

# Отчёт по лабораторной работе №14

Именованные каналы

Ганина Таисия Сергеевна, НКАбд-01-22

# Содержание

1	Цель работы	5
2	Задачи	6
3	Ход работы	7
4	Вывод	13
5	Контрольные вопросы.	14
	Список литературы	16

## Список иллюстраций

3.1	Файл common.h . . . . .	7
3.2	Файл server.c . . . . .	8
3.3	Файл client.c . . . . .	9
3.4	Файл client2.c . . . . .	10
3.5	makefile . . . . .	11
3.6	Запуск makefile и server . . . . .	11
3.7	Запуск makefile и server . . . . .	12
3.8	Запуск client . . . . .	12

## Список таблиц

# 1 Цель работы

Приобретение практических навыков работы с именованными каналами.

## 2 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?

## 3 Ход работы

1. Я создала файлы *common.h*, *server.c*, *client.c*, *client2.c*. Скопировала основной код из теоретической части лабораторной работы и немного подкорректировала его.(рис. 3.1, 3.2, 3.3, 3.4)

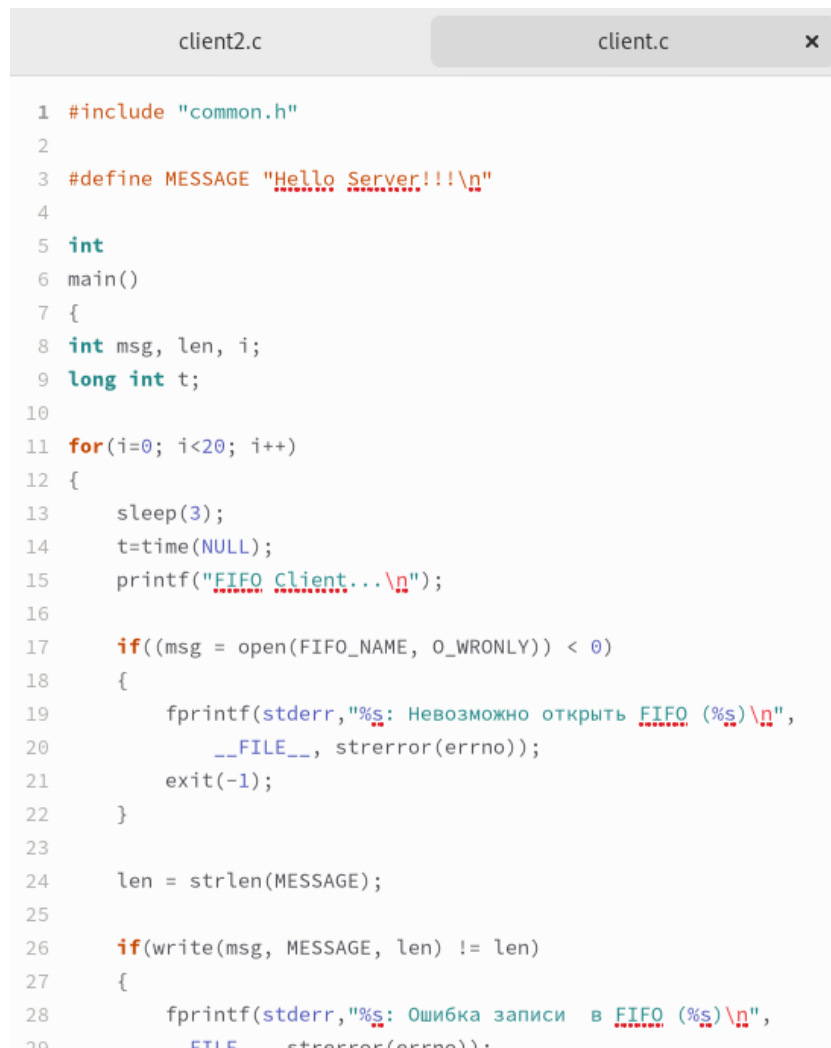
client2.c	client.c
<pre>1  #ifndef __COMMON_H__ 2  #define __COMMON_H__ 3 4  #include &lt;stdio.h&gt; 5  #include &lt;stdlib.h&gt; 6  #include &lt;string.h&gt; 7  #include &lt;errno.h&gt; 8  #include &lt;sys/types.h&gt; 9  #include &lt;sys/stat.h&gt; 10 #include &lt;fcntl.h&gt; 11 12 #define FIFO_NAME "/tmp/fifo" 13 #define MAX_BUFF 80 14 15 #endif /* __COMMON_H__ */</pre>	

