

РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

Факультет физико-математических и естественных наук

Кафедра прикладной информатики и теории вероятностей

ОТЧЕТ

ПО ЛАБОРАТОРНОЙ РАБОТЕ № 13

дисциплина: Операционные системы

Студент: Бешкуров Михаил

Группа: НК-101

МОСКВА

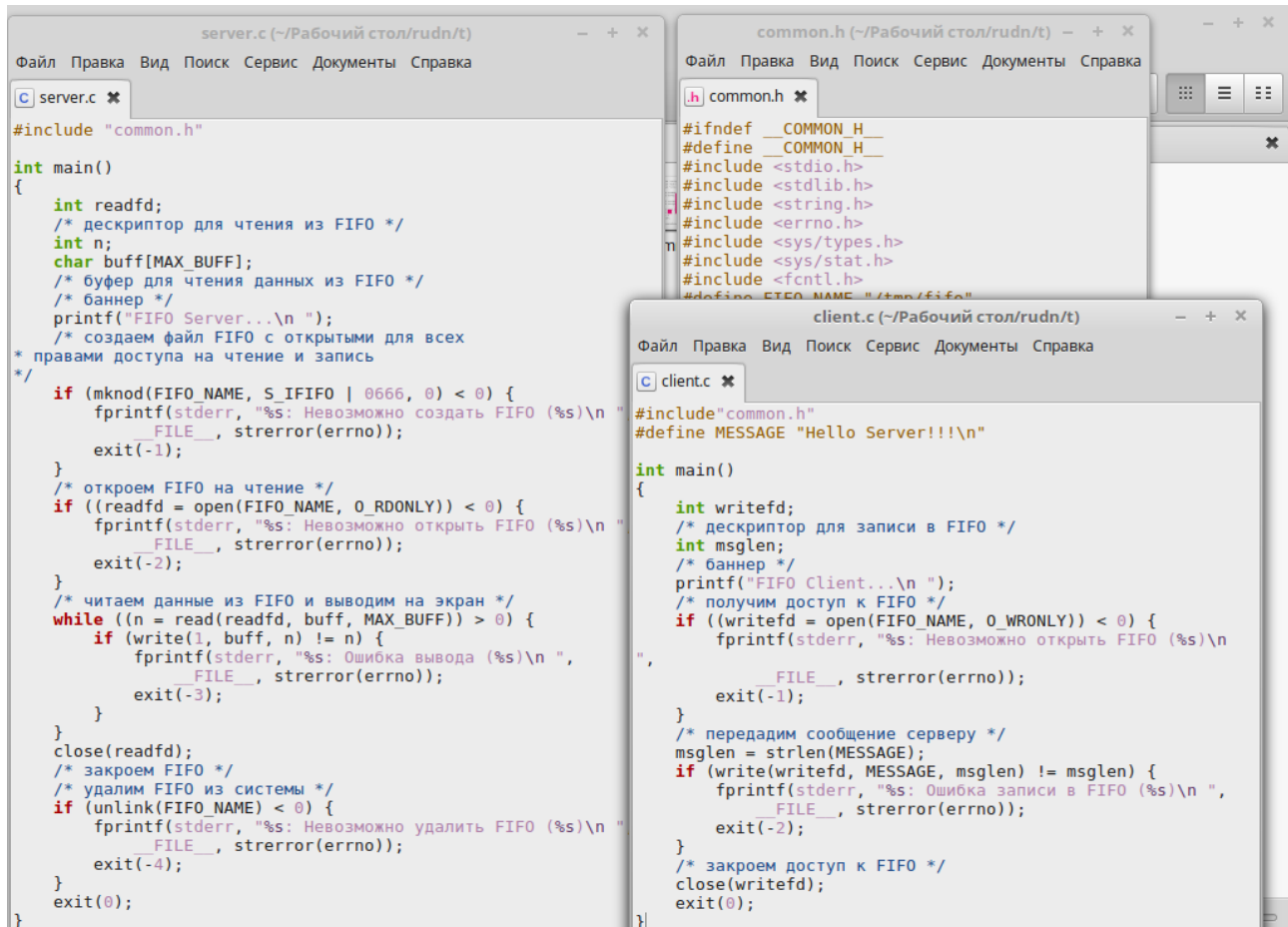
2019 г.

