

# **Лабораторная работа No 14.**

**Именованные каналы**

Валиева Марина Русланбековна

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>5</b>
<b>2</b>	<b>Задание</b>	<b>6</b>
<b>3</b>	<b>Выполнение лабораторной работы</b>	<b>7</b>
<b>4</b>	<b>Выводы</b>	<b>12</b>
<b>5</b>	<b>Контрольные вопросы</b>	<b>13</b>

## Список иллюстраций

3.1	создание файлов . . . . .	7
3.2	изменение . . . . .	8
3.3	изменение . . . . .	9
3.4	изменение . . . . .	10
3.5	неизменение . . . . .	10

## Список таблиц

# **1 Цель работы**

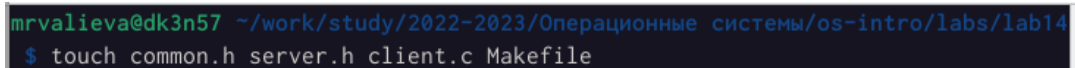
Приобретение практических навыков работы с именованными каналами.

## 2 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?

### 3 Выполнение лабораторной работы


1. Для начала я создала необходимые файлы с помощью команды touch и открыла редактор emacs для их редактирования.

A terminal window with a dark background. The prompt is 'mrvalieva@dk3n57' followed by the path '~/work/study/2022-2023/Операционные системы/os-intro/labs/lab14'. The command '\$ touch common.h server.h client.c Makefile' is entered and executed.

```
mrvalieva@dk3n57 ~/work/study/2022-2023/Операционные системы/os-intro/labs/lab14
$ touch common.h server.h client.c Makefile
```

Рис. 3.1: создание файлов

2. Далее я изменила коды программ, представленных в лабораторной работе. В файл common.h добавила стандартные заголовочные файлы unistd.h и time.h, необходимые для работы кодов других файлов. Common.h предназначен для заголовочных файлов, чтобы в остальных программах их не прописывать каждый раз.

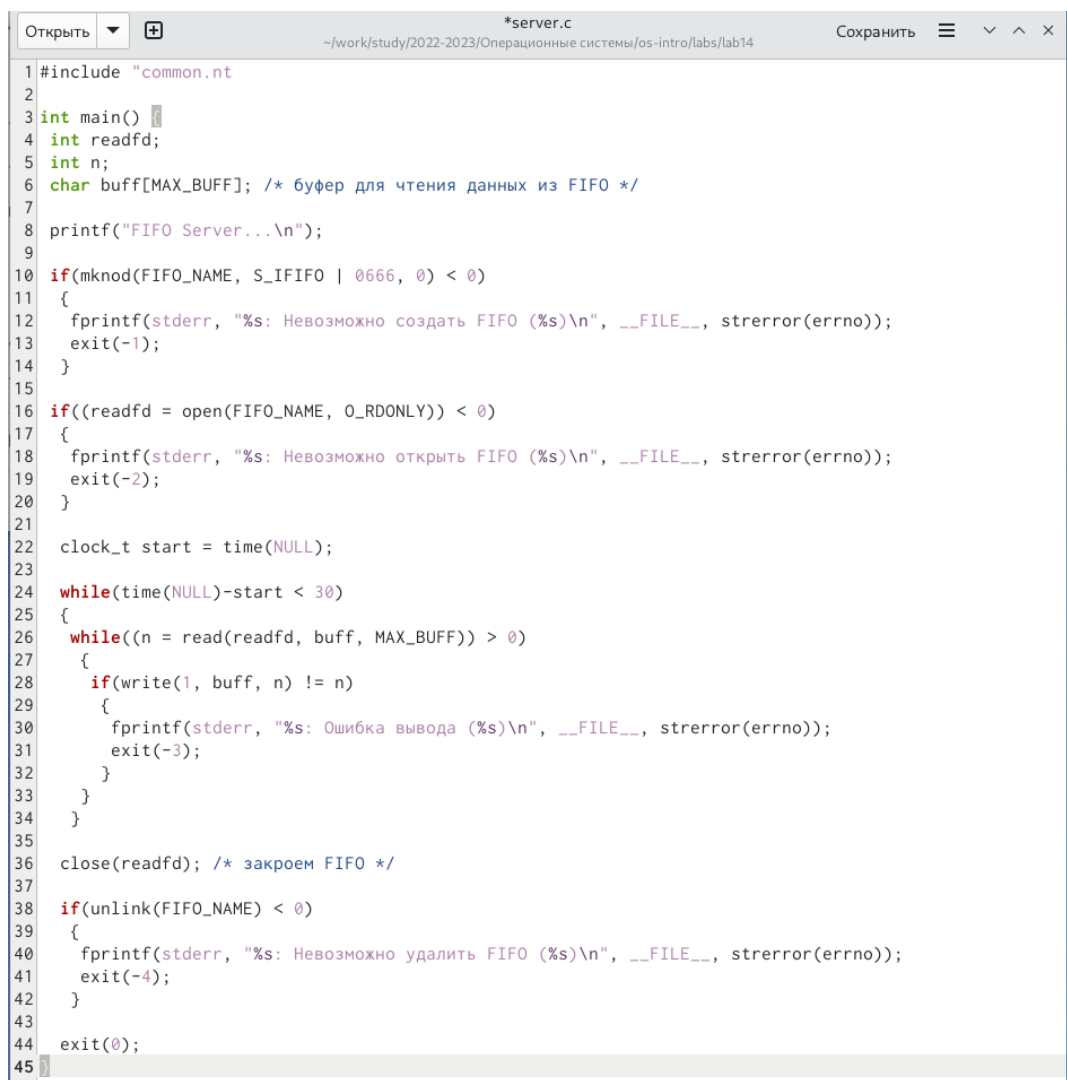


```
1 #ifndef __COMMON_H__
2 #define __COMMON_H__
3
4 #include <stdio.h>
5 #include <stdlib.h>
6 #include <string.h>
7 #include <errno.h>
8 #include <sys/types.h>
9 #include <sys/stat.h>
10 #include <fcntl.h>
11 #include <fcntl.h>
12 #include <unistd.h>
13 #include <time.h>
14
15 #define FIFO_NAME "/tmp/fifo"
16 #define MAX_BUFF 80
17
18 #endif
```

