

# **Именованные каналы**

**Отчёт по лабораторной работе №15**

Мурашко В.В.

# Содержание

<b>1</b>	<b>Теоретическое введение</b>	<b>5</b>
<b>2</b>	<b>Цель работы</b>	<b>6</b>
<b>3</b>	<b>Задание</b>	<b>7</b>
<b>4</b>	<b>Выполнение лабораторной работы</b>	<b>8</b>
<b>5</b>	<b>Выводы</b>	<b>16</b>
<b>6</b>	<b>Библиография</b>	<b>17</b>
<b>7</b>	<b>Контрольные вопросы</b>	<b>18</b>

## **Список таблиц**

## Список иллюстраций

4.1	common.h . . . . .	8
4.2	server.c . . . . .	9
4.3	client.c . . . . .	10
4.4	common.h . . . . .	11
4.5	server.c . . . . .	12
4.6	client.c . . . . .	13
4.7	client2.c . . . . .	14
4.8	./client.c и ./client2.c . . . . .	14
4.9	./server . . . . .	15

# 1 Теоретическое введение

С помощью труб могут общаться только родственные друг другу процессы, полученные с помощью `fork()`. Именованные каналы FIFO позволяют обмениваться данными с абсолютно “чужим” процессом.

С точки зрения ядра ОС FIFO является одним из вариантов реализации трубы. Системный вызов `mkfifo()` предоставляет процессу именованную трубу в виде объекта файловой системы. Как и для любого другого объекта, необходимо предоставлять процессам права доступа в FIFO, чтобы определить, кто может писать, и кто может читать данные. Несколько процессов могут записывать или читать FIFO одновременно. Режим работы с FIFO - полудуплексный, т.е. процессы могут общаться в одном из направлений. Типичное применение FIFO - разработка приложений “клиент - сервер”.

## **2 Цель работы**

Приобретение практических навыков работы с именованными каналами.

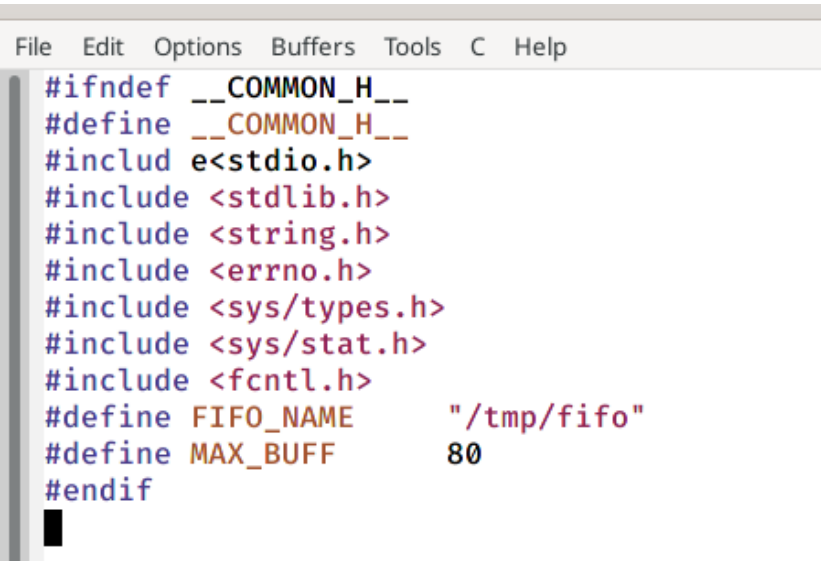
## 3 Задание

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?

## 4 Выполнение лабораторной работы

1. Я изучила приведённые в тексте программы server.c и client.c.

A screenshot of a code editor window with a menu bar (File, Edit, Options, Buffers, Tools, C, Help) and a text area containing C header file code. The code includes preprocessor directives for conditional compilation, standard library headers, and system headers, followed by macro definitions for FIFO\_NAME and MAX\_BUFF.

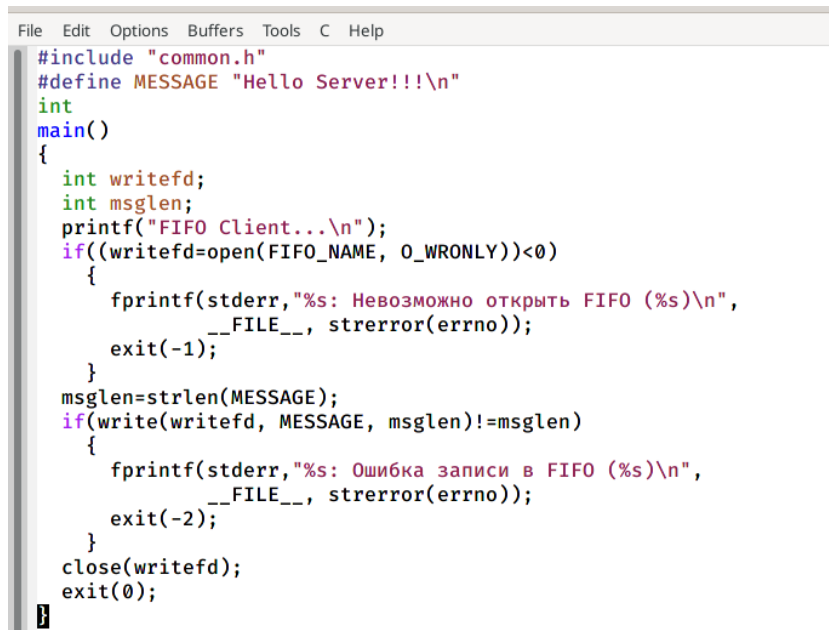
```
File Edit Options Buffers Tools C Help
#ifndef __COMMON_H__
#define __COMMON_H__
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <errno.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#define FIFO_NAME    "/tmp/fifo"
#define MAX_BUFF     80
#endif
```