1. Цель работы

Приобретение практических навыков работы с именованными каналами.

2. Описание процесса выполнения задания

0. Изучил приведённые в тексте программы server.c и client.c



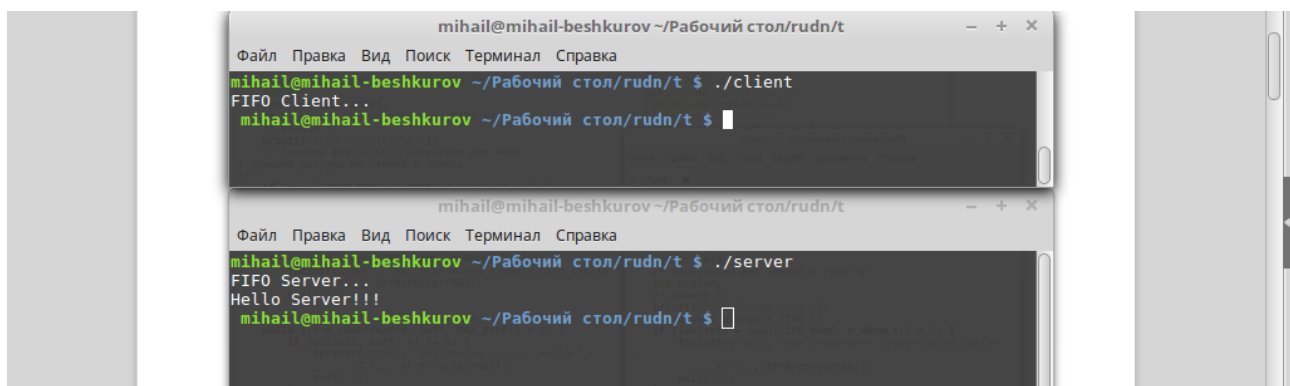
```
server.c (~/Рабочий стол/rudn/t)
#include "common.h"

int main()
{
    int readfd;
    /* дескриптор для чтения из FIFO */
    int n;
    char buff[MAX_BUFF];
    /* буфер для чтения данных из FIFO */
    /* баннер */
    printf("FIFO Server...\n");
    /* создаем файл FIFO с открытыми для всех
    * правами доступа на чтение и запись
    */
    if (mkfifo(FIFO_NAME, S_IFIFO | 0666, 0) < 0) {
        fprintf(stderr, "%s: Невозможно создать FIFO (%s)\n",
            __FILE__, strerror(errno));
        exit(-1);
    }
    /* откроем FIFO на чтение */
    if ((readfd = open(FIFO_NAME, O_RDONLY)) < 0) {
        fprintf(stderr, "%s: Невозможно открыть FIFO (%s)\n",
            __FILE__, strerror(errno));
        exit(-2);
    }
    /* читаем данные из FIFO и выводим на экран */
    while ((n = read(readfd, buff, MAX_BUFF)) > 0) {
        if (write(1, buff, n) != n) {
            fprintf(stderr, "%s: Ошибка вывода (%s)\n",
                __FILE__, strerror(errno));
            exit(-3);
        }
    }
    close(readfd);
    /* закроем FIFO */
    /* удалим FIFO из системы */
    if (unlink(FIFO_NAME) < 0) {
        fprintf(stderr, "%s: Невозможно удалить FIFO (%s)\n",
            __FILE__, strerror(errno));
        exit(-4);
    }
    exit(0);
}

common.h (~/Рабочий стол/rudn/t)
#ifndef __COMMON_H__
#define __COMMON_H__
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <errno.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#define FIFO_NAME "/tmp/fifo"

client.c (~/Рабочий стол/rudn/t)
#include "common.h"
#define MESSAGE "Hello Server!!!\n"

int main()
{
    int writefd;
    /* дескриптор для записи в FIFO */
    int msglen;
    /* баннер */
    printf("FIFO Client...\n");
    /* получим доступ к FIFO */
    if ((writefd = open(FIFO_NAME, O_WRONLY)) < 0) {
        fprintf(stderr, "%s: Невозможно открыть FIFO (%s)\n",
            __FILE__, strerror(errno));
        exit(-1);
    }
    /* передадим сообщение серверу */
    msglen = strlen(MESSAGE);
    if (write(writefd, MESSAGE, msglen) != msglen) {
        fprintf(stderr, "%s: Ошибка записи в FIFO (%s)\n",
            __FILE__, strerror(errno));
        exit(-2);
    }
    /* закроем доступ к FIFO */
    close(writefd);
    exit(0);
}
```

