

РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

**Факультет физико-математических и
естественных наук
Кафедра прикладной информатики и теории
вероятностей**

**ОТЧЕТ
ПО ЛАБОРАТОРНОЙ РАБОТЕ № 15**

Именованные каналы

Студент: Тасыбаева Наталья Сергеевна

Группа: НПИбд-02-20

МОСКВА 2021г.

Цель работы: Приобретение практических навыков работы с именованными каналами.

Ход работы:

1. Я изучила приведённые в тексте программы server.c и client.c. Взяв данные примеры за образец, я написала аналогичные программы, внося некоторые изменения.

- Я создала необходимые файлы и разрешила юзеру исполнение (рис.1)

```
nstasihbaeva@dk6n65 ~/lab15 $ emacs
nstasihbaeva@dk6n65 ~/lab15 $ ls -l
итого 3
-rw-r--r-- 1 nstasihbaeva studsci 707 июн  4 16:41 client
.c
-rw-r--r-- 1 nstasihbaeva studsci  0 июн  4 16:43 common
.h
-rw-r--r-- 1 nstasihbaeva studsci  0 июн  4 16:44 Makefile
le
-rw-r--r-- 1 nstasihbaeva studsci 1222 июн  4 16:33 server
.c
nstasihbaeva@dk6n65 ~/lab15 $ chmod u+x client.c
bash: chmod: команда не найдена
nstasihbaeva@dk6n65 ~/lab15 $ chmod u+x client.c
nstasihbaeva@dk6n65 ~/lab15 $ chmod u+x server.c
nstasihbaeva@dk6n65 ~/lab15 $ chmod u+x common.h
nstasihbaeva@dk6n65 ~/lab15 $ chmod u+x Makefile
nstasihbaeva@dk6n65 ~/lab15 $ ls -l
итого 3
-rwxr--r-- 1 nstasihbaeva studsci 707 июн  4 16:41 client
.c
-rwxr--r-- 1 nstasihbaeva studsci  0 июн  4 16:43 common
.h
-rwxr--r-- 1 nstasihbaeva studsci  0 июн  4 16:44 Makefile
le
-rwxr--r-- 1 nstasihbaeva studsci 1222 июн  4 16:33 server
.c
nstasihbaeva@dk6n65 ~/lab15 $
```

Рис.1

- Я написала файл client.c (рис.2)

```
client.c
1 #include "common.h"
2 #define MESSAGE "Hello Server!!!\n"
3 int main()
4 {
5     int writefd;
6     int msglen;
7     printf("FIFO Client...\n");
8     if ((writefd = open(FIFO_NAME, O_WRONLY)) < 0)
9     {
10         fprintf(stderr, "%s: Unable to open FIFO (%s)\n", __FILE__, strerror(errno));
11         exit(-1);
12     }
13     int i;
14     for (i=0; i<3; i++) {
15         sleep(5);
16         long ttime = time(NULL);
17         msglen = strlen(&ttime);
18         if (write(writefd, ctime(&ttime), msglen) != msglen)
19         {
20             fprintf(stderr, "%s: Write error FIFO (%s)\n", __FILE__, strerror(errno));
21             exit(-2);
22         };
23     };
24     close(writefd);
25     exit(0);
26 }
```

Рис.2

- Я написала файл server.c (рис.3)

```

1 #include "common.h"
2 int main()
3 {
4     int readfd;
5     int n;
6     char buff[MAX_BUFF];
7     printf("FIFO Server...\n");
8     if (mknod(FIFO_NAME, S_IFIFO | 0666, 0) < 0)
9     {
10         fprintf(stderr, "%s: Unable to create FIFO (%s)\n", __FILE__, strerror(errno));
11         exit(-1);
12     }
13     int start = time(NULL);
14     while(time(NULL) - start <= 30) {
15         if ((readfd = open(FIFO_NAME, O_RDONLY)) < 0)
16         {
17             fprintf(stderr, "%s: Unable to open FIFO (%s)\n", __FILE__, strerror(errno));
18             exit(-2);
19         }
20         while((n=read(readfd, buff, MAX_BUFF)) > 0)
21         {
22             if (write(1, buff, n) != n)
23             {
24                 fprintf(stderr, "%s: Input error (%s)\n", __FILE__, strerror(errno));
25                 exit(-3);
26             };
27         };
28     }
29     close(readfd);
30     if (unlink(FIFO_NAME) < 0)
31     {
32         fprintf(stderr, "%s: Unable to delete FIFO (%s)\n", __FILE__, strerror(errno));
33         exit(-4);
34     }
35     exit(0);
36 }

```

Рис.3

- Я написала файл common.h (рис.4)