Рис. 3.2: изменение

В файл `server.c` добавила цикл `while` для контроля за временем работы сервера. Разница между текущим временем `time(NULL)` и временем начала работы `clock_t start=time(NULL)` (инициализация до цикла) не должна превышать 30 секунд.

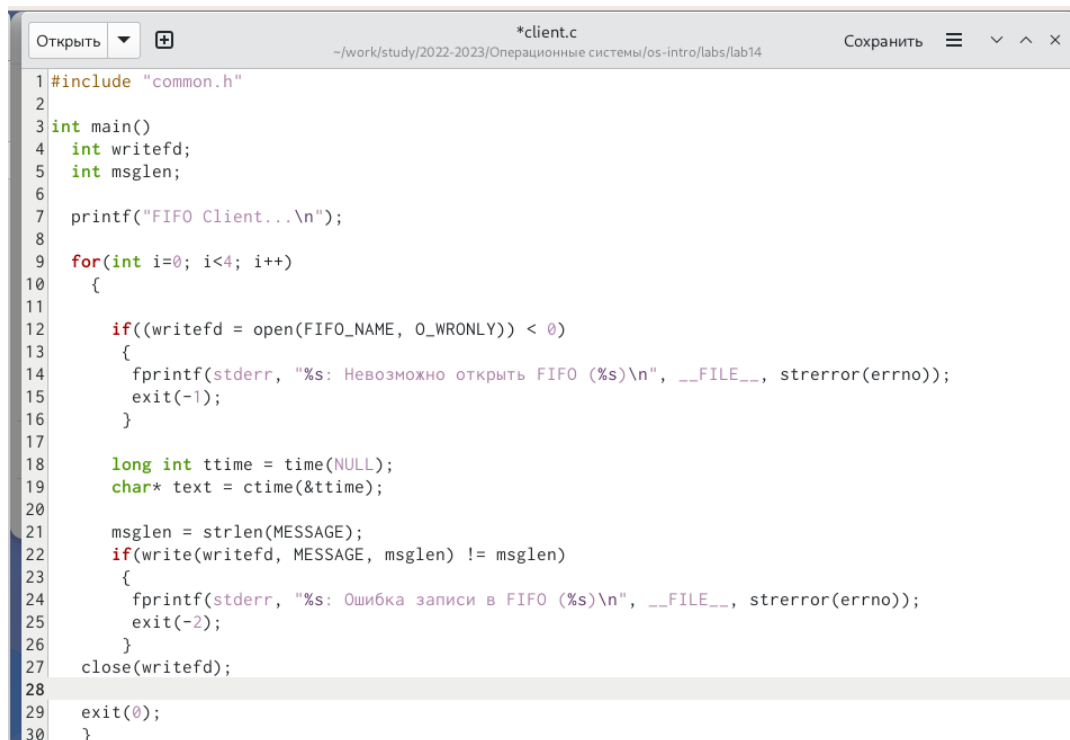




```
1 #include "common.h"
2
3 int main() {
4     int readfd;
5     int n;
6     char buff[MAX_BUFF]; /* буфер для чтения данных из FIFO */
7
8     printf("FIFO Server...\n");
9
10    if(mknod(FIFO_NAME, S_IFIFO | 0666, 0) < 0)
11    {
12        fprintf(stderr, "%s: Невозможно создать FIFO (%s)\n", __FILE__, strerror(errno));
13        exit(-1);
14    }
15
16    if((readfd = open(FIFO_NAME, O_RDONLY)) < 0)
17    {
18        fprintf(stderr, "%s: Невозможно открыть FIFO (%s)\n", __FILE__, strerror(errno));
19        exit(-2);
20    }
21
22    clock_t start = time(NULL);
23
24    while(time(NULL) - start < 30)
25    {
26        while((n = read(readfd, buff, MAX_BUFF)) > 0)
27        {
28            if(write(1, buff, n) != n)
29            {
30                fprintf(stderr, "%s: Ошибка вывода (%s)\n", __FILE__, strerror(errno));
31                exit(-3);
32            }
33        }
34    }
35
36    close(readfd); /* закроем FIFO */
37
38    if(unlink(FIFO_NAME) < 0)
39    {
40        fprintf(stderr, "%s: Невозможно удалить FIFO (%s)\n", __FILE__, strerror(errno));
41        exit(-4);
42    }
43
44    exit(0);
45 }
```