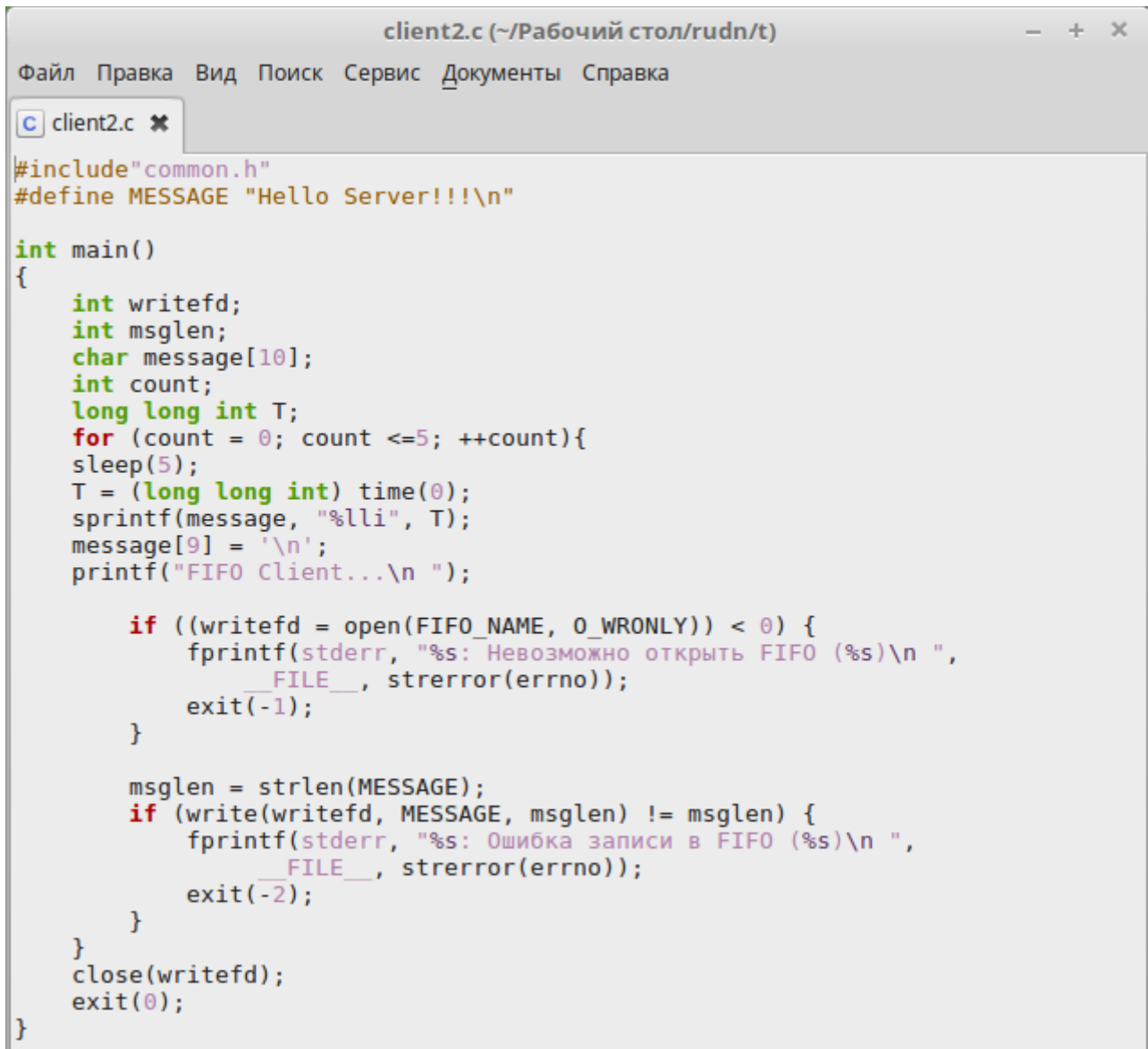


```
mihail@mihail-beshkurov ~/Рабочий стол/rudn/t
mihail@mihail-beshkurov ~/Рабочий стол/rudn/t $ ./client
FIFO Client...
mihail@mihail-beshkurov ~/Рабочий стол/rudn/t $

mihail@mihail-beshkurov ~/Рабочий стол/rudn/t
mihail@mihail-beshkurov ~/Рабочий стол/rudn/t $ ./server
FIFO Server...
Hello Server!!!
mihail@mihail-beshkurov ~/Рабочий стол/rudn/t $
```