Рис. 3.1: Файл common.h

client2.c	client.c
<pre> 1  #include "common.h" 2  int 3  main() 4  { 5      int readfd; 6      int n; 7      char buff[MAX_BUFF]; 8      printf("FIFO Server...\n"); 9 10     if(mknod(FIFO_NAME, S_IFIFO   0666, 0) &lt; 0) 11     { 12         fprintf(stderr, "%s: Невозможно создать FIFO (%s)\n", 13             __FILE__, strerror(errno)); 14         exit(-1); 15     } 16 17     if((readfd = open(FIFO_NAME, O_RDONLY)) &lt; 0) 18     { 19         fprintf(stderr, "%s: Невозможно открыть FIFO (%s)\n", 20             __FILE__, strerror(errno)); 21         exit(-2); 22     } 23     clock_t now=time(NULL), start=time(NULL); 24     while(now-start&lt;30) 25     { 26         while((n = read(readfd, buff, MAX_BUFF)) &gt; 0) 27         { 28             if(write(1, buff, n) != n) 29             { 30                 fprintf(stderr, "%s: Ошибка вывода (%s)\n", 31                     __FILE__, strerror(errno)); 32             } 33         } </pre>	

Рис. 3.2: Файл server.c





```
1 #include "common.h"
2
3 #define MESSAGE "Hello Server!!!\n"
4
5 int
6 main()
7 {
8     int msg, len, i;
9     long int t;
10
11     for(i=0; i<20; i++)
12     {
13         sleep(3);
14         t=time(NULL);
15         printf("FIFO Client...\n");
16
17         if((msg = open(FIFO_NAME, O_WRONLY)) < 0)
18         {
19             fprintf(stderr,"%s: Невозможно открыть FIFO (%s)\n",
20                     __FILE__, strerror(errno));
21             exit(-1);
22         }
23
24         len = strlen(MESSAGE);
25
26         if(write(msg, MESSAGE, len) != len)
27         {
28             fprintf(stderr,"%s: Ошибка записи в FIFO (%s)\n",
29                     __FILE__, strerror(errno));
```

Рис. 3.3: Файл client.c



```
1 #include "common.h"
2
3 #define MESSAGE "Hello Server!!!\n"
4
5 int
6 main()
7 {
8     int writefd, msglen, count;
9     long long int t;
10    char message[10];
11
12    for(count=0; count<=5; ++count)
13    {
14        sleep(5);
15        t=(long long int) time(0);
16        sprintf(message, "%lli", t);
17        if((writefd = open(FIFO_NAME, O_WRONLY)) < 0)
18        {
19            fprintf(stderr,"%s: Невозможно открыть FIFO (%s)\n",
20                __FILE__, strerror(errno));
21            exit(-1);
22        }
23
24        msglen = strlen(MESSAGE);
25        if(write(writefd, MESSAGE, msglen) != msglen)
26        {
27            fprintf(stderr,"%s: Ошибка записи в FIFO (%s)\n",
28                __FILE__, strerror(errno));
29            exit(-1);
30        }
31    }
```

Рис. 3.4: Файл client2.c

2. Создала *makefile*. (рис. 3.5)



Рис. 3.5: makefile

3. Запустила *makefile*. Затем запустила *server*. (рис. 3.6, 3.7)

