

# РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

Факультет физико-математических и естественных наук Кафедра прикладной информатики и теории вероятностей

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №14 =====

дисциплина: Операционные системы

Студент: Койфман Кирилл Дмитриевич

Группа: НПИбд-01-21

## Введение.

---

### Цель работы.

Приобретение практических навыков работы с именованными каналами.

### Задачи.

Изучите приведённые в тексте программы `server.c` и `client.c`. Взяв данные примеры за образец, напишите аналогичные программы, внося следующие изменения:

1. Работает не 1 клиент, а несколько (например, два).
2. Клиенты передают текущее время с некоторой периодичностью (например, раз в пять секунд). Используйте функцию `sleep()` для приостановки работы клиента.
3. Сервер работает не бесконечно, а прекращает работу через некоторое время (например, 30 сек). Используйте функцию `clock()` для определения времени работы сервера. Что будет в случае, если сервер завершит работу, не закрыв канал?

### Теоретическое введение.

Одним из видов взаимодействия между процессами в операционных системах является обмен сообщениями. Под сообщением понимается последовательность байтов, передаваемая от одного процесса другому.

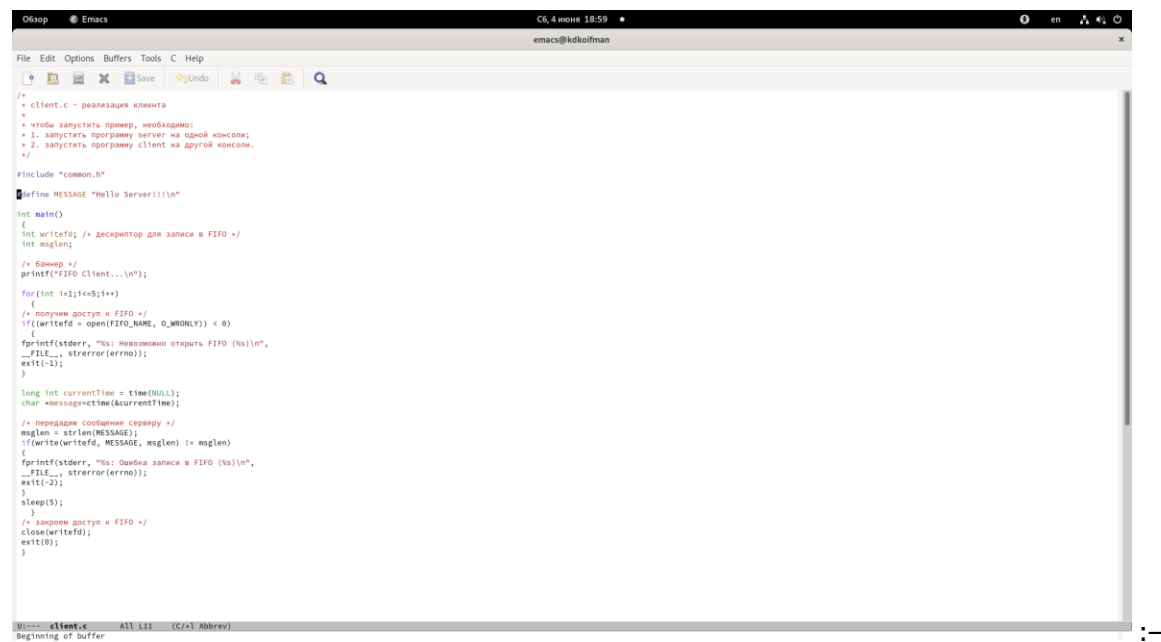
В операционных системах типа UNIX есть 3 вида межпроцессорных взаимодействий: общепонимание (именованные каналы, сигналы), System V Interface Definition (SVID — разделяемая память, очередь сообщений, семафоры) и BSD (сокеты).

Для передачи данных между неродственными процессами можно использовать механизм именованных каналов (named pipes). Данные передаются по принципу FIFO (First In First Out) (первым записан — первым прочитан), поэтому они называются также FIFO pipes или просто FIFO. Именованные каналы отличаются от неименованных наличием идентификатора канала, который представлен как специальный файл (соответственно имя именованного канала — это имя файла). Поскольку файл находится на локальной файловой системе, данное IPC используется внутри одной системы.

