

Лабораторная работа №15

Именованные каналы

Ким Михаил Алексеевич

Содержание

1	Цель работы	3
2	Выполнение лабораторной работы.	4
3	Выводы	14
4	Термины	15

1. Цель работы

Приобретение практических навыков работы с именованными каналами.

2. Выполнение лабораторной работы.

1. В домашнем каталоге создаём подкаталог ~/work. (рис. 2.1)

```
mkdir work
```

2. Создаём в нём файлы: common.h, server.c, client.c, Makefile. (рис. 2.1)

```
cd work
```

```
touch common.h server.c client.c Makefile
```

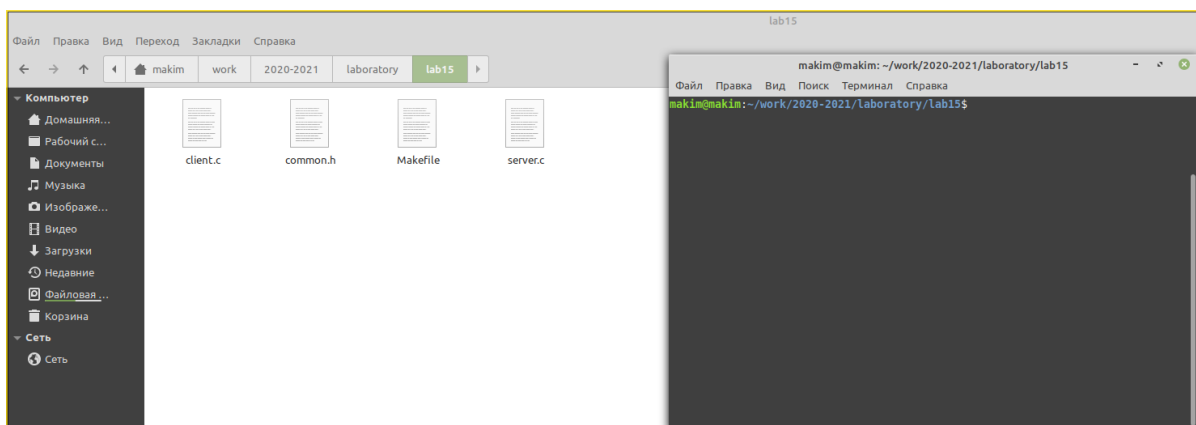


Рис. 2.1: Подготовка рабочей среды

3. Нам необходимо написать программу на основе кода из примера. Программа должна работать с несколькими клиентами. Клиенты передают текущее время с некоторой периодичностью (например, раз в пять секунд). Сервер работает не бесконечно, а прекращает работу через некоторое время (например, 30 сек). Напишем заголовочный файл. Кроме библиотек из примера, добавляем библиотеку для работы со временем. (рис. 2.2)

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

#ifndef __COMMON_H__
#define __COMMON_H__

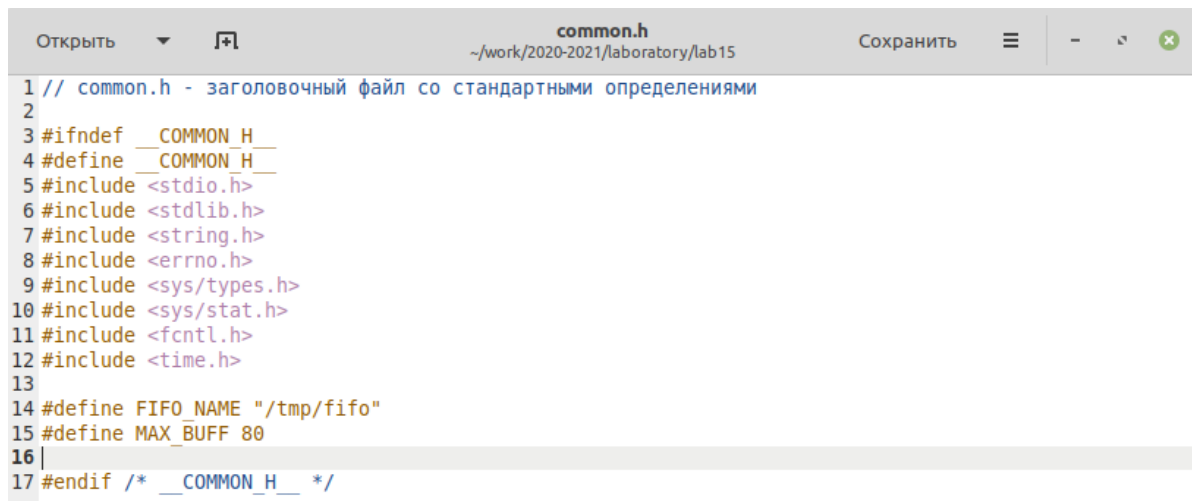
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <errno.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>

