

# **Отчёт по лабораторной работе №15**

**дисциплина: Операционные системы**

Латаева Гюзелия Андреевна

# Содержание

1	Цель работы	5
2	Задание	6
3	Выполнение лабораторной работы	7
4	Выводы	15
5	Контрольные вопросы	16
	Список литературы	18

## Список иллюстраций

3.1	Рисунок 1 . . . . .	7
3.2	Рисунок 2 . . . . .	8
3.3	Рисунок 3 . . . . .	9
3.4	Рисунок 4 . . . . .	10
3.5	Рисунок 5 . . . . .	11
3.6	Рисунок 6 . . . . .	12
3.7	Рисунок 7 . . . . .	13
3.8	Рисунок 8 . . . . .	14

## Список таблиц

# **1 Цель работы**

Приобретение практических навыков работы с именованными каналами.

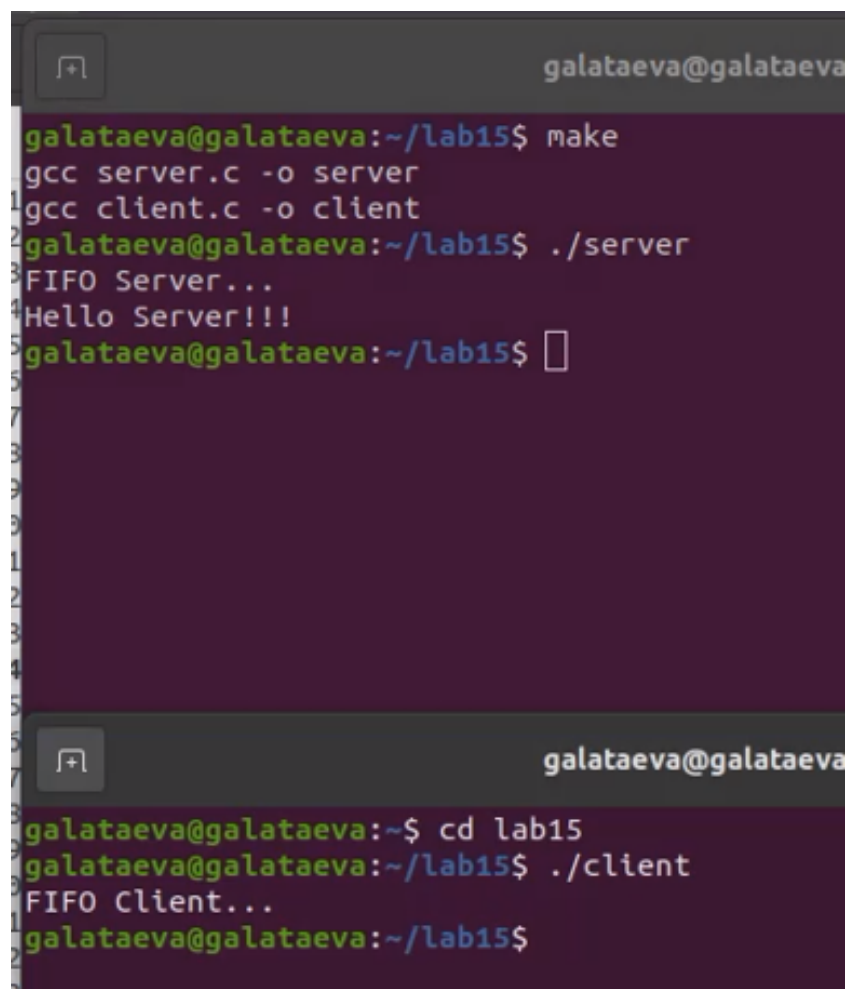
## 2 Задание

Изучить приведённые в тексте лабораторной работы программы `server.c` и `client.c`. Взяв данные примеры за образец, написать 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