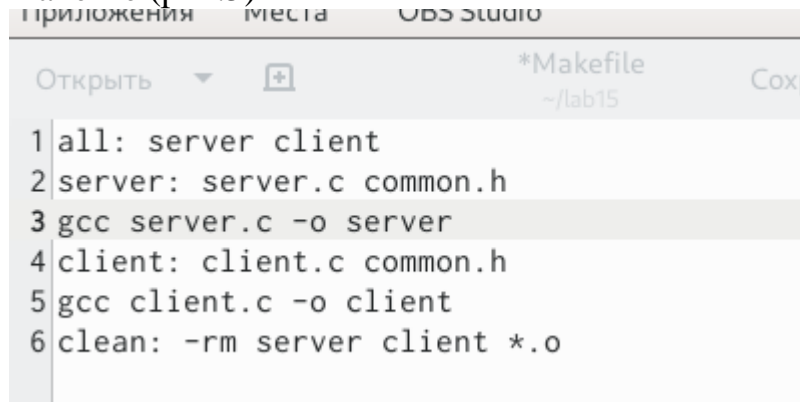
```

Открыть  + common.h
~/lab15
1 #ifndef __COMMON_H__
2 #define __COMMON_H__
3 #include <stdio.h>
4 #include <stdlib.h>
5 #include <string.h>
6 #include <errno.h>
7 #include <sys/types.h>
8 #include <sys/stat.h>
9 #include <fcntl.h>
0 #define FIFO_NAME "/tmp/fifo"
1 #define MAX_BUFF 80
2 #endif /* __COMMON_H__ */

```

Рис.4

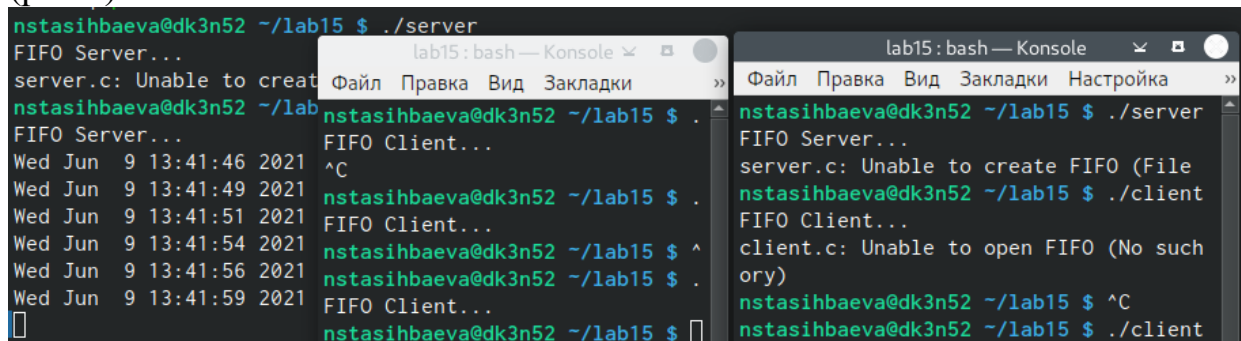
- Я написала Makefile (рис.5)



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

Рис.5

- В результате мы можем видеть, что открыты два клиента на одном сервере (рис.6)



```
nstasihbaeva@dk3n52 ~/lab15 $ ./server
FIFO Server...
server.c: Unable to create FIFO (File exists)
nstasihbaeva@dk3n52 ~/lab15 $ ./client
FIFO Client...
^C
nstasihbaeva@dk3n52 ~/lab15 $ ./server
FIFO Server...
server.c: Unable to create FIFO (File exists)
nstasihbaeva@dk3n52 ~/lab15 $ ./client
FIFO Client...
client.c: Unable to open FIFO (No such file or directory)
nstasihbaeva@dk3n52 ~/lab15 $ ^C
nstasihbaeva@dk3n52 ~/lab15 $ ./client
```

Рис.6

Вывод:

Я приобрела практические навыки работы с именованными каналами.

Ответы на контрольные вопросы:

1. В чем ключевое отличие именованных каналов от неименованных?

Именованные каналы отличаются от неименованных наличием идентификатора канала, который представлен как специальный файл (соответственно имя именованного канала — это имя файла).

2. Возможно ли создание неименованного канала из командной строки?

Для создания неименованного канала используется системный вызов `pipe`. Массив из двух целых чисел является выходным параметром этого системного вызова.

3. Возможно ли создание именованного канала из командной строки?

Да

4. Опишите функцию языка C, создающую 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. Опишите функцию языка C, создающую 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. Что будет в случае прочтения из `fifo` меньшего числа байтов, чем находится в канале? Большого числа байтов?

При чтении меньшего числа байтов, возвращается требуемое число байтов, остаток сохраняется для следующих чтений.

При чтении большего числа байтов, возвращается доступное число байтов

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

При записи большего числа байтов, 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, гарантированно атомарно. Это означает, что в случае, когда несколько процессов одновременно записывают в канал, данные будут записываться отдельно и не будут "повреждать" структуру файла.

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

Могут

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)?

Функция записывает `length` байтов из буфера `buffer` в файл, определенный дескриптором файла `fd`. Эта операция двоичная без буферизации.

Реализуется как непосредственный вызов `DOS`. С помощью функции `write` мы посылаем сигнал клиенту или серверу

10. Опишите функцию `strerror`. Строковая функция `strerror` - функция языков `C/C++`, транслирующая код ошибки, который обычно хранится в глобальной переменной `errno`, в сообщение об ошибке, понятном человеку.