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. ИТИ-11:

Бут-Гусаим Д.Г. Принял преподаватель-стажёр: 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.

Страничная организация памяти. Размер страницы n.

Запустим программу и введём количество памяти.

На рисунке 1 показан основной пункт меню.

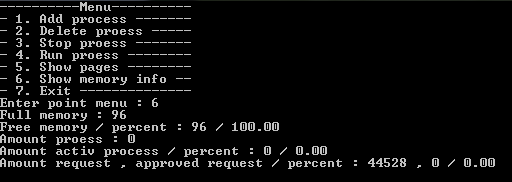


Рисунок 1 – Вывод основной информации

На рисунке 2 показана работа первого пунткта программы.

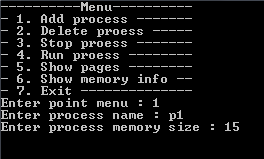


Рисунок 2 – Ввод процессов

На рисунке показана работа пятого пункта меню.

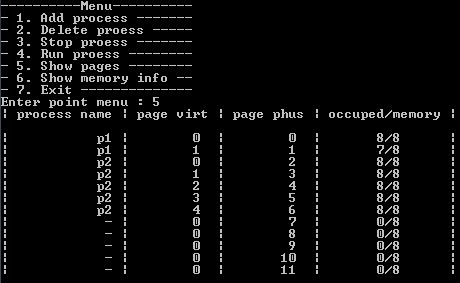


Рисунок 3 – Вывод информации о процессах

Часть ОС, которая отвечает за управление памятью, называется менеджером памяти. Запоминающие устройства компьютера разделяют, как минимум, на два уровня: основную (главную, оперативную, физическую) и вторичную (внешнюю) память.

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

**ПРИЛОЖЕНИЕ А:**

#include<stdio.h>

#include<math.h>

#include<stdlib.h>

#include<string.h>

struct process

{

char name[5];

int size;

bool activ;

int amtPage;

int tabl[100];

};

struct page

{

int virt;

int phus;

bool free;

int memory\_occ;

char proc\_name[5];

};

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 main()

{

page page[100];

process proc[100];

int n,i,j,memory,PowTwo,amtPage,menu = 0,amtP = 0,memory\_free,req,req\_true=0,g;

char name[5];

printf("Enter memory size : ");

memory = ReadValueInt();

printf("Enter power of two : ");

PowTwo = ReadValueInt();

PowTwo = pow(2,PowTwo);

amtPage = memory / PowTwo;

printf("Namber of pages : %d\n",amtPage);

memory = PowTwo\*amtPage;

memory\_free = PowTwo\*amtPage;

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

{

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

page[i].proc\_name[1] = '\0';

page[i].virt = 0;

page[i].phus = i;

page[i].memory\_occ = 0;

page[i].free = true;

}

fflush(stdin);

getchar();

while( menu != -1 )

{

system("cls");

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

printf("- 1. Add process -------\n");

printf("- 2. Delete proess -----\n");

printf("- 3. Stop proess -------\n");

printf("- 4. Run proess --------\n");

printf("- 5. Show pages --------\n");

printf("- 6. Show memory info --\n");

printf("- 7. Exit --------------\n");

printf("Enter point menu : ");

menu = ReadValueInt();

switch(menu)

{

case 1:

{

req++;

printf("Enter process name : ");

scanf("%s",&proc[amtP].name);

printf("Enter process memory size : ");

proc[amtP].size = ReadValueInt();

if(memory\_free < proc[amtP].size || proc[amtP].size < 1)

{

proc[amtP].size < 1 ? printf("Process size > 0") : printf("Out of memory");

}

else

{

req\_true++;

if(proc[amtP].size % PowTwo == 0)

{

proc[amtP].amtPage = proc[amtP].size / PowTwo;

i = 0;

j = 0;

while( i < proc[amtP].amtPage )

{

if (page[j].free)

{

strcpy(page[j].proc\_name,proc[amtP].name);

page[j].memory\_occ = PowTwo;

page[j].virt = i;

page[j].phus = j;

page[j].free = false;

memory\_free -= PowTwo;

proc[amtP].tabl[i] = j;

j++;

i++;

}

else

{

j++;

}

}

}

else

{

proc[amtP].amtPage = ( proc[amtP].size / PowTwo ) + 1;

i = 0;

j = 0;

while( i < proc[amtP].amtPage )

{

if (page[j].free)

