МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ

ГОМЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ИМЕНИ П. О. СУХОГО

Учреждение образования Республики Беларусь

«Гомельский государственный технический

университет им. П.О.Сухого»

Кафедра «Автоматизированных и информационных систем»

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ № 10

по дисциплине «Операционные системы»

на тему: «Управление виртуальной памятью»

Выполнил студент гр. ИТИ-11:

Бут-Гусаим Д.Г. Принял преподаватель-стажёр: Карась О.В.

Гомель 2022

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

**Индивидуальные задания**

Разработать программу, реализующую заданный алгоритм выделения памяти.

Менеджер памяти должен:

1. По запросу процесса выделять память, согласно заданного алгоритма (таблица 1). На экран должна выводиться следующая информация о состоянии памяти: объем памяти, объем 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.

|  |  |
| --- | --- |
| Вариант | Задание |
| 2 | Локальное размещение с динамическим увеличением количества выделенных страниц при количестве запросов к страницам больше установленного порога. Алгоритм замещения – Random. Замещается случайная страница. |

Таблица 1 – Задание по варианту

**Порядок выполнения:**

В ходе выполнения данной лабораторной работы была создана программа, реализующая алгоритм замещения в виртуальной памяти. На рисунке 1 показано начальное окно программы, в котором необходимо ввести один из вышеперечисленных пунктов.

На рисунке 1 показан основной пункт меню.

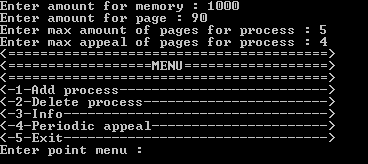


Рисунок 1 – Запускное окно программы

На рисунке 2 показан вывод информации о свободной и занятой памяти, а так же о процессах.

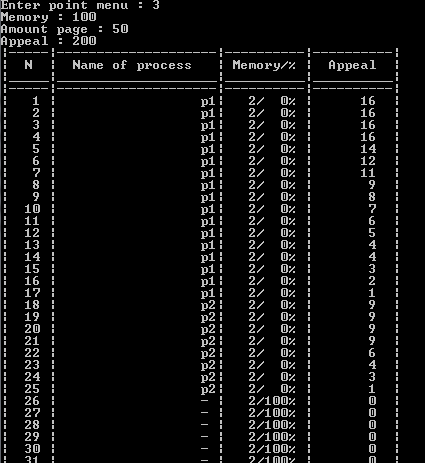


Рисунок 2 – Вывод информации о свободной и занятой памяти, а также информации о процессах

В ОС используется несколько методов выделения файлу дискового пространства. Для каждого из методов запись в директории, соответствующая символьному имени файла, содержит указатель, следуя которому можно найти все блоки данного файла. Основная память представляет собой упорядоченный массив однобайтовых ячеек, каждая из которых имеет свой уникальный адрес (номер). Вторичную память (это главным образом диски) также можно рассматривать как одномерное линейное адресное пространство, состоящее из последовательности байтов. В отличие от оперативной памяти, она является энергонезависимой, имеет существенно большую емкость и используется в качестве расширения основной памяти. Многоуровневую схему используют следующим образом. Информация, которая находится в памяти верхнего уровня, обычно хранится также на уровнях с большими номерами. Если процессор не обнаруживает нужную информацию на i-м уровне, он начинает искать ее на следующих уровнях. Когда нужная информация найдена, она переносится в более быстрые уровни

**Вывод:** в ходе лабораторной работы были изучены алгоритмы замещения и размещения файлов на страницах виртуальной памяти.

**ПРИЛОЖЕНИЕ А:**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

struct Page{

int memory;

int free\_memory;

char proc\_name[20];

int appeal;

bool active;

};

void FirstFilling(Page \*pg,int col,int size){

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

(pg+i)->memory = size;

(pg+i)->free\_memory = size;

(pg+i)->proc\_name[0] = '-';

(pg+i)->proc\_name[1] = ' ';

(pg+i)->proc\_name[2] = '\0';

(pg+i)->appeal = 0;

(pg+i)->active = false;

}

}

bool CheckForNamb(char \*string)

{

for(int i = 0; i < strlen(string); i++)

{

if(string[i] < 48 || string[i] > 57) return false;

}

return true;

}

int ReadValueInt()

{

bool right = true;

char string[255];

while(right)

{

scanf("%s",string);

if(strlen(string)<10 && CheckForNamb(string))

right = false;

else

printf("Enter number < 147483647 !\n");

}

return atoi(string);

}

int SuchFree(Page \*p,int col){

int i = -1;

bool such = true;

while(i < col && such){

i++;

if(!((p+i)->active)) such = false;

}

return(i);

}

bool In(int \* m,int col,int z){

bool r = true;

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

if(m[i] == z) r = false;

}

return(r);

}

int Reb(Page \*p,int col,int size){

int c=0;

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

if((p+i)->free\_memory == size){

(p+i)->active = false;

c++;

}

}

return(c);

