**МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ**

**УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ**

**ГОМЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ**

**УНИВЕРСИТЕТ ИМЕНИ П. О. СУХОГО**

Факультет автоматизированных и информационных систем

Кафедра «Информационные технологии»

ОТЧЁТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №9

по дисциплине: «Операционные системы»

на тему: **«**Простейшие схемы управления памятью**»**

Выполнил: студент гр. ИТП-11

Лисицин Н.С.

Принял: преподаватель-стажёр

Карась О.В.

Гомель 2022

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

**Задача:**

Разработать программу, реализующую заданный алгоритм выделения памяти.



**Ход работы**

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

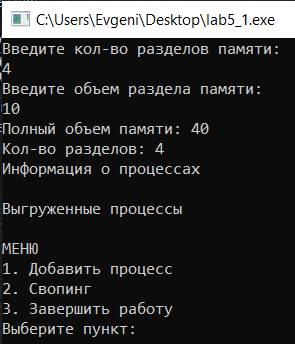


Рисунок 1 – Пример выделения памяти

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

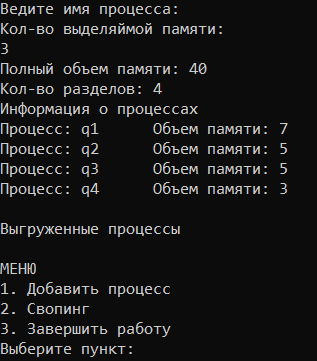


Рисунок 2 – Пример добавления нового процесса

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.

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 наибольшим объёмом памяти то именно этот процесс и будет выгружен. Пример этого указан на рисунке 3.

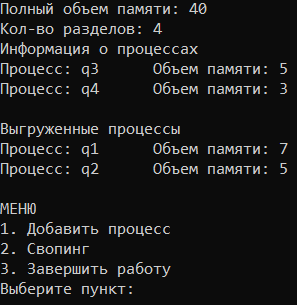
****

Рисунок 3 – Пример выгрузки процесса с наибольшим объёмом памяти

Так же по нажатию на кнопку 2 мы можем сами выгружать процессы и выгружаться будут те процессы, у которых наибольший объём памяти. В нашем случае будет выгружен процесс под номер 4.

**Вывод:** в результате выполнения лабораторной были изучены алгоритмы управления памятью и разработка программы менеджера памяти.

**ПРИЛОЖЕНИЕ А**

**Текст программы**

***lab*9.*cpp*:**

#*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*("*Ââåäèòå* *êîë*-*âî* *ðàçäåëîâ* *ïàìÿòè*: ");

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

*puts*("*Ââåäèòå* *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*("*Êîë*-*âî* *ðàçäåëîâ*: %*d*\*n*", *kol*\_*razd*);

*puts*("*Èíôîðìàöèÿ* *î* *ïðîöåññàõ*");

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

{

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

*printf*("*Ïðîöåññ*: %*s*\*t* *Îáúåì* *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*\*t* *Îáúåì* *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*\*t* *Îáúåì* *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*("*Âûáåðèòå* *ïóíêò*: ");

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

*system*("*cls*");

*switch*(*k*)

{

*case* 1:

*puts*("*Âåäèòå* *èìÿ* *ïðîöåññà*: ");

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

*puts*("*Êîë*-*âî* *âûäåëÿéìîé* *ïàìÿòè*: ");

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

*break*;

}

}

*fflush*(*stdin*);

*getchar*();

*return* 0;

}