## Ход работы.

Создадим файлы client.c, server.c, common.h и Makefile(рис.1, 2, 3, 4)

В файл client.c мы добавили цикл for, который будет регулировать количество принимаемых сообщений, а также команду sleep() с аргументом 5(т.е. программа будет приостанавливать свою работы на 5 секунд).



```
/*
 * client.c - реализация клиента
 *
 * чтобы запустить пример, необходимо:
 * 1. запустить программу server на одной консоли;
 * 2. запустить программу client на другой консоли.
 */

#include "common.h"

#define MESSAGE "Hello Server!!\n"

int main()
{
    int writefd; /* дескриптор для записи в FIFO */
    int msglen;

    /* Бампер */
    printf("FIFO Client...\n");

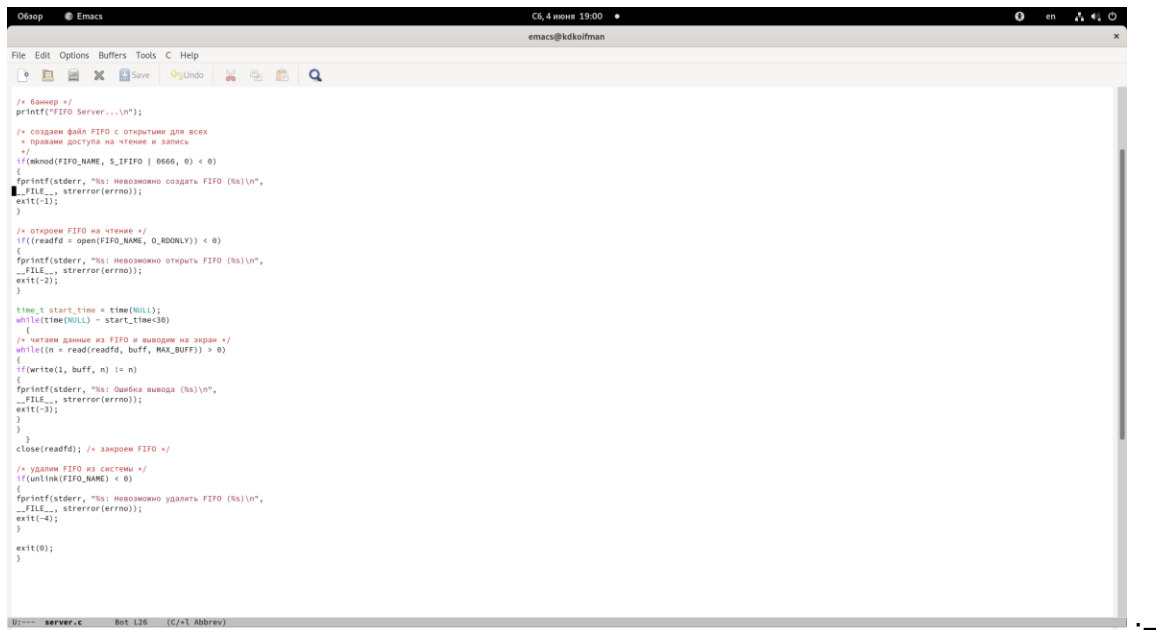
    for(int i=1;i<=5;i++)
    {
        /* получим доступ к FIFO */
        if((writefd = open(FIFO_NAME, O_WRONLY)) < 0)
        {
            fprintf(stderr, "hs: невозможно открыть FIFO (%s)\n",
                __FILE__, strerror(errno));
            exit(-1);
        }

        long int currentTime = time(NULL);
        char *message = ctime(&currentTime);

        /* передаем сообщение серверу */
        msglen = strlen(MESSAGE);
        if(write(writefd, MESSAGE, msglen) != msglen)
        {
            fprintf(stderr, "hs: ошибка записи в FIFO (%s)\n",
                __FILE__, strerror(errno));
            exit(-1);
        }
        sleep(5);
    }
    /* закрываем доступ к FIFO */
    close(writefd);
    exit(0);
}
```

