|  |  |  |
| --- | --- | --- |
| **ИВТиПТ**  **Кафедра**  **ПИиИ**  **Группа 3.6** | **Дисциплина: Операционные системы** | **Лабораторная работа №4** |
| **ФИО студента: Остапец Павел Игоревич** |

**Цели и задачи работы**

Основная задача данной лабораторной работы — работа с разделяемой памятью в ОС с целью предоставления возможности обмена информации между процессам. Также во время выполнения работы студент ознакомится командами и с системными вызовами, необходимым для управления разделяемой памятью. А именно:

* написать программу **lab4\_1.c** , создающую блок разделяемой памяти и записывающей, в нее информацию и программу **lab4\_2.c**, которая подключается к ранее созданному блоку и считывает записанную информацию, выводя ее на экран. Обе программы также должны выводить на экран ключ и идентификатор блока разделяемой памяти,
* \*написать программы подобные выше-указанным **lab4\_1\_ext.c** и **lab4\_2\_ext.c**, но вместо записи и считывания информации, в каждой из программ допишите полностью идентичный участок кода, который будет в цикле проверять и изменять содержимое переменной в разделяемой памяти
* написать отчет о выполненной работе с фиксацией результата,
* ответить на контрольные вопросы.

**Лабораторное оборудование**

Лабораторным сервером для выполнения заданий представляет собой удаленный сервер под управлением ОС Linux. Для подключения к лабораторному оборудованию необходимо использовать любое ПО для подключения к серверу по SSH.

**Краткая теория**

В Unix системах, для кооперативной работы процессов можно использовать инструменты Inter Process Communication System V (IPC). В этой лабораторной рассматривается использование разделяемой памяти (Shared Memory).

В отличие от таблицы дескрипторов открытых файлов, которую ОС ведет для каждого процесса отдельно, список ресурсов IPC ведется системой централизовано. Поэтому любой процесс может создать, а любой(ые) другой может подключить эту область в свое адресное пространство данных эту область. И все процессы смогут записывать и считывать данные из/в эту область, как будто в свою собственную.

Для идентификации, создания и подключения области памяти в адресное пространство процесса используются системные вызовы:

API следующее:

#include <sys/types.h>

#include <sys/ipc.h>

#include <sys/shm.h>

key\_t ftok(const char \*pathname, int proj\_id);

int shmget(key\_t key, size\_t size, int shmflg);

void \*shmat(int shmid, const void \*shmaddr, int shmflg);

int shmdt(const void \*shmaddr);

int shmctl(int shmid, int cmd, struct shmid\_ds \*buf);

Кроме системных вызовов, для управления объектами IPC можно использовать консольные команды:

* **ipcs** — показать все объекты IPC, в т.ч. разделяемую память
* **iprm -m id** — удалить область разделяемой памяти по id.

**Ход работы**

1. В первой программе первого задания требовалось последовательно выполнить три этапа подключения разделяемой памяти в адресное пространство процесса, после чего добавить код, который позволит записывать данные в разделяемую память (Рисунок 1)

