**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

Леди Джойс

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.



Ход работы



Рисунок 1 – Добавление процесса

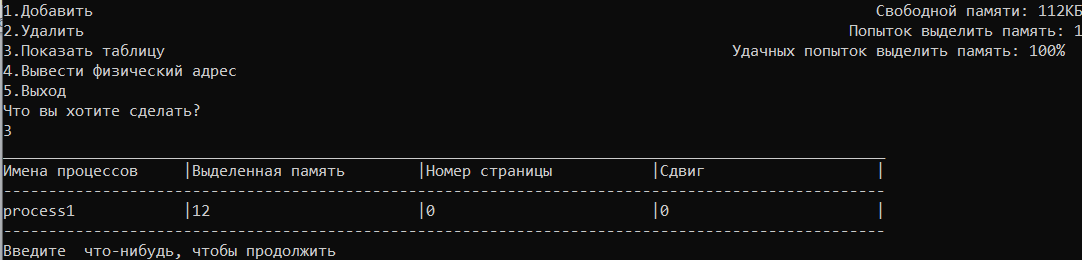


Рисунок 2 – Показ таблицы процессов

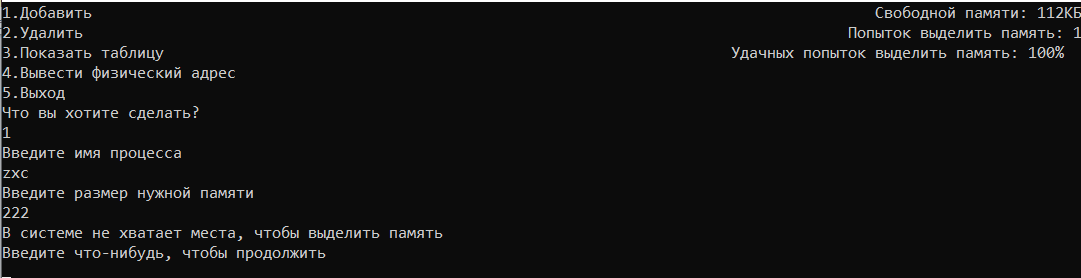


Рисунок 3 – Неудачная попытка добавления процесса



Рисунок 4 – Вывод физического адреса



Рисунок 5 – Удаление процесса



Рисунок 6 – Выход

Вывод: в ходе лабораторной работы изучили различные схемы организации памяти.

**ПРИЛОЖЕНИЕ А**

Листинг программы lab 9.cpp

#include<iostream>

int numLength(int n)

{

int s = 0;

if(n == 0)

return 1;

while(n > 0){

n = n / 10;

s++;

}

return s;

}

void makeSpaces(int n){

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

std::cout << " ";

}

struct data{

std::string name;

int size;

int page\_;

int padding\_;

};

void giveInfo(int available\_memory, int attempts, int attempts\_1){

float attempts\_1\_percent;

int translate;

if(attempts != 0){

attempts\_1\_percent = (float)attempts\_1 \* 100 / (float)attempts;

translate = attempts\_1\_percent \* 10;

attempts\_1\_percent = (float)translate / 10;

}

else

attempts\_1\_percent = 0;

std::cout << "1.To add");

makeSpaces(90 - numLength(available\_memory));//ajouter

std::cout << "Free memory: " << available\_memory << "ÊÁ\n" << "2.Delete";

makeSpaces(86 - numLength(attempts));

std::cout << "Ïîïûòîê âûäåëèòü ïàìÿòü: " << attempts << "\n3 Show the array";

makeSpaces(66 - numLength(attempts\_1\_percent));

std::cout << "Óäà÷íûõ ïîïûòîê âûäåëèòü ïàìÿòü: " << attempts\_1\_percent << "%\n4.Âûâåñòè ôèçè÷åñêèé àäðåñ\n5.Exit\n";

}

void showAddress(int page, int padding, data \*a, int prikol){

int i = 0;

std::string go\_next;

bool found = true;

while(i < prikol && found){

if((a+i)->page\_ == page && (a+i)->padding\_ == padding)

found = false;

i++;

}

std::cout << a+i << "\nÂâåäèòå ÷òî-íèáóäü, ÷òîáû ïðîäîëæèòü\n";

std::cin >> go\_next;