Я изучила приведённые в тексте программы и посмотрела их работу: (рис. 3.1)



```
galataeva@galataeva:~/lab15$ make
gcc server.c -o server
gcc client.c -o client
galataeva@galataeva:~/lab15$ ./server
FIFO Server...
Hello Server!!!
galataeva@galataeva:~/lab15$

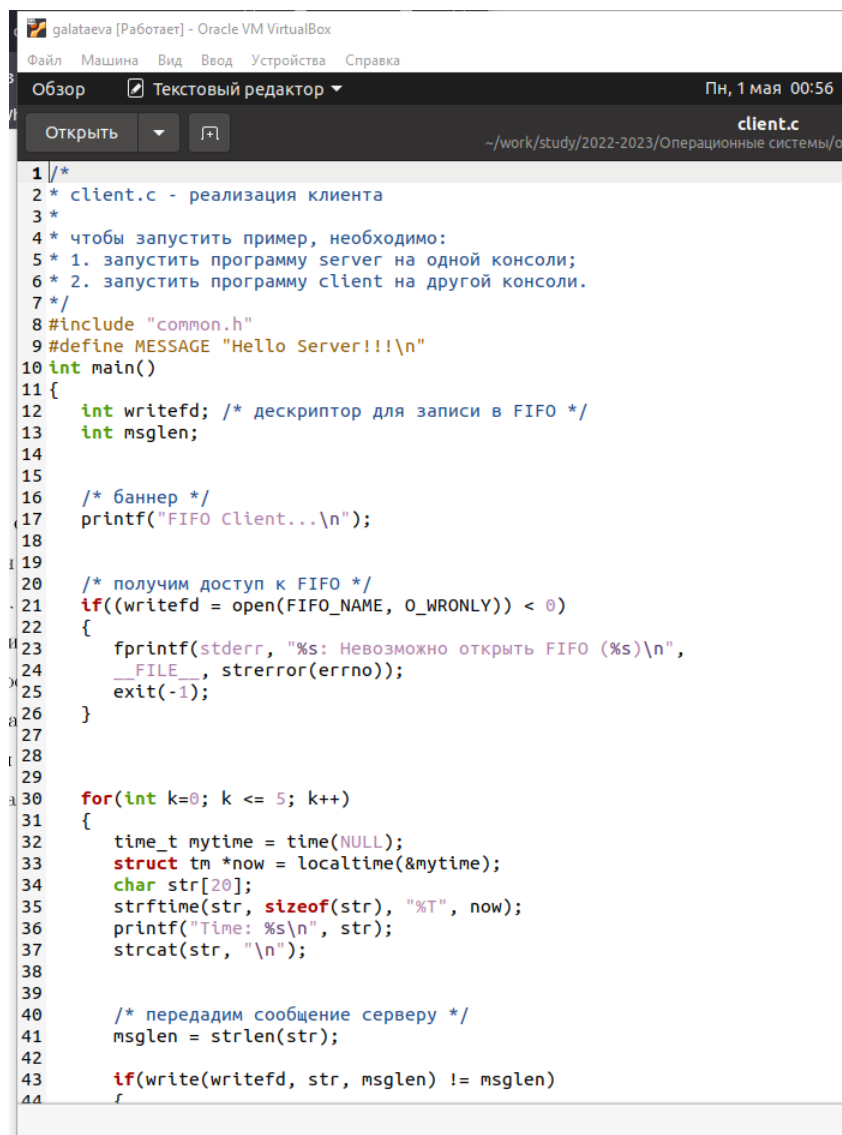
galataeva@galataeva:~$ cd lab15
galataeva@galataeva:~/lab15$ ./client
FIFO Client...
galataeva@galataeva:~/lab15$
```

Рис. 3.1: Рисунок 1

Далее я в каждый файл внесла соответствующие изменения, чтобы

предоставить в программе работу нескольким пользователям, которые раз в 5 секунд будут передавать текущее время, а сам сервер прекращает свою работу через 30 секунд:

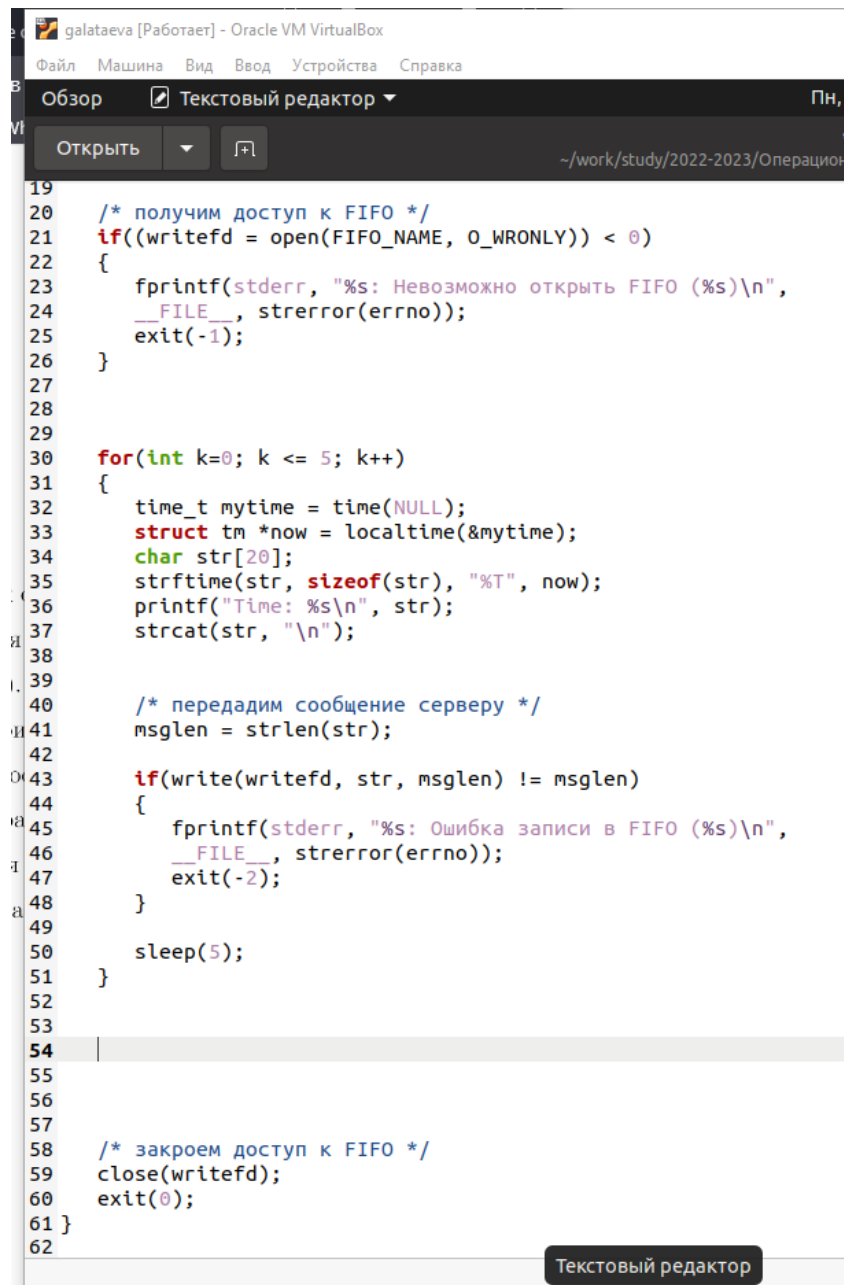
- файл client.c: (рис. 3.2), (рис. 3.3)



```
1 /*
2  * client.c - реализация клиента
3  *
4  * чтобы запустить пример, необходимо:
5  * 1. запустить программу server на одной консоли;
6  * 2. запустить программу client на другой консоли.
7  */
8 #include "common.h"
9 #define MESSAGE "Hello Server!!!\n"
10 int main()
11 {
12     int writefd; /* дескриптор для записи в FIFO */
13     int msglen;
14
15     /* баннер */
16     printf("FIFO Client...\n");
17
18     /* получим доступ к FIFO */
19     if((writefd = open(FIFO_NAME, O_WRONLY)) < 0)
20     {
21         fprintf(stderr, "%s: Невозможно открыть FIFO (%s)\n",
22             __FILE__, strerror(errno));
23         exit(-1);
24     }
25
26     for(int k=0; k <= 5; k++)
27     {
28         time_t mytime = time(NULL);
29         struct tm *now = localtime(&mytime);
30         char str[20];
31         strftime(str, sizeof(str), "%T", now);
32         printf("Time: %s\n", str);
33         strcat(str, "\n");
34
35         /* передадим сообщение серверу */
36         msglen = strlen(str);
37         if(write(writefd, str, msglen) != msglen)
38         {
```