// добавляем библиотеку для работы со временем
#include <time.h>

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

#endif

/* __COMMON_H__ */
```



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

Рис. 2.2: Исходный код common.h

4. Пишем файл client.c. В коде зацикливаем запись в файл FIFO. В каждом цикле записываем в переменную время, переводим эту переменную в строку и отправляем на сервер. (рис. 2.3)

```
// client.c - реализация клиента

#include "common.h"

#define MESSAGE "Hello Server!!!\n"

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

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

    if((writefd = open(FIFO_NAME, O_WRONLY)) < 0) /* получим доступ к FIFO */
    {
        fprintf(stderr, "%s: Невозможно открыть FIFO (%s)\n",
```

```

        __FILE__, strerror(errno));
        exit(-1);
    }

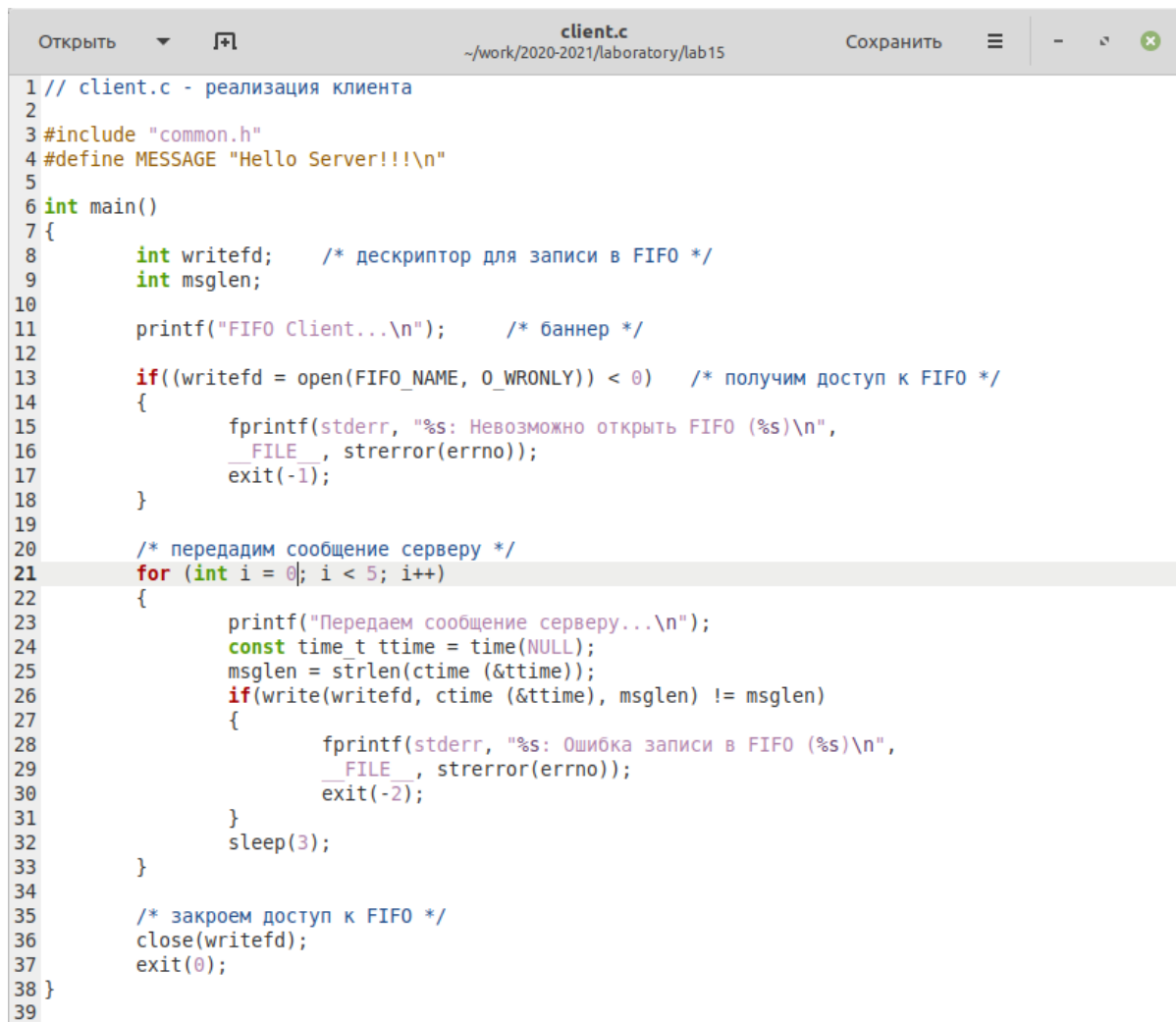
    /* передадим сообщение серверу */
    // Цикл for для передачи нескольких сообщений
    for (int i = 0; i < 5; i++)
    {
        printf("Передаем сообщение серверу...\n");