рис.1(client.c)

В файл server.c мы добавили цикл while, предназначенный здесь для регулирования времени работы нашего сервера(сервер работает не бесконечно, а прекращает работу через некоторое время (30 сек))



```
/* Баннер */
printf("FIFO Server...\n");

/* создаем файл FIFO с открытыми для всех
 * правами доступа на чтение и запись
 */
if(mknod(FIFO_NAME, S_IFIFO | 0666, 0) < 0)
{
    fprintf(stderr, "%s: невозможно создать FIFO (%s)\n",
            __FILE__, strerror(errno));
    exit(-1);
}

/* открываем FIFO на чтение */
if((readfd = open(FIFO_NAME, O_RDONLY)) < 0)
{
    fprintf(stderr, "%s: невозможно открыть FIFO (%s)\n",
            __FILE__, strerror(errno));
    exit(-2);
}

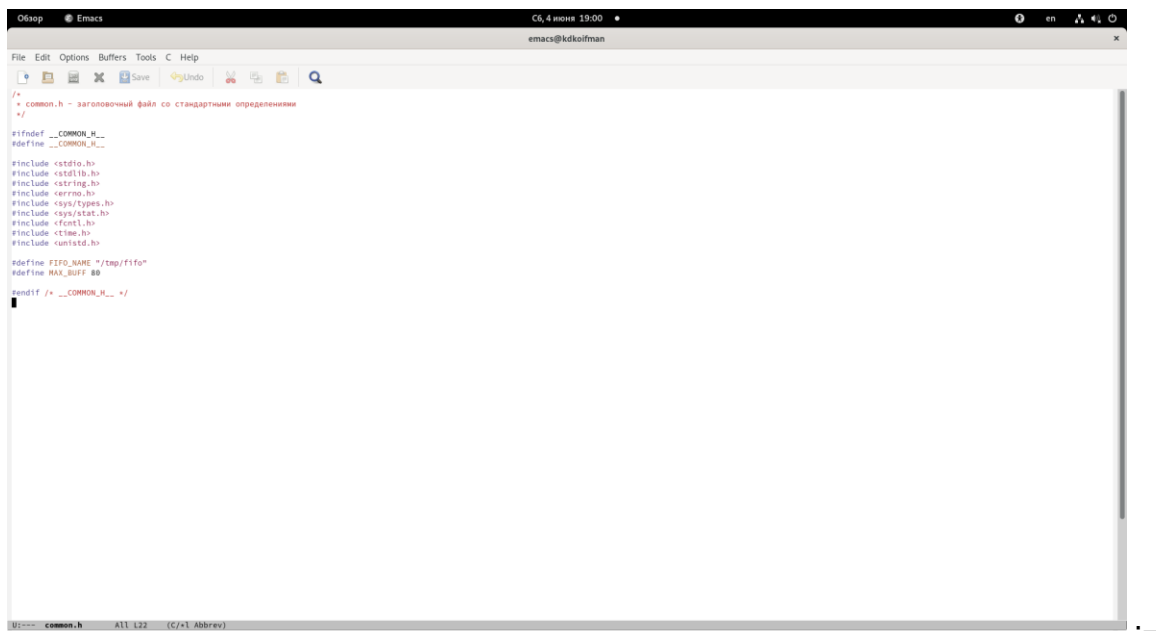
time_t start_time = time(NULL);
while(time(NULL) - start_time < 30)
{
    /* читаем данные из FIFO и выводим на экран */
    while((n = read(readfd, buff, MAX_BUFF)) > 0)
    {
        if(write(1, buff, n) != n)
        {
            fprintf(stderr, "%s: ошибка вывода (%s)\n",
                    __FILE__, strerror(errno));
            exit(-3);
        }
    }
    close(readfd); /* закрываем 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);
}
```

рис.2(server.c)