Рис. 3.2: Рисунок 2





The image shows a screenshot of a text editor window titled "galataeva [Работает] - Oracle VM VirtualBox". The window has a menu bar with "Файл", "Машина", "Вид", "Ввод", "Устройства", and "Справка". Below the menu bar is a toolbar with "Обзор", "Текстовый редактор", and "ПН,". The main area of the window displays C code for FIFO operations. The code is as follows:

```
19
20 /* получим доступ к FIFO */
21 if((writefd = open(FIFO_NAME, O_WRONLY)) < 0)
22 {
23     fprintf(stderr, "%s: Невозможно открыть FIFO (%s)\n",
24             __FILE__, strerror(errno));
25     exit(-1);
26 }
27
28
29
30 for(int k=0; k <= 5; k++)
31 {
32     time_t mytime = time(NULL);
33     struct tm *now = localtime(&mytime);
34     char str[20];
35     strftime(str, sizeof(str), "%T", now);
36     printf("Time: %s\n", str);
37     strcat(str, "\n");
38
39     /* передадим сообщение серверу */
40     msglen = strlen(str);
41
42     if(write(writefd, str, msglen) != msglen)
43     {
44         fprintf(stderr, "%s: Ошибка записи в FIFO (%s)\n",
45                 __FILE__, strerror(errno));
46         exit(-2);
47     }
48
49     sleep(5);
50 }
51
52
53
54
55
56
57
58 /* закроем доступ к FIFO */
59 close(writefd);
60 exit(0);
61 }
62
```

The code is written in a dark-themed text editor. The file path at the bottom right is "~/work/study/2022-2023/Операцион".