2. Мы написали аналогичные программы и внесли изменения:

Работает 2 клиента.(client2.c уже написан с изменениями, представленными ниже, а client.c написан без изменений и передаёт сообщение серверу).



```
client2.c (~/Рабочий стол/rudn/t)
Файл Правка Вид Поиск Сервис Документы Справка

client2.c ✖
#include "common.h"
#define MESSAGE "Hello Server!!!\n"

int main()
{
    int writefd;
    int msglen;
    char message[10];
    int count;
    long long int T;
    for (count = 0; count <=5; ++count){
        sleep(5);
        T = (long long int) time(0);
        sprintf(message, "%lli", T);
        message[9] = '\n';
        printf("FIFO Client...\n ");

        if ((writefd = open(FIFO_NAME, O_WRONLY)) < 0) {
            fprintf(stderr, "%s: Невозможно открыть FIFO (%s)\n ",
                __FILE__, strerror(errno));
            exit(-1);
        }

        msglen = strlen(MESSAGE);
        if (write(writefd, MESSAGE, msglen) != msglen) {
            fprintf(stderr, "%s: Ошибка записи в FIFO (%s)\n ",
                __FILE__, strerror(errno));
            exit(-2);
        }
    }
    close(writefd);
    exit(0);
}
```

Мы переписали программу, где клиент работает с некоторой периодичностью, с использованием функции sleep(). А сервер прекращает работу через некоторое время. Клиент передаёт серверу время работы.

```
mi hail@mi hail-beshkurov ~/Рабочий стол/rudn/t $ ./client
FIFO Client...
client.c: Невозможно открыть FIFO (No such file or directory)
mi hail@mi hail-beshkurov ~/Рабочий стол/rudn/t $ ./client
FIFO Client...
client.c: Невозможно открыть FIFO (No such file or directory)
mi hail@mi hail-beshkurov ~/Рабочий стол/rudn/t $ ./server
FIFO Server...
Hello Server!!!
Hello Server!!!
Hello Server!!!
Hello Server!!!
Hello Server!!!
Hello Server!!!
mi hail@mi hail-beshkurov ~/Рабочий стол/rudn/t $ ./server
FIFO Server...
Hello Server!!!
mi hail@mi hail-beshkurov ~/Рабочий стол/rudn/t $

mi hail@mi hail-beshkurov ~/Рабочий стол/rudn/t $ ./client
client client2
mi hail@mi hail-beshkurov ~/Рабочий стол/rudn/t $ ./client2
FIFO Client...
FIFO Client...
FIFO Client...
FIFO Client...
FIFO Client...
mi hail@mi hail-beshkurov ~/Рабочий стол/rudn/t $

mi hail@mi hail-beshkurov ~/Рабочий стол/rudn/t $ ./client
FIFO Client...
mi hail@mi hail-beshkurov ~/Рабочий стол/rudn/t $ ./client
FIFO Client...
mi hail@mi hail-beshkurov ~/Рабочий стол/rudn/t $
```

В случае, если сервер завершит работу, не закрыв канал, файл FIFO не удалится. Поэтому в следующий раз создать этот файл будет нельзя и возникнет ошибка.