В файл common.h мы подключили необходимые для файлов server.c и client.c заголовочные файлы time.h и unistd.h.



```
/*
 * common.h - заголовочный файл со стандартными определениями
 */

#ifndef __COMMON_H__
#define __COMMON_H__

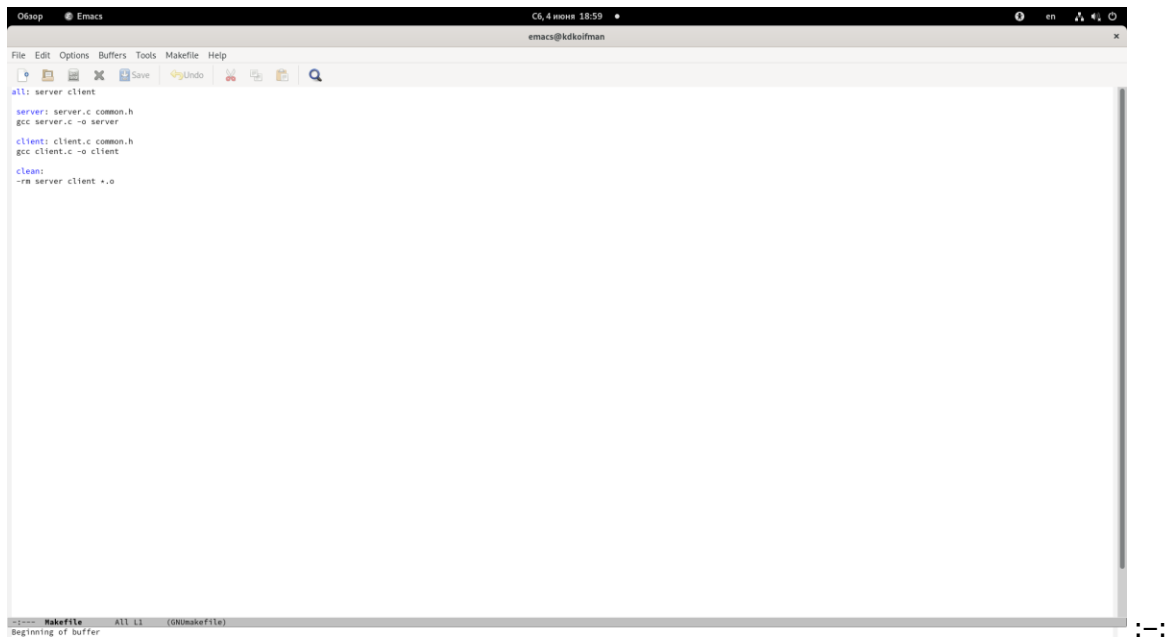
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <errno.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <time.h>
#include <unistd.h>

#define FIFO_NAME "/tmp/fifo"
#define MAX_BUFF 80

#endif /* __COMMON_H__ */

```

рис.3(common.h)



```
all: server client

server: server.c common.h
gcc server.c -o server

client: client.c common.h
gcc client.c -o client

clean:
rm server client *.o
```

Makefile All LL (GNUmakefile)

рис.4(Makefile)

Теперь проверим работу сервера(рис.5, 6, 7).