}

void showTable(data \*a, int prikol){

for(int i = 0; i < 98;i++)

std::cout << "\_";

std::cout << "\n";

std::cout << "Èìåíà ïðîöåññîâ |" << "Âûäåëåííàÿ ïàìÿòü" << " |Íîìåð ñòðàíèöû" << " |Ñäâèã |\n";

for(int i = 0;i < 98;i++)

std::cout << "-";

std::cout << "\n";

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

std::cout << (a+i)->name;

makeSpaces(19 - (a+i)->name.length());

std::cout << " |" << (a+i)->size;

makeSpaces(25 - numLength((a+i)->size));

std::cout << "|" << (a+i)->page\_;

makeSpaces(25 - numLength((a+i)->page\_));

std::cout << "|" << (a+i)->padding\_;

makeSpaces(24 - numLength((a+i)->padding\_));

std::cout << "|\n";

}

for(int i = 0;i < 98;i++)

std::cout << "-";

std::cout << "\nÂâåäèòå ÷òî-íèáóäü, ÷òîáû ïðîäîëæèòü\n";

std::string go;

std::cin >> go;

}

void remove(data \*a, int \*n, int page\_to\_kill, int padding\_to\_kill, int \*available\_memory){

bool i\_find\_it = false;

for(int i = 0;i < \*n;i++){

if((a+i)->page\_ == page\_to\_kill && (a+i)->padding\_ == padding\_to\_kill){

i\_find\_it = true;

\*(available\_memory) += (a+i)->size;

}

if(i\_find\_it)

\*(a+i) = \*(a+i+1);

}

\*n = \*n - 1;

}

void addPlace(data \*a, int \*n, int \*j, int \*prikol, int \*available\_memory, int \*attempts\_1){

int size;

std::string name;

std::cout << "Ââåäèòå èìÿ ïðîöåññà\n";

std::cin >> name;

std::cout << "Ââåäèòå ðàçìåð íóæíîé ïàìÿòè\n";

std::cin >> size;

if(size > \*available\_memory){

std::cout << "Â ñèñòåìå íå õâàòàåò ìåñòà, ÷òîáû âûäåëèòü ïàìÿòü\nÂâåäèòå ÷òî-íèáóäü, ÷òîáû ïðîäîëæèòü\n";

std::cin >> name;

return;

}

\*attempts\_1 = \*attempts\_1 + 1;

\*(available\_memory) -= size;

(a+\*prikol)->size = size;

(a+\*prikol)->name = name;

if(\*n > 10){

\*j = \*j + 1;

\*n = 0;

}

(a+\*prikol)->page\_ = \*j;

(a+\*prikol)->padding\_ = \*n;

\*n = \*n + 1;

\*prikol = \*prikol + 1;

}

int main(){

setlocale(LC\_ALL, "Russian");

bool dialog = true;

int your\_choose;

data memory[100];

int n = 0;

int j = 0;

int prikol = 0; // Êîë-âî ýëåìåíòîâ

int available\_memory;

int attempts = 0;

int attempts\_1 = 0;

std::cout << "Ââåäèòå êîë-âî äîñòóïíîé ïàìÿòè(ÊÁ)\n";

std::cin >> available\_memory;

while(dialog){

system("cls");

giveInfo(available\_memory, attempts, attempts\_1);

std::cout << "×òî âû õîòèòå ñäåëàòü?\n";

std::cin >> your\_choose;

switch(your\_choose){

case 1:

addPlace(&memory[0], &n, &j, &prikol, &available\_memory, &attempts\_1);

attempts++;

break;

case 2:

int page\_to\_kill;

int padding\_to\_kill;

std::cout << "Ââåäèòå íîìåð ñòðàíèöû è îòñòóï\n";

std::cin >> page\_to\_kill >> padding\_to\_kill;

remove(&memory[0], &prikol, page\_to\_kill, padding\_to\_kill, &available\_memory);

break;

case 3:

showTable(&memory[0], prikol);

break;

case 4:

std::cout << "Ââåäèòå íîìåð ñòðàíèöû è îòñòóï\n";

int page;

int padding;

std::cin >> page >> padding;

showAddress(page, padding, &memory[0], prikol);

break;

case 5:

dialog = false;

break;

}

}

std::cout << "\nÕîðîøåãî äíÿ!";

return 0;

}