3. Вывод

Приобрел практические навыки работы с именованными каналами.

4. Контрольные вопросы

1. Именованные каналы отличаются от неименованных наличием идентификатора канала, который представлен как специальный файл (соответственно имя именованного канала — это имя файла).
2. Для создания неименованного канала используется системный вызов `pipe`. Массив из двух целых чисел является выходным параметром этого системного вызова.
3. Вы можете создавать именованные каналы из командной строки и внутри программы. С давних времен программой создания их в командной строке была команда: `mknod - $ mknod имя_файла`, однако команды `mknod` нет в списке команд `X/Open`, поэтому она включена не во все UNIX-подобные системы. Предпочтительнее применять в командной строке - `$ mkfifo имя_файла`.
4. Описание функции Си, создающей неименованный канал:
`int read(int pipe_fd, void *area, int cnt);`

```
int write(int pipe_fd, void *area, int cnt);
```

Первый аргумент этих вызовов - дескриптор канала, второй - указатель на область памяти, с которой происходит обмен, третий - количество байт. Оба вызова возвращают число переданных байт (или -1 - при ошибке).

5. Описание функции Си, создающей именованный канал:

```
int mkfifo (const char *pathname, mode_t mode);
```

Первый параметр — имя файла, идентифицирующего канал, второй параметр маска прав доступа к файлу. Вызов функции `mkfifo()` создаёт файл канала (с именем, заданным макросом `FIFO_NAME`): `mkfifo(FIFO_NAME, 0600);`

6. При чтении меньшего числа байтов, чем находится в канале, возвращается требуемое число байтов, остаток сохраняется для последующих чтений. При чтении большего числа байтов, чем находится в канале или FIFO возвращается доступное число байтов.

7. При записи большего числа байтов, чем это позволяет канал или FIFO, вызов `write(2)` блокируется до освобождения требуемого места. При этом атомарность операции не гарантируется. Если процесс пытается записать данные в канал, не открытый ни одним процессом на чтение, процессу генерируется сигнал. Запись числа байтов, меньшего емкости канала или FIFO, гарантированно атомарно. Это означает, что в случае, когда несколько

процессов одновременно записывают в канал, порции данных от этих процессов не перемешиваются.

8. В общем случае возможна многонаправленная работа процессов с каналом, т.е. возможна ситуация, когда с одним и тем же каналом взаимодействуют два и более процесса, и каждый из взаимодействующих каналов пишет и читает информацию в канал. Но традиционной схемой организации работы с каналом является однонаправленная организация, когда канал связывает два, в большинстве случаев, или несколько взаимодействующих процесса, каждый из которых может либо читать, либо писать в канал.

9. `write` — функция, записывающая `length` байтов из буфера `buffer` в файл, определенный дескриптором файла `fd`. Эта операция чисто 'двоичная' и без буферизации. Реализуется как непосредственный вызов DOS. С помощью функции `write` мы посылаем сообщение клиенту или серверу.

10. Строковая функция `strerror` - функция языков C/C++, транслирующая код ошибки, который обычно хранится в глобальной переменной `errno`, в сообщение об ошибке, понятном человеку. Ошибки эти возникают при вызове функций стандартных Си-библиотек. Возвращенный указатель ссылается на статическую строку с ошибкой, которая не должна быть изменена программой. Дальнейшие вызовы функции `strerror` перезапишут содержание этой строки. Интерпретированные сообщения об ошибках могут различаться, это зависит от платформы и компилятора.