Рис. 4.1: common.h



```
File Edit Options Buffers Tools C Help
#include "common.h"
int
main()
{
    int readfd;
    int n;
    char buff[MAX_BUFF];
    printf("FIFO Server...\n");
    if(mknod(FIFO_NAME, S_IFIFO|0666,0)<0)
    {
        fprintf(stderr,"%s: Невозможно создать FIFO (%s)\n",
            __FILE__, strerror(errno));
        exit(-1);
    }
    if((readfd=open(FIFO_NAME, O_RDONLY))<0)
    {
        fprintf(stderr,"%s: Невозможно открыть FIFO (%s)\n",
            __FILE__, strerror(errno));
        exit(-2);
    }
    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);
    if(unlink(FIFO_NAME)<0)
    {
        fprintf(stderr,"%s: Невозможно удалить FIFO (%s)\n",
            __FILE__, strerror(errno));
        exit(-4);
    }
    exit(0);
}
```

Рис. 4.2: server.c



```

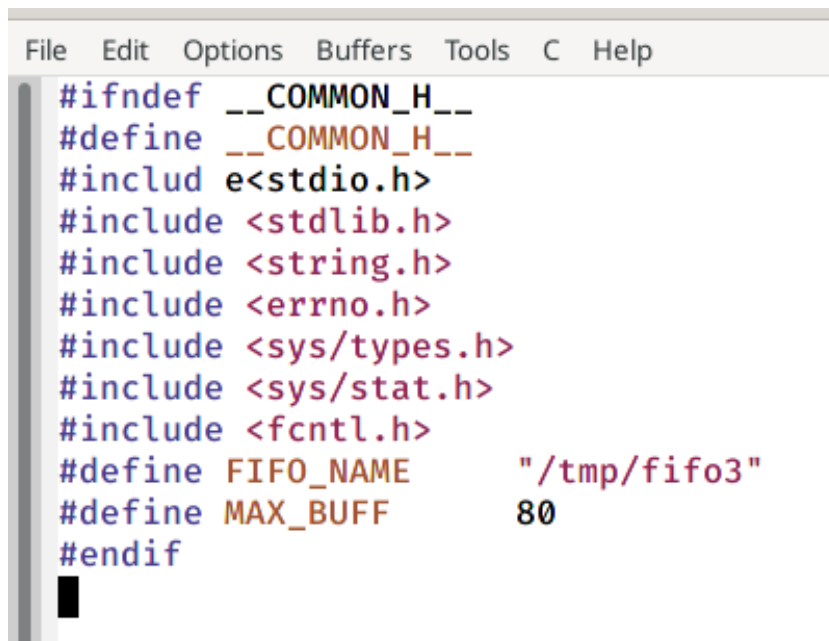
File Edit Options Buffers Tools C Help
#include "common.h"
#define MESSAGE "Hello Server!!!\n"
int
main()
{
    int writefd;
    int msglen;
    printf("FIFO Client...\n");
    if((writefd=open(FIFO_NAME, O_WRONLY))<0)
    {
        fprintf(stderr,"%s: Невозможно открыть FIFO (%s)\n",
            __FILE__, 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);
}

```

Рис. 4.3: client.c

2. Я написала аналогичные программы, внося следующие изменения:

- работает не 1 клиент, а несколько (например, два).
- клиенты передают текущее время с некоторой периодичностью (например, раз в пять секунд). Использовала функцию `sleep()` для приостановки работы клиента.
- сервер работает не бесконечно, а прекращает работу через некоторое время (например, 30 сек). Использовала функцию `clock()` для определения времени работы сервера.

