**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»**

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

Выполнил: студент гр. ИТ-22

Реуцкий А.А.

Принял: преподаватель Соболев Д.В.

**Гомель 2016**

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

Разработать программу, реализующую заданный алгоритм выделения памяти.

Менеджер памяти должен: 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 (%).

|  |
| --- |
| **Свопинг.** Выгружается процесс, занимающий наименьший объём памяти. |

Листинг программы:

using System;

using System.Collections.Generic;

using System.Text;

namespace ConsoleApplication1

{

class D

{

public static int V, V\_sv, n, k\_a = 0, k\_success = 0;

public static Process[] proc;

public static Process[] sw = new Process[20];

public static void Create()

{

int t;

Console.Write("\nВведите количество разделов памяти: ");

n = Convert.ToInt32(Console.ReadLine());

proc = new Process[n];

Console.Write("Введите объём раздела памяти: ");

t = Convert.ToInt32(Console.ReadLine());

V = n \* t;

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

proc[i] = new Process(t);

V\_sv = V;

for (int i = 0; i < sw.Length; i++)

sw[i] = new Process(t);

}

public static int Swap()

{

Console.Clear();

int min = V;

int r = -1;

for (int i = 0; i < proc.Length; i++)

{

if (proc[i].Mem\_p != 0 && proc[i].Mem\_p < min) { min = proc[i].Mem\_p; r = i; }

}

if (r != -1)

{

int g = 0, ff = 0;

while (g < sw.Length && ff == 0)

{

if (sw[g].Name\_p == null)

{

sw[g].Name\_p = proc[r].Name\_p;

sw[g].Mem\_p = proc[r].Mem\_p;

sw[g].data = proc[r].data;

ff = 1;

}

g++;

}

proc[r].Mem\_p = 0;

proc[r].Name\_p = null;

proc[r].data = null;

V\_sv += proc[r].V0;

Console.WriteLine("Процесс, занимающий наименьший объём памяти выгружен");

}

else { Console.WriteLine("Процессов не обнаружено"); Console.ReadKey(); }

return r;

}

public static void add()

{

Console.Clear();

string name;

int vv;

Console.Write("\nВведите имя процесcа: ");

name = Console.ReadLine();

Console.Write("\nВведите объём памяти требуемой данному процессу(Кб): ");

vv = Int32.Parse(Console.ReadLine());

k\_a++;

if (vv > V) Console.WriteLine("Ошибка: объём всей памяти меньше объёма памяти требуемой процессу");

else

{

if (vv <= proc[0].V0)

{

int f = 0, i = 0;

while (i < proc.Length && f == 0)

{

if (proc[i].Name\_p == null)

{

proc[i].Name\_p = name;

proc[i].Mem\_p = vv;

k\_success++; f++;

proc[i].data = new string[vv];

V\_sv = V\_sv - proc[i].V0;

}

i++;

}

if (f == 0)

{

int r = Swap();

proc[r].Name\_p = name;

proc[r].Mem\_p = vv;

k\_success++;

proc[r].data = new string[vv];

V\_sv = V\_sv - proc[r].V0;

}

}

else { Console.WriteLine("Ошибка: объём раздела меньше объёма требуемой памяти"); Console.ReadKey(); }

}

}

public static void Vyvod()

{

Console.WriteLine("\nИнформация о процессах\n");

int f = 0;

for (int i = 0; i < proc.Length; i++)

{

if (proc[i].Mem\_p != 0)

{ Console.WriteLine("Процесс: {0,-9}Объём памяти(Кб): {1,-5} Номер блока памяти: {2} ", proc[i].Name\_p, proc[i].Mem\_p, i + 1); f++; }

}

if (f == 0) Console.WriteLine("Нет процессов.");

}

public static void SW()

{

Console.WriteLine("\nВыгруженные процессы\n");

int f = 0;

for (int i = 0; i < sw.Length; i++)

{

if (sw[i].Mem\_p != 0)