        // переменная для хранения текущего времени
        const time_t ttime = time(NULL);

        // сохраняем в msglen длину времени, приводя его к адекватному виду
        msglen = strlen(ctime (&ttime));
        if(write(writefd, ctime (&ttime), msglen) != msglen)
        {
            fprintf(stderr, "%s: Ошибка записи в FIFO (%s)\n",
                __FILE__, strerror(errno));
            exit(-2);
        }
        sleep(3);
    }

    /* закроем доступ к FIFO */
    close(writefd);
    exit(0);
}

```



```
1 // client.c - реализация клиента
2
3 #include "common.h"
4 #define MESSAGE "Hello Server!!!\n"
5
6 int main()
7 {
8     int writefd;    /* дескриптор для записи в FIFO */
9     int msglen;
10
11     printf("FIFO Client...\n");    /* баннер */
12
13     if((writefd = open(FIFO_NAME, O_WRONLY)) < 0)    /* получим доступ к FIFO */
14     {
15         fprintf(stderr, "%s: Невозможно открыть FIFO (%s)\n",
16             _FILE_, strerror(errno));
17         exit(-1);
18     }
19
20     /* передадим сообщение серверу */
21     for (int i = 0; i < 5; i++)
22     {
23         printf("Передаем сообщение серверу...\n");
24         const time_t ttime = time(NULL);
25         msglen = strlen(ctime (&ttime));
26         if(write(writefd, ctime (&ttime), msglen) != msglen)
27         {
28             fprintf(stderr, "%s: Ошибка записи в FIFO (%s)\n",
29                 _FILE_, strerror(errno));
30             exit(-2);
31         }
32         sleep(3);
33     }
34
35     /* закроем доступ к FIFO */
36     close(writefd);
37     exit(0);
38 }
39
```

Рис. 2.3: Исходный код client.c

5. Реализуем файл server.c. Создаем переменную, которая хранит время начала запуска программы. Производим чтение файла FIFO, пока не истекли 30 секунд. (рис. 2.4)

```
// server.c

#include "common.h"

int main()
```



```

{
    int readfd; /* дескриптор для чтения из FIFO */
    int n;
    char buff[MAX_BUFF]; /* буфер для чтения данных из FIFO */

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

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

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

    // создаём константу, где будет храниться время начала работы программы
    const time_t start = time(NULL);

    // пока не прошло 30 секунд, сервер будет "принимать сообщения"
    while (time(NULL) - start < 30)

```

```

{
    printf("Сервер работает\n");
    /* 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);
        }
    }
    sleep(3);
}

close(readfd); /* закроем FIFO */
printf("Сервер закрыт\n");