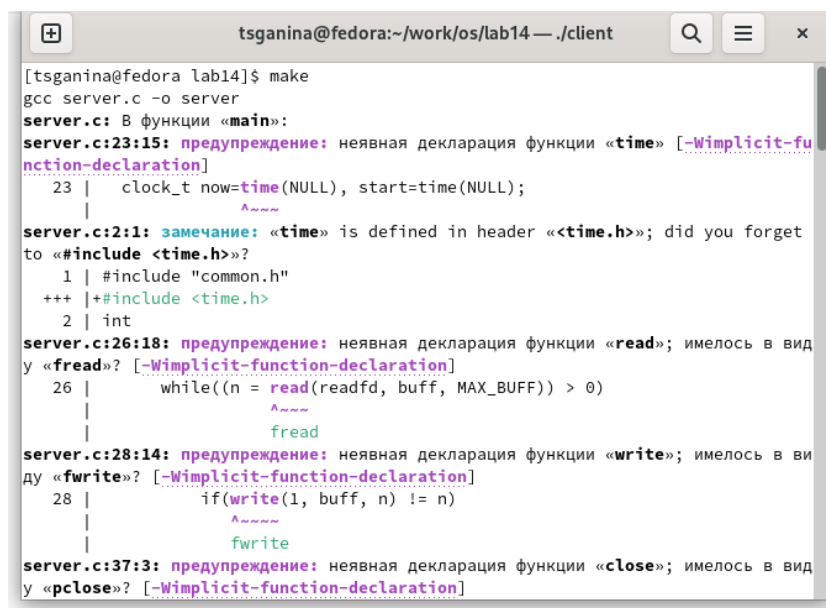
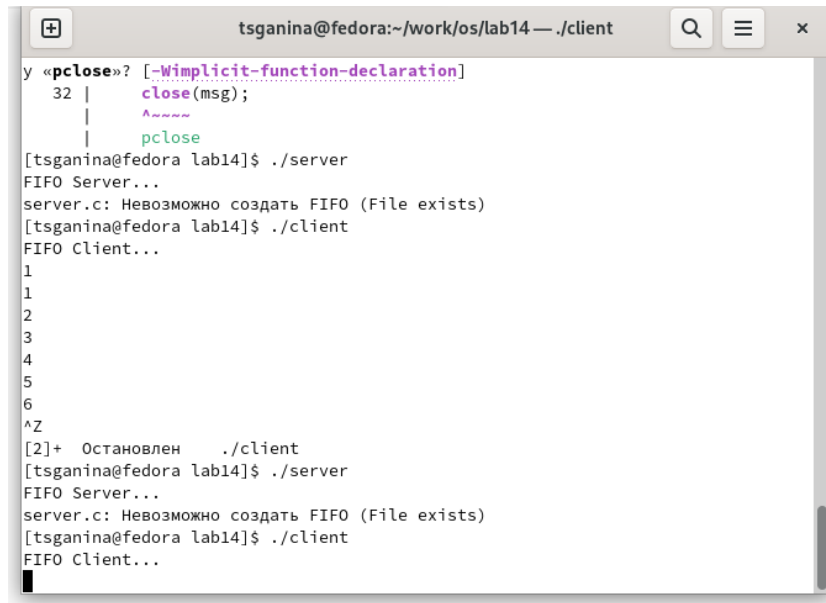