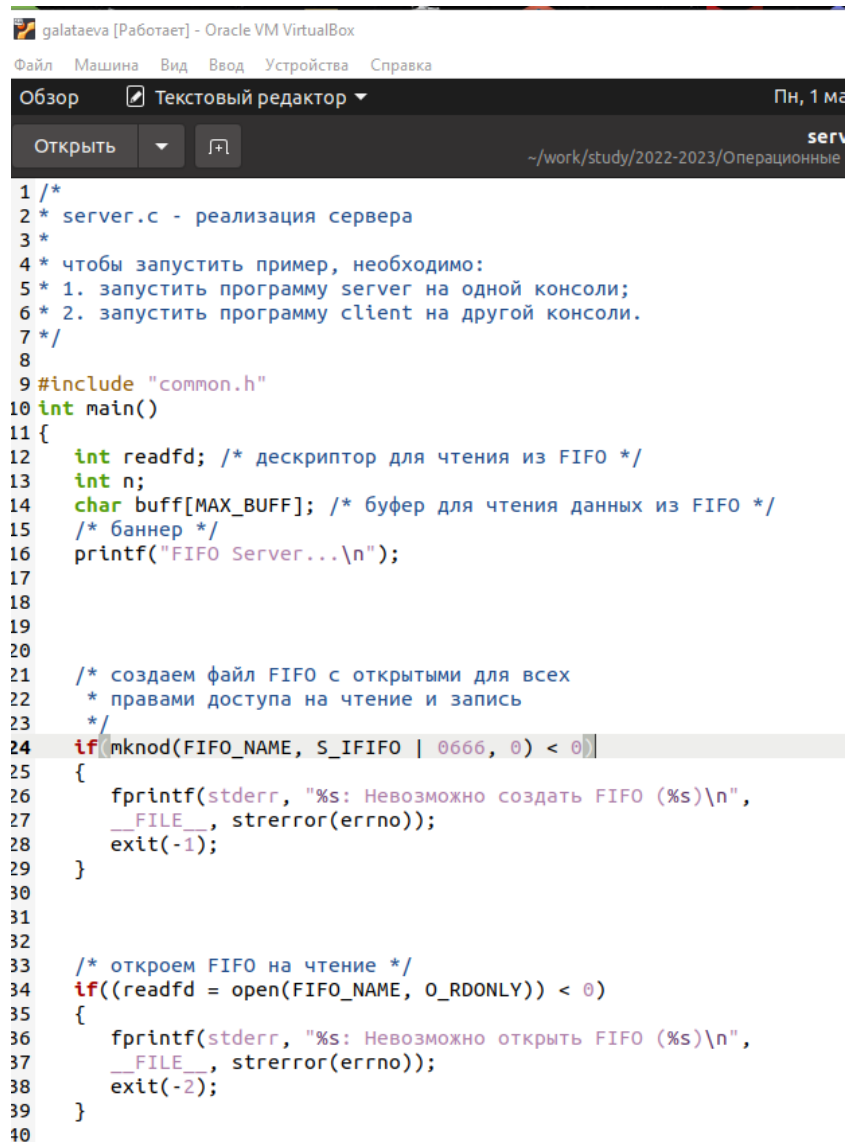
Рис. 3.3: Рисунок 3

- файл common.h: (рис. 3.4)