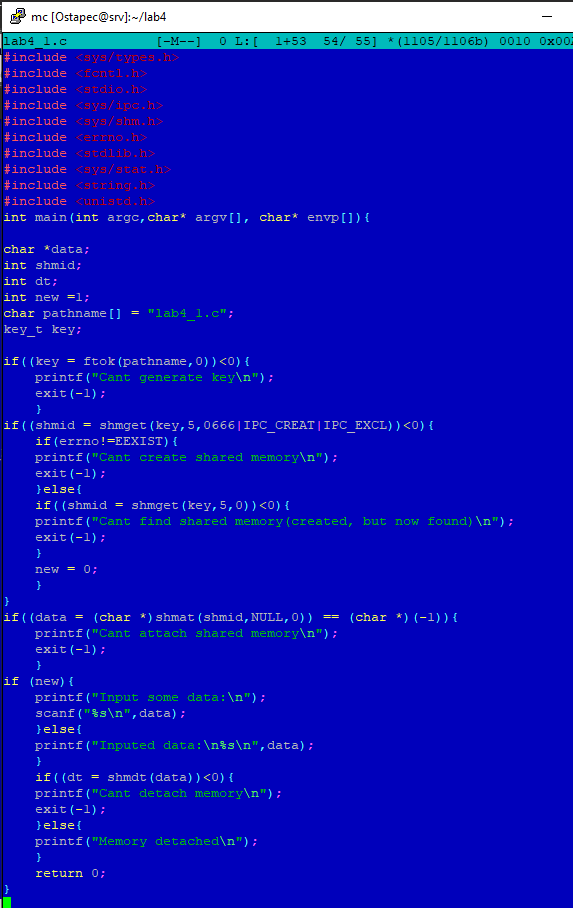


Рисунок 1

1. Во второй программе первого задания требовалось выполнить аналогичные действия, но нужно было добавить код, с помощью которого данные, записанные первой программой, считывались бы из разделяемой памяти и выводились на экран (Рисунок 2)

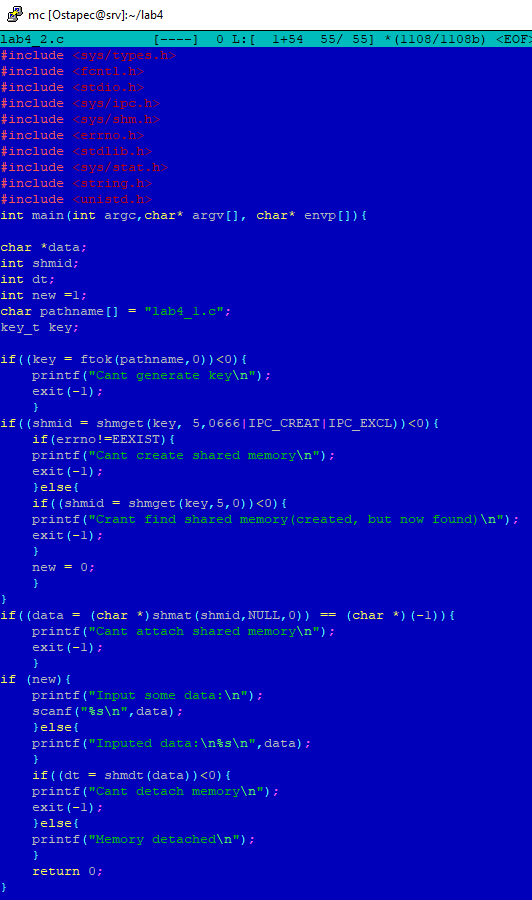


Рисунок 2

1. После написания программ, требовалось откомпилировать и запустить обе программы, зафиксировав выведенную информацию (Рисунок 3).

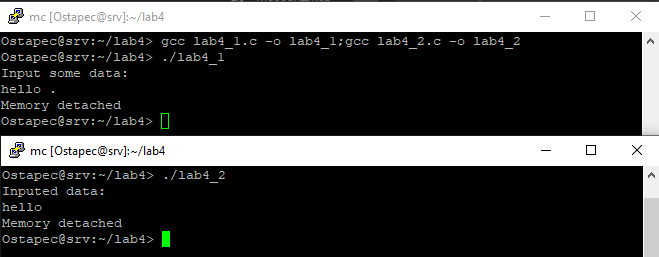


Рисунок 3

Также на рисунке представлено использование команды **ipcs,** с помощью которой показана созданная ячейка разделяемой памяти. (Рисунок 4)

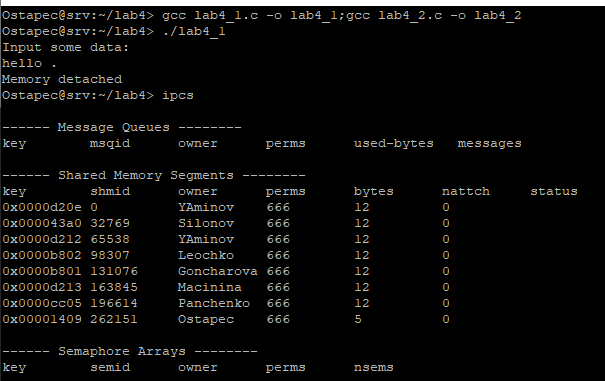


Рисунок 4

**Результаты работы.**

В результате работы было выполнено задание, в ходе которого была создана программа, которая создает ячейку разделяемой памяти, подключается к ней и успешно передает данные между процессами посредством разделяемой памяти.

**Литература**

1. В.Е. Карпов, К.А. Коньков «Основы операционных систем Практикум» http://www.intuit.ru/studies/courses/2249/52/info