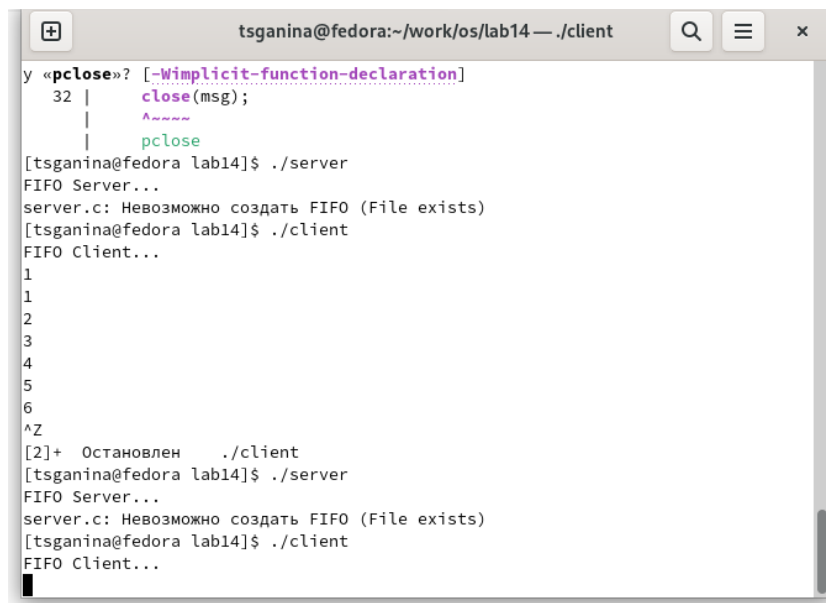
Рис. 3.6: Запуск makefile и server



```
tsganina@fedora:~/work/os/lab14 — ./client
y «pclose»? [-Wimplicit-function-declaration]
32 |     close(msg);
    |     ^~~~~~
    |     pclose
[tsganina@fedora lab14]$ ./server
FIFO Server...
server.c: Невозможно создать FIFO (File exists)
[tsganina@fedora lab14]$ ./client
FIFO Client...
1
1
2
3
4
5
6
^Z
[2]+  Остановлен ./client
[tsganina@fedora lab14]$ ./server
FIFO Server...
server.c: Невозможно создать FIFO (File exists)
[tsganina@fedora lab14]$ ./client
FIFO Client...
```

Рис. 3.7: Запуск makefile и server

#### 4. Запустила client (рис. 3.8)



```
tsganina@fedora:~/work/os/lab14 — ./client
y «pclose»? [-Wimplicit-function-declaration]
32 |     close(msg);
    |     ^~~~~~
    |     pclose
[tsganina@fedora lab14]$ ./server
FIFO Server...
server.c: Невозможно создать FIFO (File exists)
[tsganina@fedora lab14]$ ./client
FIFO Client...
1
1
2
3
4
5
6
^Z
[2]+  Остановлен ./client
[tsganina@fedora lab14]$ ./server
FIFO Server...
server.c: Невозможно создать FIFO (File exists)
[tsganina@fedora lab14]$ ./client
FIFO Client...
```

Рис. 3.8: Запуск client

## 4 Вывод

Мы научились пользоваться именованными каналами.

## 5 Контрольные вопросы.

1. Именованные каналы отличаются от неименованных наличием идентификатора канала, который представлен как специальный файл (соответственно имя именованного канала — это имя файла).
2. Создание неименованного канала из командной строки возможно командой `pipe`.
3. Создание именованного канала из командной строки возможно с помощью `mkfifo`.
4. Функция языка C, создающая 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. Функция языка C, создающая 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, возвращается требуемое число байтов, остаток сохраняется для следующих чтений. При чтении большего числа

байтов, возвращается доступное число байтов 7. 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. 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 `SIGPIPE`, а вызов `write(2)` возвращает 0 с установкой ошибки (`errno=EPipe`) (если процесс не установил обработки сигнала `SIGPIPE`, производится обработка по умолчанию – процесс завершается).

7. Два и более процессов могут читать и записывать в канал.
8. Функция `write` записывает `length` байтов из буфера `buffer` в файл, определенный дескриптором файла `fd`. Эта операция чисто 'двоичная' и без буферизации. При единице возвращает действительное число байтов. Функция `write` возвращает число действительно записанных в файл байтов или -1 при ошибке, устанавливая при этом `errno`.
9. Строковая функция `strerror` - функция языков C/C++, транслирующая код ошибки, который обычно хранится в глобальной переменной `errno`, в сообщение об ошибке, понятном человеку.

# Список литературы

1. Руководство к лабораторной работе №14.