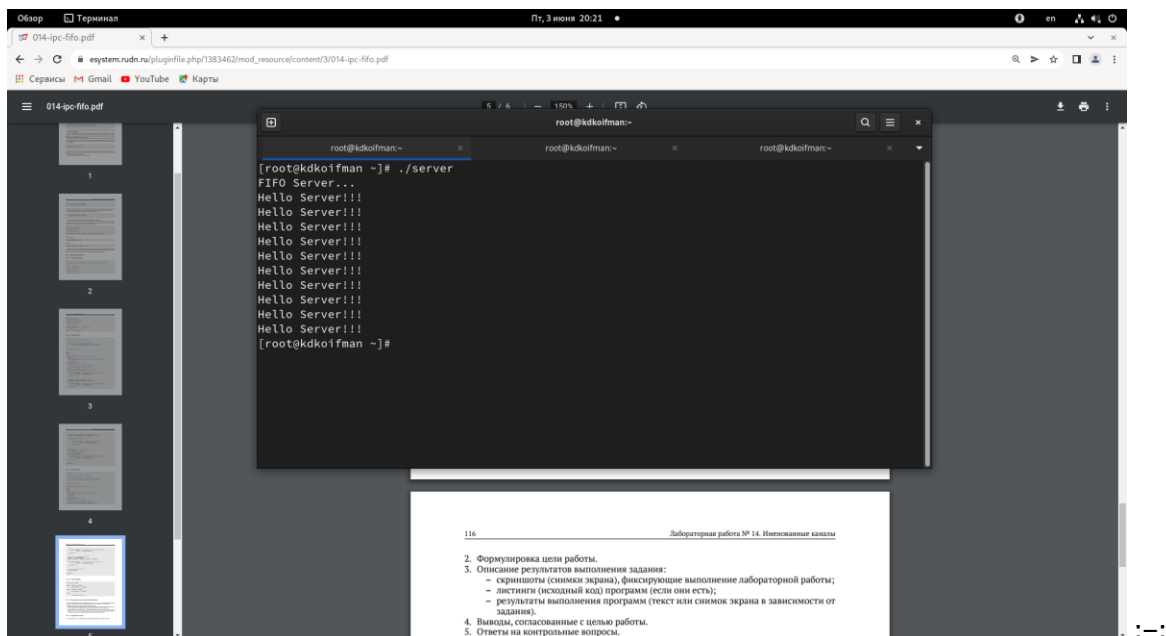


рис.5(запуск сервера(1-ый клиент))

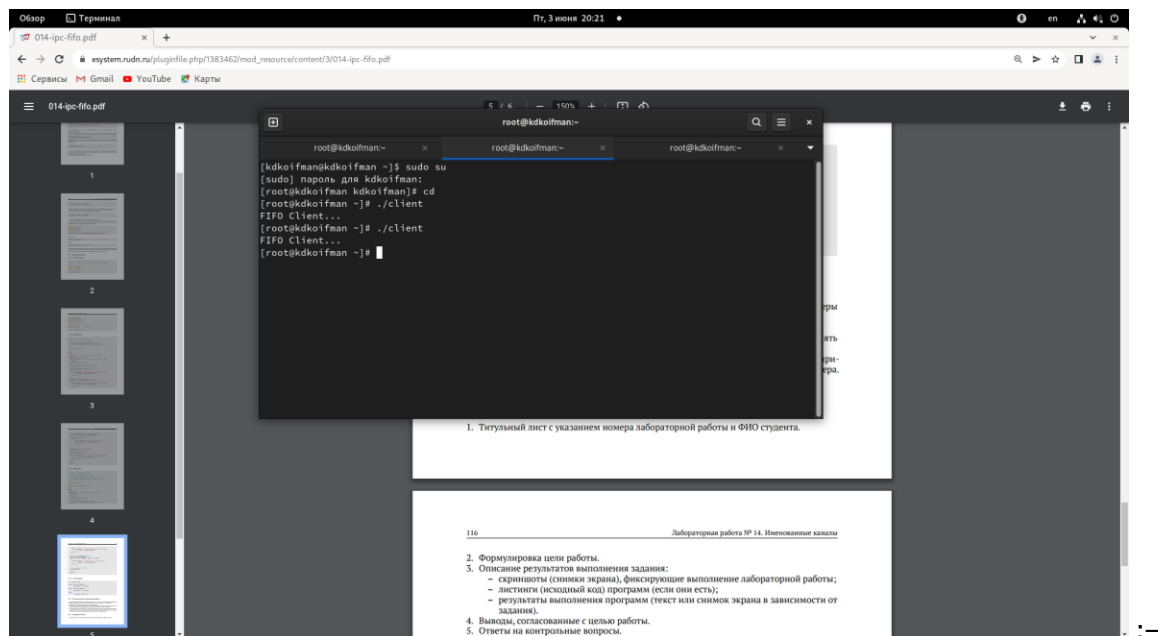


рис.6(2-ой клиент)