Рис. 3.3: изменение

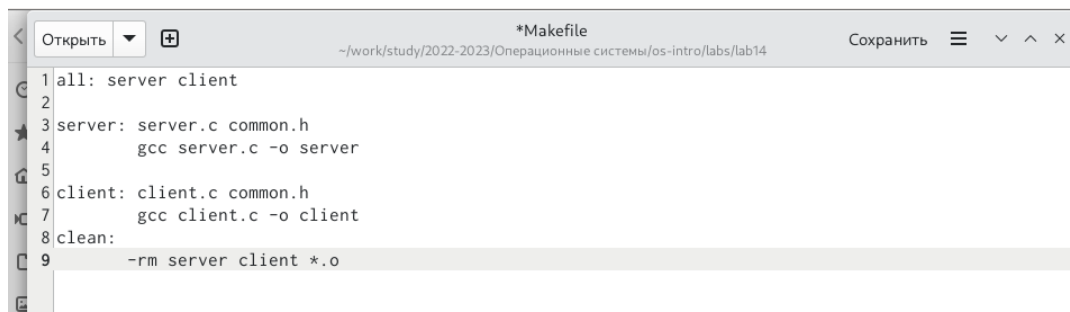
В файл client.c добавила цикл, который отвечает за количество сообщений о текущем времени (4 сообщения), которое получается в результате выполнения команд, и команду sleep(5) для приостановки работы клиента на 5 секунд.



```
1 #include "common.h"
2
3 int main()
4 {
5     int writefd;
6     int msglen;
7
8     printf("FIFO Client...\n");
9
10    for(int i=0; i<4; i++)
11    {
12        if((writefd = open(FIFO_NAME, O_WRONLY)) < 0)
13        {
14            fprintf(stderr, "%s: Невозможно открыть FIFO (%s)\n", __FILE__, strerror(errno));
15            exit(-1);
16        }
17
18        long int ttime = time(NULL);
19        char* text = ctime(&ttime);
20
21        msglen = strlen(MESSAGE);
22        if(write(writefd, MESSAGE, msglen) != msglen)
23        {
24            fprintf(stderr, "%s: Ошибка записи в FIFO (%s)\n", __FILE__, strerror(errno));
25            exit(-2);
26        }
27        close(writefd);
28    }
29    exit(0);
30 }
```

