**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. ИТП-11

Иванов Д.А.

Принял: преподаватель

Карась О.В.

Гомель 2022

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

**Задача:**

Разработать программу, реализующую заданный алгоритм выделения памяти. Менеджер памяти должен:

1. По запросу процесса выделять память, согласно заданному алгоритма. На экран должна выводится следующая информация о состоянии памяти: объём 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 на кнопку “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. Организовать циклическое выделение и освобождение памяти. При этом случайным образом задаётся количество выделяемых блоков и их размер.

Вариант 12:

Схема с переменными разделами. Случайный выбор раздела.

**Задание**

На первом рисунке показан приме выделения памяти

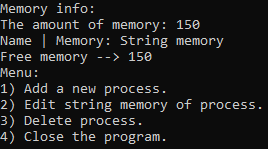


Рисунок 1 – пример выделения памяти

После того как мы выделили память у нас появляется меню, в котором мы можем добавить новый процесс, добавить строковую память процесса, удалить процесс или завершить работу программы. Если мы выберем первый пункт меню, то если захотим создать новый процесс у нас появится новое меню, где мы можем написать имя процесса и объем памяти, который он будет занимать. На рисунке 2 изображено меню, в котором происходит добавление процесса.



Рисунок 2 – пример добавления нового процесса

После того как мы добавим ещё несколько процессов, а именно еще 4 раза у нас в главном меню будет добавлено 5 процессов. Пример указан на рисунке 3.

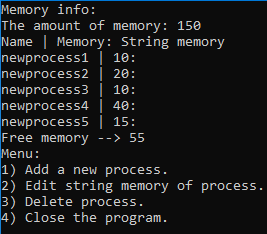


Рисунок 3 – пример добавления 5 новых процессов

После удаления 2 процессов, а именно 2 и 4 на месте остались блоки со свободной памятью (рис.4). При добавлении новых процессов они будут помещаться рандомно в один из блоков. Например добавив процесс *random* он поместился на место процесса 4, забрав нужное количество из свободной памяти (рис.5).

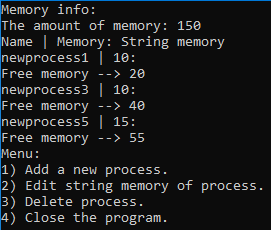


Рисунок 4 – пример удаления процессов 2 и 4

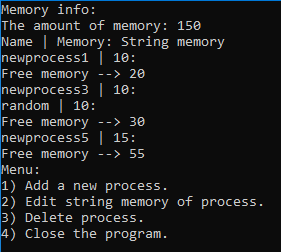


Рисунок 5 – пример добавления нового процесса после удаления

**Вывод**: были изучены различные способы работы с памятью. Была разработана программа, реализующая схему с переменными разделами, случайный выбор раздела.

**Листинг программы**

#include <bits/stdc++.h>

using namespace std;

vector<string> mem, name;

vector<int> proc;

vector<bool> full;

int m;

int mem\_rand(int am)

{

vector<int> temp\_num;

bool flag = false;

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

if (!full[i] && proc[i] >= am){

flag = true;

temp\_num.push\_back(i);

}

if (flag) return temp\_num[rand() % temp\_num.size()];

else return -1;

}

void mem\_insert(string str\_n, int amount)

{

int num = mem\_rand(amount);

if (num == -1) {

cout << "Not enough memory.";

system("pause");

return;

}

proc[num] -= amount;

proc.insert(proc.begin() + num, amount);

full.insert(full.begin() + num, true);

mem.insert(mem.begin() + num, "");

name.insert(name.begin() + num, str\_n);

}

int main()

{

cout << "Enter the amount of memory: ";

cin >> m;

name.reserve(m);

mem.reserve(m);

proc.reserve(m);

full.reserve(m);

proc.push\_back(m);

full.push\_back(false);

name.push\_back("");

mem.push\_back("");

int p = 0;

while (p != 4){

system("cls");

cout << "Memory info:" << endl << "The amount of memory: " << m << endl << "Name | Memory: String memory" << endl;

for (int i=0; i<name.size(); i++)

if (full[i]){

cout << name[i] << " | " << proc[i] << ": " << mem[i] << endl;

} else cout << "Free memory --> " << proc[i] << endl;

cout << "Menu:" << endl << "1) Add a new process." << endl << "2) Edit string memory of process." << endl << "3) Delete process." << endl << "4) Close the program." << endl;

cin >> p;

system("cls");

string temp\_n;

int temp\_m = 0;

switch(p){

case 1:

{

cout << "Enter the name of the new process: ";

cin.ignore();

getline(cin, temp\_n);

cout << "Enter the amount of memory needed for the process: ";

cin >> temp\_m;

mem\_insert(temp\_n, temp\_m);

break;

}

case 2:

{

cout << "Enter the name of the process: ";

cin.ignore();

getline(cin, temp\_n);

bool flag = false;

for (int i=0; i<name.size(); i++)

if (name[i] == temp\_n){

flag = true;

temp\_m = i;

break;

}

if (flag){

cout << "Enter a new string memory(the length of this string must be less than or equal to " << proc[temp\_m] << "): ";

getline(cin, temp\_n);

if (temp\_n.length() <= proc[temp\_m]) mem[temp\_m] = temp\_n;

else {

cout << "The length of this string is longer than " << proc[temp\_m] << "." << endl;

system("pause");

}

} else {

cout << "There are no processes with that name." << endl;

system("pause");

}

break;

}

case 3:

{

cout << "Enter the name of the process: ";

cin.ignore();

getline(cin, temp\_n);

bool flag = false;

for (int i=0; i<name.size(); i++)

if (name[i] == temp\_n){

flag = true;

temp\_m = i;

break;

}

if (flag){

full[temp\_m] = false;

for (int i=0; i<name.size() - 1; i++)

while ((i < name.size() - 1) && full[i] == false && full[i + 1] == false){

proc[i + 1] += proc[i];

proc.erase(proc.begin() + i);

full.erase(full.begin() + i);

mem.erase(mem.begin() + i);

name.erase(name.begin() + i);

}

} else {

cout << "There are no processes with that name." << endl;

system("pause");

}

break;

}

}

}

}