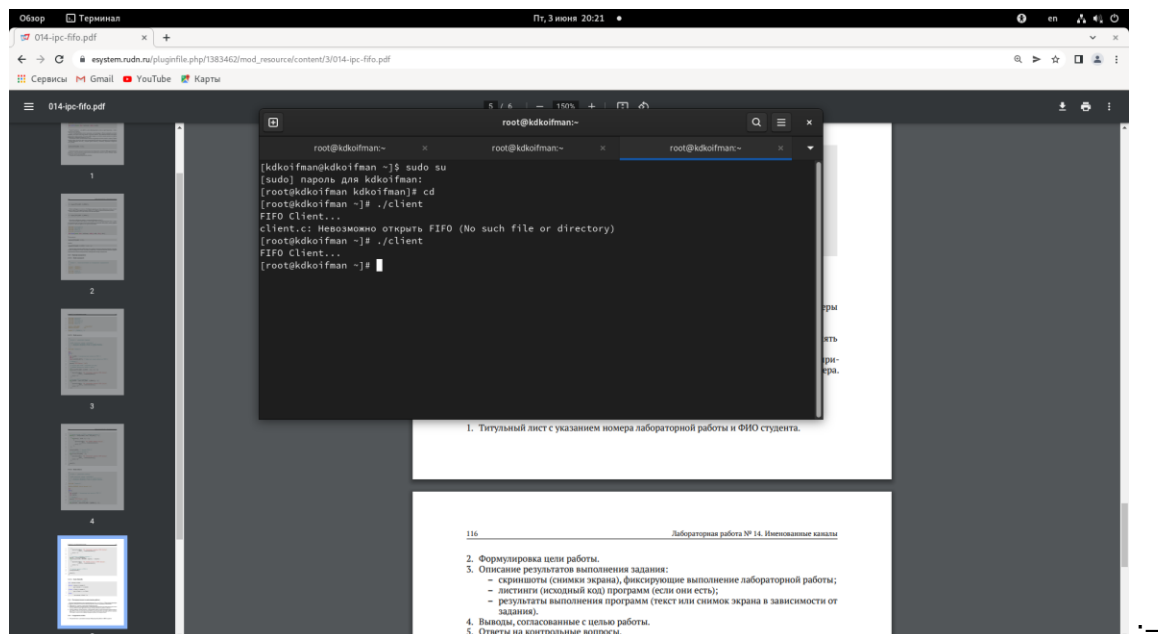


рис.7(3-ий клиент)

Как можно заметить, при повторном запуске программы сервера возникает ошибка при попытке создать файл канала(рис.7).

## Вывод.

В ходе проделанной лабораторной работы мной были усвоены основные работы с именованными каналами.

### Контрольные вопросы.

1. В чем ключевое отличие именованных каналов от неименованных? Основным отличием именованных каналов от неименованных является наличие идентификатора(названия) канала.
2. Возможно ли создание неименованного канала из командной строки? Да, возможно(с помощью оператора `|`).
3. Возможно ли создание именованного канала из командной строки? Да,возможно(с помощью команд `mkfifo` или `mknod`).
4. Опишите функцию языка C, создающую неименованный канал. Вызов функции `pipe()`, которая кладёт в массив типа `int` размера *[2]* 2 дескриптора открытых файлов. Первый `mas[0]` - для чтения, второй `mas[1]` - для записи.
5. Опишите функцию языка C, создающую именованный канал. Вызов функции `mkfifo()` создаёт файл канала (с именем, заданным макросом `FIFO_NAME`).
6. Что будет в случае прочтения из `fifo` меньшего числа байтов, чем находится в канале? Большого числа байтов? Будет возвращено запрошенное число байт(остаток сохранится до следующего чтения).
7. Аналогично, что будет в случае записи в `fifo` меньшего числа байтов, чем позволяет буфер? Большого числа байтов?
8. Могут ли два и более процессов читать или записывать в канал? Да, могут, но одновременно можно только читать данные.
9. Опишите функцию `write` (тип возвращаемого значения, аргументы и логику работы). Что означает 1 (единица) в вызове этой функции в программе `server.c` (строка 42)?
10. Опишите функцию `strerror`. Функция `strerror` используется для конвертации номера ошибки в текстовое описание этой ошибки.