Рис. 3.4: изменение

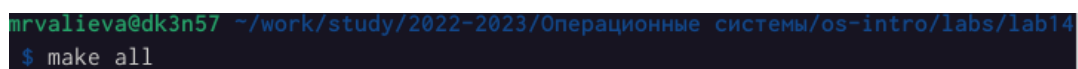
Makefile (файл для сборки) не изменяла



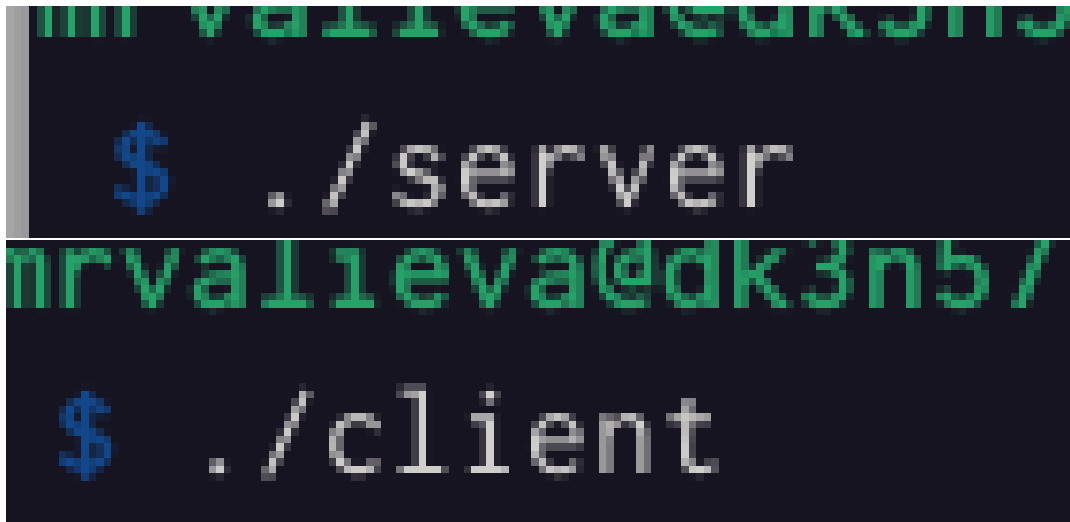
```
1 all: server client
2
3 server: server.c common.h
4     gcc server.c -o server
5
6 client: client.c common.h
7     gcc client.c -o client
8 clean:
9     -rm server client *.o
```

Рис. 3.5: неизменение

3. После написания кодов, я, используя команду «make all», скомпилировала необходимые файлы



```
mrvalieva@dk3n57 ~/work/study/2022-2023/Операционные системы/os-intro/labs/lab14
$ make all
```

A screenshot of a terminal window with a dark background. The prompt 'nrval1eva@dk3n5 /' is visible in green. The command '\$ ./server' is entered in white, and the command '\$ ./client' is entered in white on the next line.

```
nrval1eva@dk3n5 /  
$ ./server  
nrval1eva@dk3n5 /  
$ ./client
```

Далее я проверила работу написанного кода. Открыла 3 консоли (терминала) и запустила: в первом терминале – «./server», в остальных двух – «./client». В результате каждый терминал-клиент вывел по 4 сообщения. Спустя 30 секунд работа сервера была прекращена. Программа работает корректно.

## **4 Выводы**

В результате данной лабораторной работы я приобрела практические навыки работы с именованными каналами.

## 5 Контрольные вопросы

Именованные каналы отличаются от неименованных наличием идентификатора канала, ко

Чтобы создать неименованный канал из командной строки нужно использовать символ |

Чтобы создать именованный канал из командной строки нужно использовать либо коман

Неименованный канал является средством взаимодействия между связанными процессами

Файлы именованных каналов создаются функцией `mkfifo()` или функцией `mknod`: • «int

1. После создания файла канала процессы, участвующие в обмене данными, должны отк

При чтении меньшего числа байтов, чем находится в канале или FIFO, возвращается т

Запись числа байтов, меньшего емкости канала или FIFO, гарантированно атомарно. Э

Количество процессов, которые могут параллельно присоединяться к любому концу кан

У байтов данных в канал. В результате данные в канал записываются поочередно двум

Функция `write` записывает байты `count` из буфера `buffer` в файл, связанный с `handle`.

1 указывает на ошибку; `errno` устанавливается в одно из следующих значений: `EACCES`

р файла, `ENOSPC` – на устройстве нет свободного места. Единица в вызове функции `wr`

Прототип функции `strerror`: «`char * strerror( int errornum );`». Функция `strerror` и библиотек. То есть хорошим тоном программирования будет – использование этой функ