МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ

ГОМЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ИМЕНИ П. О. СУХОГО

Факультет автоматизированных и информационных систем

Кафедра «Информационные технологии»

Дисциплина «Операционные системы»

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ № 9

**«Простейшие схемы управления памятью»**

Выполнил: студент гр. ИТП-11

Демиденко Д.В.

#### Принял: 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, 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.

Алгоритм: **9.Схема с фиксированными разделами с общей очередью процессов. (Количество разделов и их размер задать с клавиатуры).**

Запустим программу и введём количество памяти.

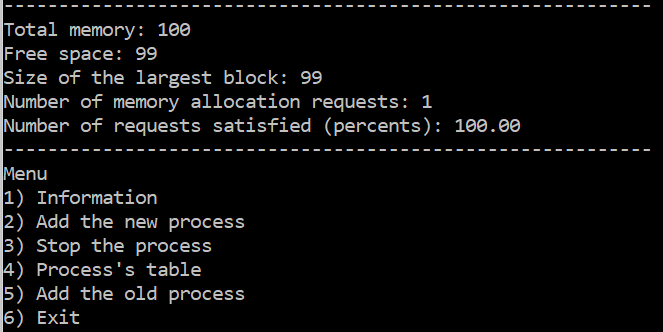


Рисунок 1 – Вывод основной информации

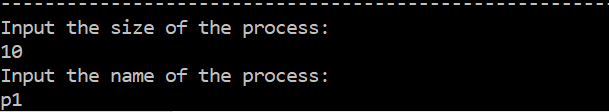


Рисунок 2 – Ввод процессов

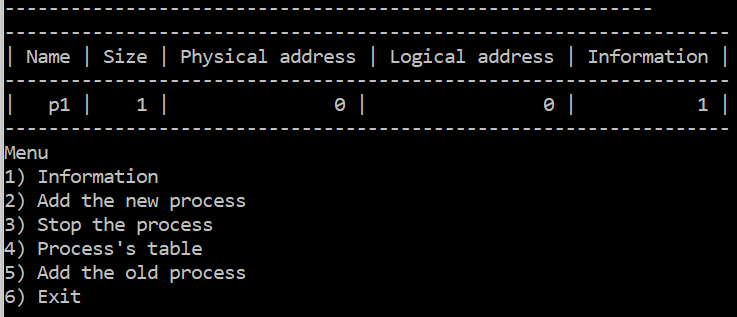


Рисунок 3 – Вывод информации о процессах

**Вывод:** в процессе выполнения данной лабораторной работы были изучены основные алгоритмы управления памятью, а также был разработан менеджер памяти.

**ПРИЛОЖЕНИЕ А**

Листинг программы

#include <stdio.h>

#include <string.h>

#include <stdlib.h>

#include <windows.h>

struct condition

{

int sizeofblocks;

int condition;

int number;

}array[1000];

struct operations

{

char name[1000];

int size;

int condition;

int N;

}oper[1000];

int main()

{

int totalmemory,activtotalmemory,numberofblocks,freememory,largestmemoryblock,numberofprocess=0,success=0;

int i,n,g=0,col,flag;

float successful=100,config=100;

printf("Input total memory for process: ");

scanf("%d",&totalmemory);

activtotalmemory=totalmemory;

freememory=totalmemory;

printf("Input number of memory blocks for process: ");

scanf("%d",&numberofblocks);

while(numberofblocks>totalmemory)

{

printf("Input number of memory blocks for process <=%d: ",totalmemory);

scanf("%d",&numberofblocks);

}

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

{

array[i].sizeofblocks=1;

array[i].condition=1;

activtotalmemory-=array[i].sizeofblocks;

}

for(i=0;i<numberofblocks && activtotalmemory!=0;i++)

{

activtotalmemory+=array[i].sizeofblocks;

if(i==numberofblocks-1)

{

array[i].sizeofblocks=activtotalmemory;

}

else

{

printf("Input size of memory block for process,but <=%d: ",activtotalmemory);

scanf("%d",&n);

if(activtotalmemory-n<0)

