**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

Король В.Н.

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 3:

Свопинг. Выгружается процесс, занимающий наименьший объём памяти.

**Задание**

Свопинг — это перемещение процессов из главной памяти на диск и обратно. В моём варианте я должен выгружать процессы, которые занимают наименьший объём памяти. То есть если я, к примеру задаю 4 раздела памяти и каждый этот раздел будет занимать по 10кб (рисунок 1).

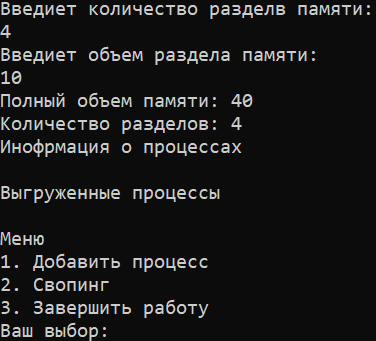


Рисунок 1 – пример выделения памяти

После того как мы выделили память у нас появляется меню, в котором мы можем добавить новый процесс или совершить свопинг процесса, который занимает наименьший объём памяти или завершить работу программы. Если мы выберем первый пункт меню, то есть захотим создать новый процесс у на появится новое меню, где мы можем казать имя процесса и объем памяти, который он будет занимать. На рисунке 2 изображенное меню, где происходит добавление процесса.

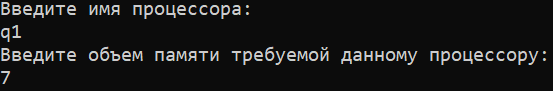


Рисунок 2 – пример добавления нового процесса

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

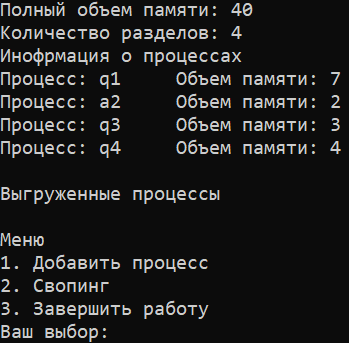


Рисунок 3 – пример добавления 4 новых процессов

После этого мы можем добавить еще один процесс и так как у на всего 4 блока памяти то одному из процессов придётся выгрузится. И так у нас свопинг происходит по принципу того, что выгружается процесс с наименьшим объёмом памяти то именно этот процесс и будет выгружен. Пример этого указан на рисунке 4.

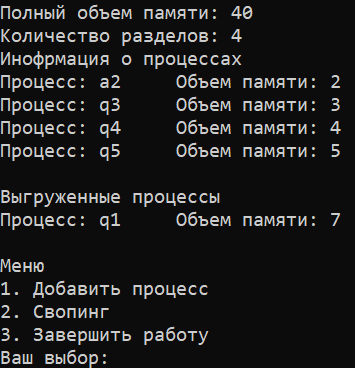


Рисунок 4 – пример выгрузки процесса с наименьшим объёмом памяти

Так же по нажатию на кнопку 2 мы можем сами выгружать процессы и выгружаться будут те процессы, у которых наименьший объём памяти. В нашем случае будет выгружен процесс под номер 4. Пример выполнения свопинга указан на рисунке 5.

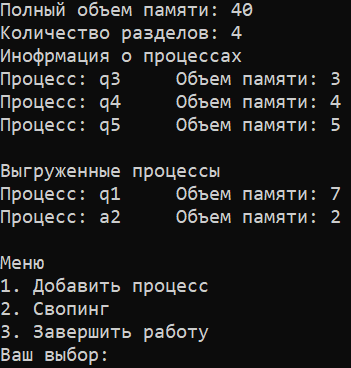


Рисунок 5 – свопинг процесса занимающий наименьший объём памяти

**Вывод**: были изучены различные способы работы с памятью. Была разработана программа, реализующая свопинг процессов, занимающих наименьший объём памяти.

**Листинг программы**

**#include<stdio.h>**

**#include<string.h>**

**#include<conio.h>**

**#include<ctype.h>**

**#include <windows.h>**

**#include<math.h>**

**struct proces{**

**int size;**

**char name[100];**

**int time;**

**};**

**int main()**

**{**

**SetConsoleCP (1251);**

**SetConsoleOutputCP (1251);**

**int v\_razdela;**

**int kol\_razd, k = 0, i, kol=0, time = 1, del\_proc, j=0, m, r = 1;**

**puts("Ââåäèåò êîëè÷åñòâî ðàçäåëâ ïàìÿòè: ");**

**scanf("%d", &kol\_razd);**

**puts("Ââåäèåò îáúåì ðàçäåëà ïàìÿòè: ");**

**scanf("%d", &v\_razdela);**

**struct proces proc\_kol[1000];**

**while(k != 3)**

**{**

**printf("Ïîëíûé îáúåì ïàìÿòè: %d\n", kol\_razd\*v\_razdela);**

**printf("Êîëè÷åñòâî ðàçäåëîâ: %d\n", kol\_razd);**

**puts("Èíîôðìàöèÿ î ïðîöåññàõ");**

**for(i = 0; i < kol\_razd; i++)**

**{**

**if(proc\_kol[i].size > 0 && kol <= kol\_razd)**

**printf("Ïðîöåññ: %s\tÎáúåì ïàìÿòè: %d\n", proc\_kol[i].name, proc\_kol[i].size);**

**if(kol > kol\_razd)**

**{**

**j = kol-kol\_razd;**

**m = j;**

**for(m; m < kol\_razd+r; m++)**

**{**

**if(proc\_kol[m].size != -1)**

**printf("Ïðîöåññ: %s\tÎáúåì ïàìÿòè: %d\n", proc\_kol[m].name, proc\_kol[m].size);**

**}**

**r += 1;**

**break;**

**}**

**}**

**printf("\n");**

**puts("Âûãðóæåííûå ïðîöåññû");**

**if(j != 0)**

**{**

**for(i = 0; i < j; i++)**

**{**

**if(proc\_kol[i].size > 0)**

**printf("Ïðîöåññ: %s\tÎáúåì ïàìÿòè: %d\n", proc\_kol[i].name, proc\_kol[i].size);**

**}**

**}**

**printf("\n");**

**puts("Ìåíþ");**

**puts("1. Äîáàâèòü ïðîöåññ");**

**puts("2. Ñâîïèíã");**

**puts("3. Çàâåðøèòü ðàáîòó");**

**puts("Âàø âûáîð: ");**

**scanf("%d", &k);**

**system("cls");**

**switch(k)**

**{**

**case 1:**

**puts("Ââåäèòå èìÿ ïðîöåññîðà: ");**

**scanf("%s", &proc\_kol[kol].name);**

**puts("Ââåäèòå îáúåì ïàìÿòè òðåáóåìîé äàííîìó ïðîöåññîðó: ");**

**scanf("%d", &proc\_kol[kol].size);**

**if(proc\_kol[kol].size > v\_razdela)**

**{**

**proc\_kol[kol].size = 0;**

**puts("Îáúåì ïàìÿòè óêàçàí íå âåðíî");**

**break;**

**}**

**else**

**{**

**proc\_kol[kol].time = kol + time;**

**kol += 1;**

**time += 1;**

**break;**

**}**

**case 2:**

**proc\_kol[kol].size = -1;**

**kol += 1;**

**break;**

**case 3:**

**break;**

**default:**

**puts("Ïóíêò ìåíþ âûáðàí íå âåðíî");**

**break;**

**}**

**}**

**fflush(stdin);**

**getchar();**

**return 0;**

**}**