{ Console.WriteLine("Процесс: {0,-9}Объём памяти(Кб): {1,-5} ", sw[i].Name\_p, sw[i].Mem\_p); f++; }

}

if (f == 0) Console.WriteLine("Нет процессов.");

}

public static void MaxR()

{

int temp;

int max = -1;

for (int i = 0; i < proc.Length; i++)

{

temp = proc[i].V0 - proc[i].Mem\_p;

if (temp > max) max = temp;

}

if (max != -1) Console.WriteLine("Размер наибольшего свободного блока: " + max + " Кб");

}

public static void KolPercent()

{

if (k\_a != 0)

{

float k = 100 \* (float)k\_success / k\_a;

Console.WriteLine("Количество удовлетворённых запросов: {0,-4:f1}%", k);

}

}

}

class Process

{

int mem\_p, V\_razd;

string name\_p;

public Process(int V\_r) { V\_razd = V\_r; }

public string[] data;

public int V0

{

get { return V\_razd; }

set { V\_razd = value; }

}

public int Mem\_p

{

get { return mem\_p; }

set { mem\_p = value; }

}

public string Name\_p

{

get { return name\_p; }

set { name\_p = value; }

}

}

class Program

{

static void Main(string[] args)

{

Console.Title = "Схема с фиксированными разделами ";

int k = 12;

D.Create();

do

{

Console.Clear();

Console.WriteLine("Полный объём памяти: " + D.V + " Кб");

Console.WriteLine("Количество разделов: " + D.n);

Console.WriteLine("Объём свободной памяти: " + D.V\_sv + " Кб");

Console.WriteLine("Количество запросов на выделение памяти: " + D.k\_a);

D.KolPercent();

D.MaxR();

D.Vyvod();

D.SW();

Console.WriteLine("\n\t╔═══════════════════════════════════════════════════════════╗");

Console.WriteLine("\t║ Главное меню ║");

Console.WriteLine("\t║ Выберите действие: ║");

Console.WriteLine("\t╟─────┬─────────────────────────────────────────────────────╢");

Console.WriteLine("\t║ 1 │ Добавить процесс ║");

Console.WriteLine("\t╟─────┼─────────────────────────────────────────────────────╢");

Console.WriteLine("\t║ 2 │ Свопинг ║");

Console.WriteLine("\t╟─────┼─────────────────────────────────────────────────────╢");

Console.WriteLine("\t║ 0 │ Завершить работу программы ║");

Console.WriteLine("\t╚═══════════════════════════════════════════════════════════╝");

Console.Write("\n\t\t\t Ваш выбор(0-6): ");

k = Convert.ToInt32(Console.ReadLine());

switch (k)

{

case 1: D.add(); break;

case 2: D.Swap(); break;

case 0: Console.Clear(); Console.WriteLine("Выход..."); break;

default: break;

}

}

while (k != 0);

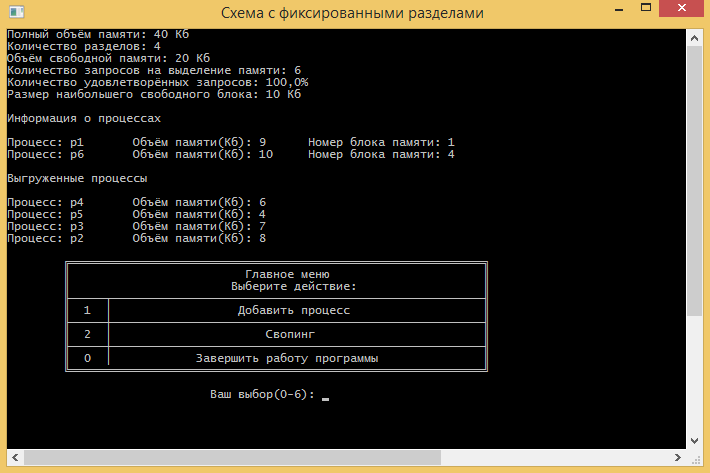
Console.ReadKey();

}

}

}

Результат выполнения программы:



Вывод:разработал программу моделирующую реализацию свопинга.