{

strcpy(page[j].proc\_name,proc[amtP].name);

page[j].memory\_occ = PowTwo;

page[j].virt = i;

page[j].phus = j;

page[j].free = false;

memory\_free -= PowTwo;

proc[amtP].tabl[i] = j;

if( i == proc[amtP].amtPage - 1 )

{

page[j].memory\_occ = proc[amtP].size % PowTwo;

}

j++;

i++;

}

else

{

j++;

}

}

}

proc[amtP].activ = true;

amtP++;

}

break;

}

case 2:

{

printf("Enter process name : ");

scanf("%s",&name);

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

{

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

{

if(proc[i].activ)

{

for( j = 0 ; j < proc[i].amtPage ; j++ )

{

page[proc[i].tabl[j]].proc\_name[0] = '-';

page[proc[i].tabl[j]].proc\_name[1] = '\0';

page[proc[i].tabl[j]].virt = 0;

page[proc[i].tabl[j]].phus = proc[i].tabl[j];

page[proc[i].tabl[j]].memory\_occ = 0;

page[proc[i].tabl[j]].free = true;

}

memory\_free += PowTwo \* proc[i].amtPage;

}

printf("Process %s delete\n",proc[i].name);

for( j = i ; j < amtP ; j++ )

{

proc[j] = proc[j+1];

}

amtP--;

i--;

}

}

break;

}

case 3:

{

printf("Enter process name : ");

scanf("%s",&name);

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

{

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

{

for( j = 0 ; j < proc[i].amtPage ; j++ )

{

page[proc[i].tabl[j]].proc\_name[0] = '-';

page[proc[i].tabl[j]].proc\_name[1] = '\0';

page[proc[i].tabl[j]].virt = 0;

page[proc[i].tabl[j]].phus = proc[i].tabl[j];

page[proc[i].tabl[j]].memory\_occ = 0;

page[proc[i].tabl[j]].free = true;

}

memory\_free += PowTwo \* proc[i].amtPage;

proc[i].activ = false;

printf("Process %s stopped\n",proc[i].name);

}

}

break;

}

case 4:

{

printf("Enter process name : ");

scanf("%s",&name);

for( g = 0 ; g < amtP ; g++)

{

if( strcmp( proc[g].name , name ) == 0 )

{

if(memory\_free < proc[g].size)

{

printf("Out of memory");

}

else

{

printf("Run process %s\n",proc[g].name);

if(proc[g].size % PowTwo == 0)

{

i = 0;

j = 0;

while( i < proc[g].amtPage )

{

if (page[j].free)

{

strcpy(page[j].proc\_name,proc[g].name);

page[j].memory\_occ = PowTwo;

page[j].virt = i;

page[j].phus = j;

page[j].free = false;

memory\_free -= PowTwo;

proc[g].tabl[i] = j;

j++;

i++;

}

else

{

j++;

}

}

}

else

{

i = 0;

j = 0;

while( i < proc[g].amtPage )

{

if (page[j].free)

{

strcpy(page[j].proc\_name,proc[g].name);

page[j].memory\_occ = PowTwo;

page[j].virt = i;

page[j].phus = j;

page[j].free = false;

memory\_free -= PowTwo;

proc[g].tabl[i] = j;

if( i == proc[g].amtPage - 1 )

{

page[j].memory\_occ = proc[g].size % PowTwo;

}

j++;

i++;

}

else

{

j++;

}

}

}

proc[g].activ = true;

}

}

}

break;

}

case 5:

{

printf("| process name | page virt | page phus | occuped/memory |\n\n");

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

{

printf("| %12s | %7d | %7d | %4d/%-4d |\n",page[i].proc\_name,page[i].virt,page[i].phus,page[i].memory\_occ,PowTwo);

}

break;

}

case 6:

{

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

printf("Free memory / percent : %d / %.2f \n",memory\_free,(memory\_free/(float)memory)\*100);

printf("Amount proess : %d\n",amtP);

g = 0;

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

{

if( proc[i].activ )

{

g++;

}

}

if( amtP ){printf("Amount activ process / percent : %d / %.2f \n",g,(g/(float)amtP)\*100);}else{printf("Amount activ process / percent : 0 / 0.00 \n");}

if( req ){printf("Amount request , approved request / percent : %d , %d / %.2f \n",req,req\_true,(req\_true/(float)req)\*100);}else{printf("Amount request , approved request / percent : 0 , 0 / 0.00 \n");}

break;

}

case 7:

{

printf("Finsh work\n");

menu = -1;

break;

}

default:

{

printf("Not correct point\n");

break;

}

}

fflush(stdin);

getchar();

}

} }