{

activtotalmemory-=array[i].sizeofblocks;

i--;

}

else

{

array[i].sizeofblocks=n;

activtotalmemory-=n;

}

}

}

while(g!=5)

{

printf("\n--------------------------------------------");

printf("\n1)Total information\n2)Information about process\n3)Add new process\n4)Delete process\n5)Exit\n");

printf("\nYour choice: ");

scanf("%d",&g);

switch(g)

{

case 1:

{

largestmemoryblock=array[0].sizeofblocks;

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

{

if(largestmemoryblock<array[i].sizeofblocks)

{

largestmemoryblock=array[i].sizeofblocks;

}

}

printf("\n---------------------------------");

printf("\nTotal memory: %d",totalmemory);

printf("\nFree memory: %d",freememory);

printf("\nLargest memory block: %d",largestmemoryblock);

printf("\nFree blocks: %.2f%%",config);

printf("\nNumber of blocks: %d",numberofblocks);

printf("\nSize of blocks: ");

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

{

printf("%d ",array[i].sizeofblocks);

}

printf("\n---------------------------------");

break;

}

case 2:

{

printf("\n------------------------------------------------------------------------------");

printf("\n| n | NameProcess |NumberofProcess| sizeofblocks | ON |");

printf("\n------------------------------------------------------------------------------");

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

{

if(array[i].condition==1)

{

printf("\n|%3d.| | |%22d| - |",i+1,array[i].sizeofblocks);

}

else if(array[i].condition==0)

{

printf("\n|%3d.|%20s|%15d|%22d| + |",i+1,oper[array[i].number].name,oper[i].N,array[i].sizeofblocks);

}

printf("\n------------------------------------------------------------------------------");

}

printf("\n\n");

break;

}

case 3:

{

printf("\n---------------------------------");

printf("\nInput name for process: ");

scanf("%s",&oper[numberofprocess].name);

flag=1;

for(i=1;i<=numberofprocess && flag==1;i++)

{

if(oper[i].condition==1)

{

if(strcmp(oper[i].name,oper[numberofprocess].name)==0 && oper[i].condition==1)

{

flag=2;

}

while(flag==2)

{

col=1;

printf("\nThis name exist, create new name for process: ");

scanf("%s",&oper[numberofprocess].name);

for(i=0;i<numberofprocess && col==1;i++)

{

if(strcmp(oper[i].name,oper[numberofprocess].name)==0)

{

col=2;

}

}

if(col==1)

{

flag=0;

}

}

}

}

printf("\nInput size for process: ");

scanf("%d",&oper[numberofprocess].size);

largestmemoryblock=0;

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

{

if(largestmemoryblock<array[i].sizeofblocks && array[i].condition==1)

{

largestmemoryblock=array[i].sizeofblocks;

n=i;

}

}

if(largestmemoryblock>=oper[numberofprocess].size)

{

oper[numberofprocess].condition=1;

freememory-=oper[numberofprocess].size;

array[n].condition=0;

oper[numberofprocess].N=n;

array[n].number=numberofprocess;

success++;

printf("\nSuccess add\n");

}

else

{

printf("\nThere is no such memory block in which this process fits");

}

printf("\n---------------------------------");

n=0;

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

{

if(array[i].condition==1)

{

n++;

}

}

numberofprocess++;

config=n\*100/numberofblocks;

successful=success\*100/numberofprocess;

break;

}

case 4:

{

printf("\n----------------------------------------------");

printf("\nInput number of process which u want to OFF: ");

scanf("%d",&n);

if(n<numberofprocess)

{

freememory+=oper[n].size;

strcpy(oper[n].name,"");

oper[n].condition=0;

array[oper[n].N].condition=1;

oper[n].condition=0;

}

else

{

printf("Process doesn't exist");

}

n=0;

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

{

if(array[i].condition==1)

{

n++;

}

}

config=n\*100/numberofblocks;

printf("\n---------------------------------");

break;

}

}

}

return 0;

}