/* удалим FIFO из системы */
if(unlink(FIFO_NAME) < 0)
{
    fprintf(stderr, "%s: Невозможно удалить FIFO (%s)\n",
        __FILE__, strerror(errno));
    exit(-4);
}

exit(0);
}

```

```
Открыть  server.c
~/work/2020-2021/laboratory/lab15

3 #include "common.h"
4
5 int main()
6 {
7     int readfd; /* дескриптор для чтения из FIFO */
8     int n;
9     char buff[MAX_BUFF]; /* буфер для чтения данных из FIFO */
10
11     /* баннер */
12     printf("FIFO Server...\n");
13
14     /* создаем файл FIFO с открытыми для всех правами доступа на чтение и запись */
15     if(mknod(FIFO_NAME, S_IFIFO | 0666, 0) < 0)
16     {
17         fprintf(stderr, "%s: Невозможно создать FIFO (%s)\n",
18             _FILE_, strerror(errno));
19         exit(-1);
20     }
21
22     /* откроем FIFO на чтение */
23     if((readfd = open(FIFO_NAME, O_RDONLY)) < 0)
24     {
25         fprintf(stderr, "%s: Невозможно открыть FIFO (%s)\n",
26             _FILE_, strerror(errno));
27         exit(-2);
28     }
29
30     const time_t start = time(NULL);
31     while (time(NULL) - start < 30)
32     {
33         printf("Сервер работает\n");
34         /* читаем данные из FIFO и выводим на экран */
35         while((n = read(readfd, buff, MAX_BUFF)) > 0)
36         {
37             if(write(1, buff, n) != n)
38             {
39                 fprintf(stderr, "%s: Ошибка вывода (%s)\n",
40                     _FILE_, strerror(errno));
41                 exit(-3);
42             }
43         }
44         sleep(3);
45     }
46
47     close(readfd); /* закроем FIFO */
48     printf("Сервер закрыт\n");
49
50     /* удалим FIFO из системы */
51     if(unlink(FIFO_NAME) < 0)
52     {
53         fprintf(stderr, "%s: Невозможно удалить FIFO (%s)\n",
54             _FILE_, strerror(errno));
55         exit(-4);
56     }
57
58     exit(0);
59 }
```

Рис. 2.4: Исходный код server.c

6. Создаём Makefile и компилируем программу. (рис. 2.5, 2.6)

```
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
```

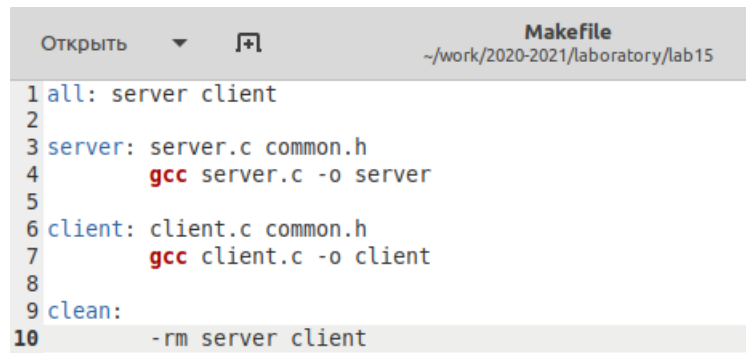


Рис. 2.5: Makefile

make

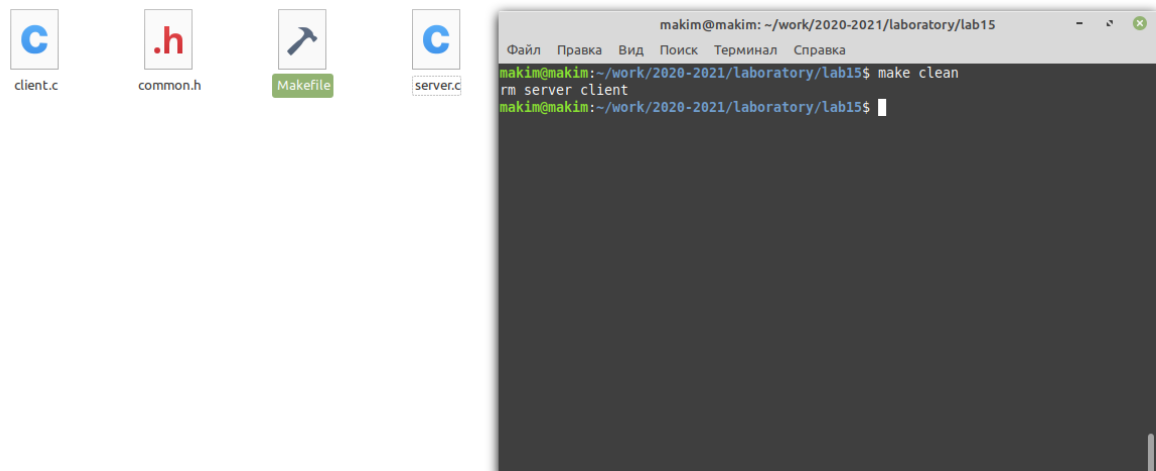


Рис. 2.6: Использование Makefile

7. Наблюдаем процесс работы. (рис. 2.7)

