|  |  |  |
| --- | --- | --- |
| **ИВТиПТ**  **Кафедра**  **ПИиИ**  **Группа 3.6** | **Дисциплина: Операционные системы** | **Лабораторная работа №3** |
| **ФИО студента: Остапец Павел Игоревич** |

**Цели и задачи работы**

Основная задача данной лабораторной работы — работа с процессами в ОС. Также во время выполнения работы студент ознакомится с системными вызовами, необходимым для управления процессами. А именно:

* написать программу **lab3\_1.c** на языке C, создающую файл на диске и записывающую информацию в этот файл,
* написать программу **lab3\_2.c** на языке C, для передачи информации внутри одного процесса через канал связи (pipe),
* написать программу **lab3\_3.c** на языке C, для организации однонаправленной связи между родственными процессами через канал связи (pipe),
* написать программу **lab3\_4.c\*** на языке С, для организации однонаправленной связи между родственными процессами через именованный канал связи (FIFO),,
* 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**

Для работы с каналами связи между процессами (pipe) используются те же системные вызовы, что и для работы с файлами. С точки зрения программирования на языке C в операционной системе Linux используются функции API, такие как **open**, **read**, **write**, **close**. Слушатели курса должны были уже ранее сталкиваться с использованием этого функционала. Однако теперь становится понятно, что на самом деле эти функции всего лишь обертки, в которых делается системный вызов к операционной системе. И именно ОС делает всю работу по управлению файлами и и обменом информации. При этом в **PCB** процесса инициирующего открытие файла , в таблицу открытых файлов добавляется строка с новым индексом, в которой указывается к какому файлу или каналу связи произошло подключение. Индекс же называется дескриптором файла. Кроме того в этой работе используются два новых системных вызова:

**pipe** — этот вызов создает канал неименованный канал связи. На вход ему подается массив, размером в два целочисленных значения. Если выполнение успешно, то в указанном массиве в ячейке с индексом 0 будет находится файловый дескриптор предназначенный для чтения, а в ячейке с индексом 1 будет находиться файловый дескриптор предназначенный на запись. Таким образом процесс (процессы) может записывать в декскриптор1 и считывать переданную информацию из дескриптора0. Поскольку канал связи неименованный, то работать с ним можно, только через эти два дескриптора. Т.о. работать с открытым каналом может только сам процесс, либо его подпроцессы, т. к. они получат по наследованию таблицу дескрипторов открытых файлов от родительского процесса.

**mknod** — этот вызов создаст именованный канал связи, который в Linux называется fifo - очередь связи с дисциплиной First In First Out . При выполнении этого вызова на файловой системе будет создан специальный псевдофайл типа fifo. Именно его и должны открывать процессы для чтения или записи при использования канала связи (pipe). Важно отметить, что fifo является именно каналом межпроцессного взаимодействия при помощи ОС, но не файлом! Т.е. например не получится воспользоваться этим псевдофайлом для обмена данными между процессам на разных компьютерах.

API следующее:

#include <sys/types.h>

#include <fcntl.h>

#include <stdio.h>

#include <stdlib.h>

#include <sys/stat.h>

#include <unistd.h>

int pipe(int fildes[2]);

int mknod(const char \*path, mode\_t mode, dev\_t dev);

int open(const char \*path, int oflag, ...);

int close(int fildes);

ssize\_t write(int fildes, const void \*buf, size\_t nbyte);

ssize\_t read(int fildes, void \*buf, size\_t nbyte);

**Ход работы**

1. В первом задании используется несколько системных вызовов:

Среди них **open()**, **write()** и **close().** В программе создается файл **testfile.txt** посредством open(), после чего в него записывается строка. Далее строка считывается с помощью системного вызова **read().** Вывод на содержимого файла на экран производится с помощью команды **cat <имя файла>** (Рисунок 1,2)

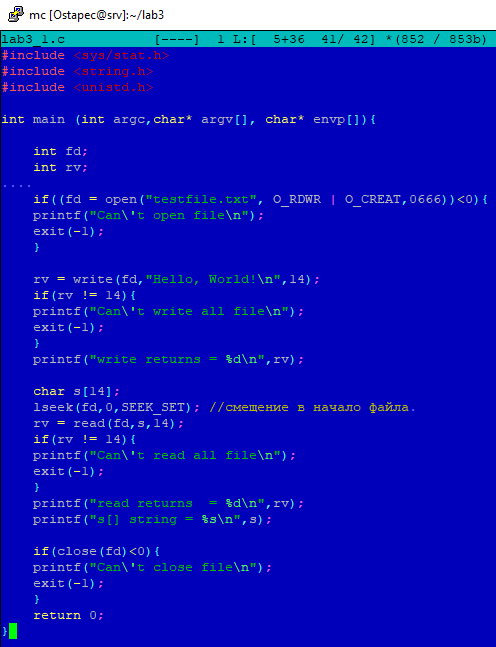


Рисунок 1.