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

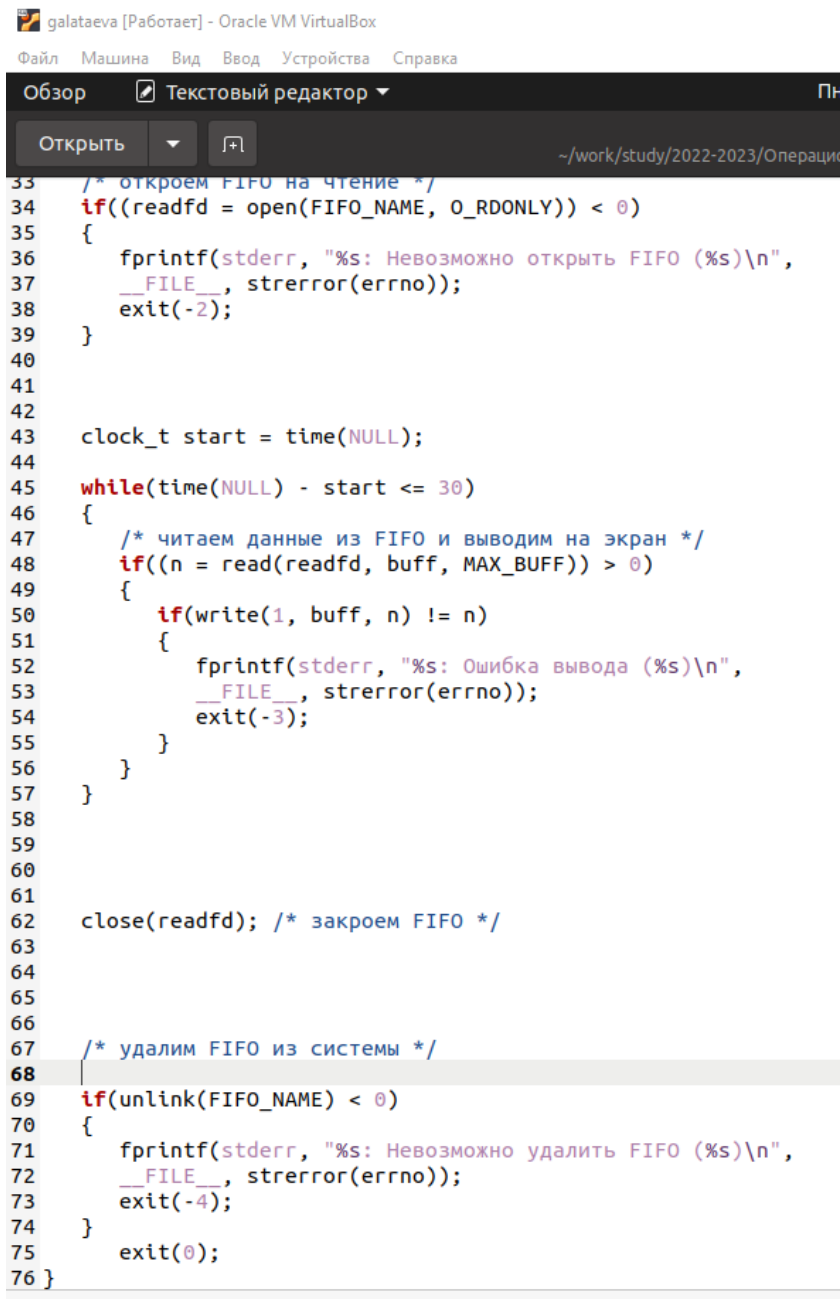
Рис. 3.4: Рисунок 4

- файл server.c: (рис. 3.5), (рис. 3.6)



```
1 /*
2 * server.c - реализация сервера
3 *
4 * чтобы запустить пример, необходимо:
5 * 1. запустить программу server на одной консоли;
6 * 2. запустить программу client на другой консоли.
7 */
8
9 #include "common.h"
10 int main()
11 {
12     int readfd; /* дескриптор для чтения из FIFO */
13     int n;
14     char buff[MAX_BUFF]; /* буфер для чтения данных из FIFO */
15     /* баннер */
16     printf("FIFO Server...\n");
17
18
19
20
21     /* создаем файл FIFO с открытыми для всех
22      * правами доступа на чтение и запись
23      */
24     if (mknod(FIFO_NAME, S_IFIFO | 0666, 0) < 0)
25     {
26         fprintf(stderr, "%s: Невозможно создать FIFO (%s)\n",
27             __FILE__, strerror(errno));
28         exit(-1);
29     }
30
31
32
33     /* откроем FIFO на чтение */
34     if ((readfd = open(FIFO_NAME, O_RDONLY)) < 0)
35     {
36         fprintf(stderr, "%s: Невозможно открыть FIFO (%s)\n",
37             __FILE__, strerror(errno));
38         exit(-2);
39     }
40 }
```

Рис. 3.5: Рисунок 5



The screenshot shows a text editor window titled "galataeva [Работает] - Oracle VM VirtualBox". The menu bar includes "Файл", "Машина", "Вид", "Ввод", "Устройства", and "Справка". The toolbar has "Обзор", "Текстовый редактор", and "Пн". The status bar shows the path "~/work/study/2022-2023/Операци". The code is as follows:

