts("Введите номер процесса: ");

scanf("%d",&number);

puts("Введите номер сегмента процесса (от 0 до 2)");

scanf("%d",&segment);

int \*a = &process[number].segments[segment];

printf("Адрес сегмента %d процесса %d - %x",segment,number,a);

}

int main()

{

setlocale(LC\_ALL,"RUS");

struct processes process[100];

puts("Введите размер памяти (в килобайтах)");

scanf("%d",&freeSpace);

freeSpace\*=1024;

fullMemory=freeSpace;

int t=1;

int menu;

while (t)

{

puts("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

puts(" 1. СОСТОЯНИЕ ПАМЯТИ");

puts(" 2. ДОБАВИТЬ");

puts(" 3. УДАЛИТЬ");

puts(" 4. ТАБЛИЦА ПРОЦЕССОВ");

puts(" 5. ВЫВОД АДРЕСА");

puts(" 6. ВЫХОД\n");

puts("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

scanf("%d",&menu);

switch(menu)

{

case 1:

showData();

break;

case 2:

addProcess(process);

break;

case 3:

deleteProcess(process);

break;

case 4:

showTable(process);

break;

case 5:

showAddress(process);

break;

case 6:

t=0;

break;

default:

printf("Неверно введённые данные\n");

}

}

}