}

int main(){

Page pag[100];

int max\_appeal,r,p,menu = 1,memory,free\_memory,max\_page,amt\_page,free\_page,appeal = 0,successful\_appeal = 0,pr\_size,page[100],amt\_mas\_page,shud\_page,max\_appel,pr\_appeal;

char pr\_name[20];

bool add\_page,tmp;

printf("Enter amount for memory : ");

memory = ReadValueInt();

free\_memory = memory;

printf("Enter amount for page : ");

amt\_page = ReadValueInt();

free\_page = amt\_page;

FirstFilling(pag,amt\_page,memory/amt\_page);

printf("Enter max amount of pages for process : ");

max\_page = ReadValueInt();

printf("Enter max appeal of pages for process : ");

max\_appeal = ReadValueInt();

while(menu != 5){

printf("<========================================>\n");

printf("<==================MENU==================>\n");

printf("<========================================>\n");

printf("<-1-Add process-------------------------->\n");

printf("<-2-Delete process----------------------->\n");

printf("<-3-Info--------------------------------->\n");

printf("<-4-Periodic appeal---------------------->\n");

printf("<-5-Exit--------------------------------->\n");

printf("Enter point menu : ");

menu = ReadValueInt();

switch(menu){

case 1:{

printf("Enter name of process : \n");

scanf("%s",&pr\_name);

printf("Enter size of process : \n");

pr\_size = ReadValueInt();

if(pr\_size > 0){

amt\_mas\_page = max\_page;

free\_page = Reb(pag,amt\_page,memory/amt\_page);

for(int i = 0; i < amt\_mas\_page; i++){

if(SuchFree(pag,amt\_page) < amt\_page){

page[i] = SuchFree(pag,amt\_page);

pag[SuchFree(pag,amt\_page)].active = true ;

}else{

p = rand() % amt\_page;

if(In(page,amt\_mas\_page,p)){page[i] = p;}else{i--;}

}

printf("Form local pages : page[%d] = %d\n",i,page[i]);

}

r = 0;

pr\_appeal = 0;

while(pr\_size != 0){

appeal++;

printf("Wark with page : page[%d] = %d\n",r,page[r]);

if(pr\_size > (memory / amt\_page)){

pag[page[r]].memory = (memory / amt\_page);

pag[page[r]].free\_memory = 0;

pr\_size -= (memory / amt\_page);

}else{

pag[page[r]].memory = pr\_size;

pag[page[r]].free\_memory = (memory / amt\_page) - pr\_size;

pr\_size = 0;

}

strcpy(pag[page[r]].proc\_name,pr\_name);

pag[page[r]].appeal++;

pr\_appeal++;

if(pr\_appeal >= max\_appeal){

printf("Add page\n");

add\_page = true;

pr\_appeal = 0;

amt\_mas\_page++;

while(add\_page && amt\_mas\_page <= amt\_page){

if(SuchFree(pag,amt\_page) < amt\_page){p = SuchFree(pag,amt\_page);pag[SuchFree(pag,amt\_page)].active = true;}else{p = rand() % amt\_page;};

printf("Form new page = %d, amt page = %d",p,amt\_mas\_page);

if(In(page,amt\_mas\_page,p)){page[amt\_mas\_page-1] = p; add\_page = false;}

printf("\n");

}

}

r++;

if(r >= amt\_mas\_page) r = 0;

}

}else{

printf("Size > 0\n");

}

break;

}

case 2:{

printf("Enter delete name of process : \n");

scanf("%s",&pr\_name);

tmp = false;

for(int i = 0 ; i < amt\_page ; i++)

{

if(!strcmp(pag[i].proc\_name,pr\_name)){

pag[i].memory = memory / amt\_page;

pag[i].free\_memory = memory / amt\_page;

pag[i].proc\_name[0] = '-';

pag[i].proc\_name[1] = ' ';

pag[i].proc\_name[2] = '\0';

pag[i].appeal = 0;

pag[i].active = false;

tmp = true ;

}

}

if(tmp){

printf("Process %s delete!\n",pr\_name);

}else{

printf("This process not found!\n");

}

break;

}

case 3:{

printf("Memory : %d\n",memory);

printf("Amount page : %d\n",amt\_page);

printf("Free page : %d\n",free\_page);

printf("Appeal : %d\n",appeal);

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

printf("| N | Name of process | Memory/%% | Appeal |\n");

printf("|\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_|\n");

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

for(int i = 0; i < amt\_page; i++){

printf("| %3d |%20s|%4d/%3.0f%% | %6d |\n",i+1,pag[i].proc\_name,pag[i].memory,(pag[i].free\_memory/(float)(memory / amt\_page))\*100,pag[i].appeal);

}

printf("|\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_|\n");

break;

}

case 4:{

printf("Enter amount appeal : ");

p = ReadValueInt();

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

r = rand() % amt\_page;

pag[r].appeal++;

appeal++;

printf("Appeal to page N = %d\n",r);

}

break;

}

case 5:{

printf("Exit...\n");

break;

}

default:{

printf("Error : Not right argument of menu!\n");

break;

}

}

}

return 0;

}