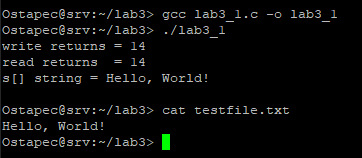


Рисунок 2.

1. Во втором задании обмен информации происходит по каналу связи **pipe()**. С помощью двух дескрипторов файлов вызывается сначала системный вызов **write()**, записывающий информацию в канал, а после системный вызов **read()** считывает ее из канала связи. (Рисунок 3, 4).

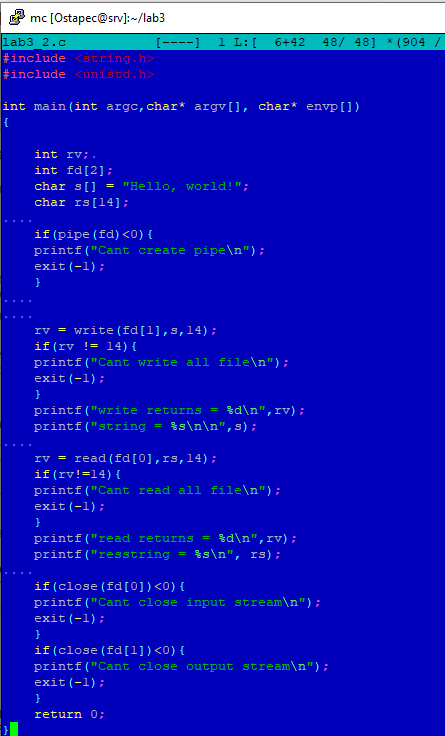


Рисунок 3

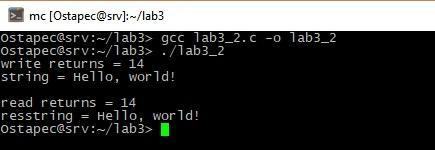


Рисунок 4.

1. В третьем задании производятся аналогичные второму заданию действия, но теперь после создания канала связи создается так же подпроцесс с помощью (**fork())**. По ходу программы в дочернем процессе происходит запись строки в канал (**write()**),а в родительском эта строка считывается (**read()**) (Рисунок 5, 6)

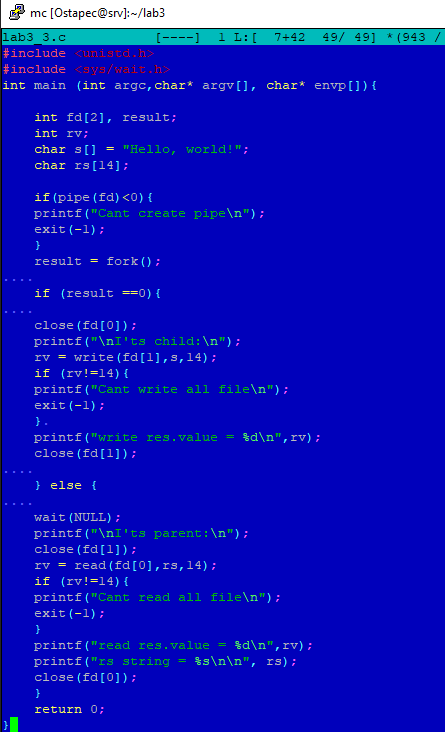


Рисунок 5.

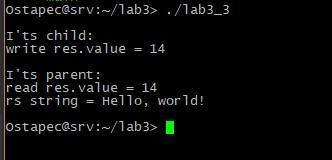


Рисунок 6.

**Результаты работы.**

В результате работы было выполнено 3 задания, в которых затрагивалась работа с файлами и пайпами, были изучены способы обмена информацией между процессами в ОС и системные вызовы, необходимые для работы с файлами.

**Литература**

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