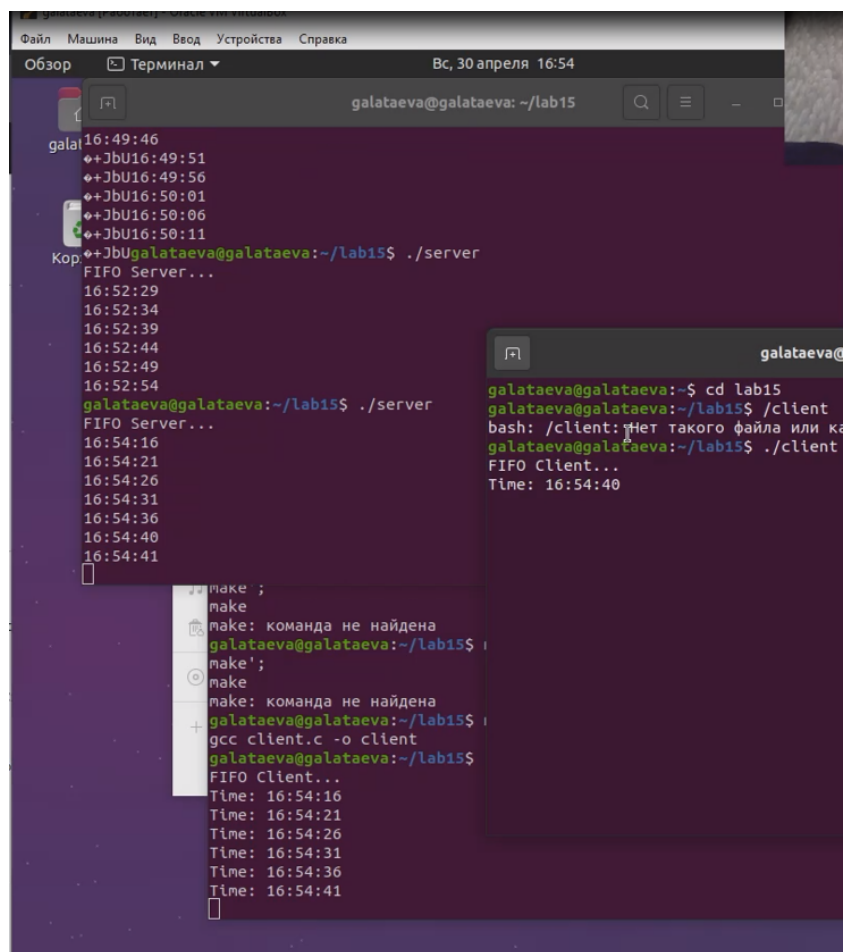
```
33  /* откроем FIFO на чтение */
34  if((readfd = open(FIFO_NAME, O_RDONLY)) < 0)
35  {
36      fprintf(stderr, "%s: Невозможно открыть FIFO (%s)\n",
37              __FILE__, strerror(errno));
38      exit(-2);
39  }
40
41
42
43  clock_t start = time(NULL);
44
45  while(time(NULL) - start <= 30)
46  {
47      /* читаем данные из FIFO и выводим на экран */
48      if((n = read(readfd, buff, MAX_BUFF)) > 0)
49      {
50          if(write(1, buff, n) != n)
51          {
52              fprintf(stderr, "%s: Ошибка вывода (%s)\n",
53                      __FILE__, strerror(errno));
54              exit(-3);
55          }
56      }
57  }
58
59
60
61  close(readfd); /* закроем FIFO */
62
63
64
65
66
67  /* удалим FIFO из системы */
68  if(unlink(FIFO_NAME) < 0)
69  {
70      fprintf(stderr, "%s: Невозможно удалить FIFO (%s)\n",
71              __FILE__, strerror(errno));
72      exit(-4);
73  }
74
75  exit(0);
76 }
```

Рис. 3.6: Рисунок 6

Результат работы с одним пользователем (рис. 3.7) и двумя (рис. 3.8)

```
galataeva@galataeva: ~/lab15
b5V16:46:11
b5V16:46:16
b5V16:46:21
b5V16:46:26
b5Vgalataeva@galataeva:~/lab15$ ./server
FIFO Server...
16:49:46
Kop:
++JbU16:49:51
++JbU16:49:56
++JbU16:50:01
++JbU16:50:06
++JbU16:50:11
++JbUgalataeva@galataeva:~/lab15$ ./server
FIFO Server...
16:52:29
16:52:34
16:52:39
16:52:44
16:52:49
16:52:54
galataeva@galataeva:~/lab15$ sudo snap install q
sudo apt install python3-q-text-as-data # version 1.6.3-1,
ed46-2
See 'snap info q' for additional versions.
galataeva@galataeva:~/lab15$ make
make';
make
make: команда не найдена
galataeva@galataeva:~/lab15$ make
make';
make
make: команда не найдена
galataeva@galataeva:~/lab15$ make
gcc client.c -o client
galataeva@galataeva:~/lab15$ ./client
FIFO Client...
Time: 16:54:16
```