A screenshot of a code editor window. The title bar at the top contains the menu items: File, Edit, Options, Buffers, Tools, C, and Help. The editor area displays the content of a C header file named common.h. The code includes a guard to prevent multiple inclusions, followed by several standard system headers, and two macro definitions for FIFO\_NAME and MAX\_BUFF. A black cursor is visible on the line following the #endif.

```
#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/fifo3"
#define MAX_BUFF      80
#endif
```

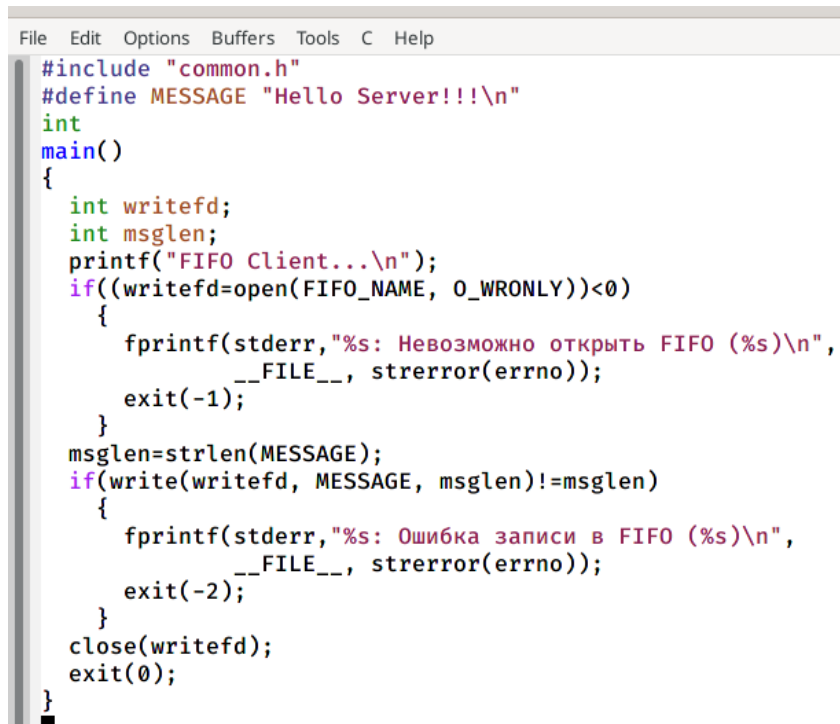
Рис. 4.4: common.h

```

File Edit Options Buffers Tools C Help
#include "common.h"
int
main()
{
    int readfd;
    int n;
    char buff[MAX_BUFF];
    printf("FIFO Server...\n");
    if(mknod(FIFO_NAME, S_IFIFO|0666,0)<0)
    {
        fprintf(stderr,"%s: Невозможно создать FIFO (%s)\n",
            __FILE__, strerror(errno));
        exit(-1);
    }
    if((readfd=open(FIFO_NAME, O_RDONLY))<0)
    {
        fprintf(stderr,"%s: Невозможно открыть FIFO (%s)\n",
            __FILE__, strerror(errno));
        exit(-2);
    }
    clock_t now=time(NULL), start=time(NULL);
    while(now-start<30)
    {
        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);
            }
        }
        now=time(NULL)
    }
    printf("\n-----\nserver timeout\n%u seconds passed!\n-----\n",now-start);
    close(readfd);
    if(unlink(FIFO_NAME)<0)
    {
        fprintf(stderr,"%s: Невозможно удалить FIFO (%s)\n",
            __FILE__, strerror(errno));
        exit(-4);
    }
    exit(0);
}

```

Рис. 4.5: server.c



```
File Edit Options Buffers Tools C Help
#include "common.h"
#define MESSAGE "Hello Server!!!\n"
int
main()
{
    int writefd;
    int msglen;
    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);
}
```

Рис. 4.6: client.c

```
File Edit Options Buffers Tools C Help
#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);
    }
}
```

Рис. 4.7: client2.c

```
vvmurashko1@dk4n64 ~/laboratory/lab15 $ ./client
FIFO Client...
vvmurashko1@dk4n64 ~/laboratory/lab15 $ ./client2
FIFO Client...
FIFO Client...
FIFO Client...
FIFO Client...
FIFO Client...
FIFO Client...
vvmurashko1@dk4n64 ~/laboratory/lab15 $ █
```

Рис. 4.8: ./client.c и ./client2.c

```
vvmurashko1@dk4n64 ~/laboratory/lab15 $ ./server
FIFO Server...
Hello Server!!!
Hello Server!!!
Hello Server!!!
Hello Server!!!
Hello Server!!!
Hello Server!!!
Hello Server!!!
Hello Server!!!
Hello Server!!!

-----
server timeout
41 seconds passed!
-----
```

Рис. 4.9: ./server

В случае, если сервер завершит работу, не закрыв канал, файл FIFO не удалится, поэтому его в следующий раз создать будет нельзя и вылезет ошибка, следовательно, работать ничего не будет.

## **5 Выводы**

Я приобрела практические навыки работы с именованными каналами.



## 6 Библиография

[https://www.opennet.ru/docs/RUS/linux\\_parallel/node17.html](https://www.opennet.ru/docs/RUS/linux_parallel/node17.html)

## 7 Контрольные вопросы

1. 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.
2. Создание неименованного канала из командной строки невозможно.
3. Создание именованного канала из командной строки возможно.
4. `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. `int mkfifo (const char *pathname, mode_t mode) ; mkfifo(FIFO_NAME, 0600)`  
; 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`).
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 много направленная 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 записывает 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.