```
makim@makim: ~/work/2020-2021/laboratory/lab15
makim@makim:~/work/2020-2021/Laboratory/lab15$ ./server
FIFO Server...
Сервер работает
Mon Oct 25 22:21:26 2021
Mon Oct 25 22:21:28 2021
Mon Oct 25 22:21:29 2021
Mon Oct 25 22:21:31 2021
Mon Oct 25 22:21:32 2021
Mon Oct 25 22:21:34 2021
Mon Oct 25 22:21:35 2021
Mon Oct 25 22:21:37 2021
Mon Oct 25 22:21:38 2021
Mon Oct 25 22:21:40 2021
Сервер работает
Сервер работает
Сервер работает
Сервер работает
Сервер закрыт
makim@makim:~/work/2020-2021/Laboratory/lab15$

makim@makim:~/work/2020-2021/laboratory/lab15
makim@makim:~/work/2020-2021/Laboratory/lab15$ ./client
FIFO Client...
Передает сообщение серверу...
Передает сообщение серверу...
Передает сообщение серверу...
Передает сообщение серверу...
Передает сообщение серверу...
makim@makim:~/work/2020-2021/Laboratory/lab15$

makim@makim:~/work/2020-2021/laboratory/lab15
makim@makim:~/work/2020-2021/Laboratory/lab15$ ./client
FIFO Client...
Передает сообщение серверу...
Передает сообщение серверу...
Передает сообщение серверу...
Передает сообщение серверу...
makim@makim:~/work/2020-2021/Laboratory/lab15$
```

Рис. 2.7: Результат (1)

3. Выводы

Мы приобрели практические навыки работы с именованными каналами.
Реализовали программу для обмена сообщениями.

4. Термины

- Сообщение - последовательность байтов, передаваемая от одного процесса другому.
- Named pipes - механизм именованных каналов для передачи данных между неродственными процессами.
- FIFO - принцип передачи данных: First In First Out (первым записан — первым прочитан).
- GCC (GNU Compiler Collection) - это набор компиляторов для разного рода языков программирования (C, C++, Java, Фортран и др.).
- Утилита make позволяет автоматизировать процесс преобразования файлов программы из одной формы в другую, отслеживает взаимосвязи между файлами.
- Командный процессор (командная оболочка, интерпретатор команд shell) — это программа, позволяющая пользователю взаимодействовать с операционной системой компьютера.
- Последовательность команд может быть помещена в текстовый файл. Такой файл называется командным.
- Каталог, он же директория, (от английского Directory) – это объект в ФС (файловой системе), необходимый для того, чтобы упростить работу с файлами.

- Домашний каталог - каталог, предназначенный для хранения собственных данных пользователя Linux. Как правило, является текущим непосредственно после регистрации пользователя в системе.
- Команда - записанный по специальным правилам текст (возможно с аргументами), представляющий собой указание на выполнение какой-либо функций (или действий) в операционной системе.