Рис. 3.7: Рисунок 7



```
galataeva@galataeva: ~/lab15
16:49:46
♦+JbU16:49:51
♦+JbU16:49:56
♦+JbU16:50:01
♦+JbU16:50:06
♦+JbU16:50:11
Кор♦+JbUgalataeva@galataeva:~/lab15$ ./server
FIFO Server...
16:52:29
16:52:34
16:52:39
16:52:44
16:52:49
16:52:54
galataeva@galataeva:~/lab15$ ./server
FIFO Server...
16:54:16
16:54:21
16:54:26
16:54:31
16:54:36
16:54:40
16:54:41
make ;
make
make: команда не найдена
galataeva@galataeva:~/lab15$ make ;
make
make: команда не найдена
galataeva@galataeva:~/lab15$ gcc client.c -o client
galataeva@galataeva:~/lab15$
FIFO Client...
Time: 16:54:16
Time: 16:54:21
Time: 16:54:26
Time: 16:54:31
Time: 16:54:36
Time: 16:54:41
```

Рис. 3.8: Рисунок 8

Если же завершить работу сервера и при этом не закрыть канал FIFO, то клиент может зависнуть в ожидании ответа от сервера.

## **4 Выводы**

Я приобрела практические навыки работы с именованными каналами.

## 5 Контрольные вопросы

1. В чем ключевое отличие именованных каналов от неименованных?

Именованные каналы отличаются от неименованных наличием имени, то есть идентификатора канала, потенциально видимого всем процессам системы. Именованный программный канал может служить для общения и синхронизации произвольных процессов, знающих имя данного программного канала и имеющих соответствующие права доступа. Неименованным программным каналом могут пользоваться только создавший его процесс и его потомки (необязательно прямые).

2. Возможно ли создание неименованного канала из командной строки?

Нет.

3. Возможно ли создание именованного канала из командной строки?

Да.

4. Опишите функцию языка C, создающую неименованный канал.

Функция `pipe()`. Она принимает в качестве аргумента массив из двух целых чисел: `pipefd[0]` - чтение из канала и `pipefd[1]` - запись в канал. Функция возвращает 0 в случае успешного выполнения и -1 в случае ошибки.

5. Опишите функцию языка C, создающую именованный канал.



Функция `mkfifo()`. Она в качестве аргумента передается путь к создаваемому каналу и права доступа. Функция возвращает 0 в случае успешного выполнения и -1 в случае ошибки.

6. Что будет в случае прочтения из `fifo` меньшего числа байтов, чем находится в канале? Большого числа байтов?

В первом случае процесс блокируется до тех пор, пока в канале не появятся новые данные. Во втором процесс блокируется до тех пор, пока другой процесс не запишет новые данные в канал.

7. Аналогично, что будет в случае записи в `fifo` меньшего числа байтов, чем позволяет буфер? Большого числа байтов?

В первом случае данные записываются в буфер частично. Во втором процесс блокируется до тех пор, пока другой процесс не прочтет часть данных из канала и не освободит место в буфере.

8. Могут ли два и более процессов читать или записывать в канал?

Да.

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)?

Она используется для записи данных в файловый дескриптор и возвращает количество записанных байтов или -1 в случае ошибки.

10. Опишите функцию `strerror`.

Используется для получения строки с описанием ошибки, связанной с кодом ошибки.

# Список литературы

1. ya.ru