

РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

Факультет физико-математических и естественных наук

Кафедра прикладной информатики и теории вероятностей

ОТЧЕТ

ПО ЛАБОРАТОРНОЙ РАБОТЕ №14

дисциплина: Операционные системы

Студент: Кармацкий Никита Сергеевич

Группа: НФИбд-01-21

Москва

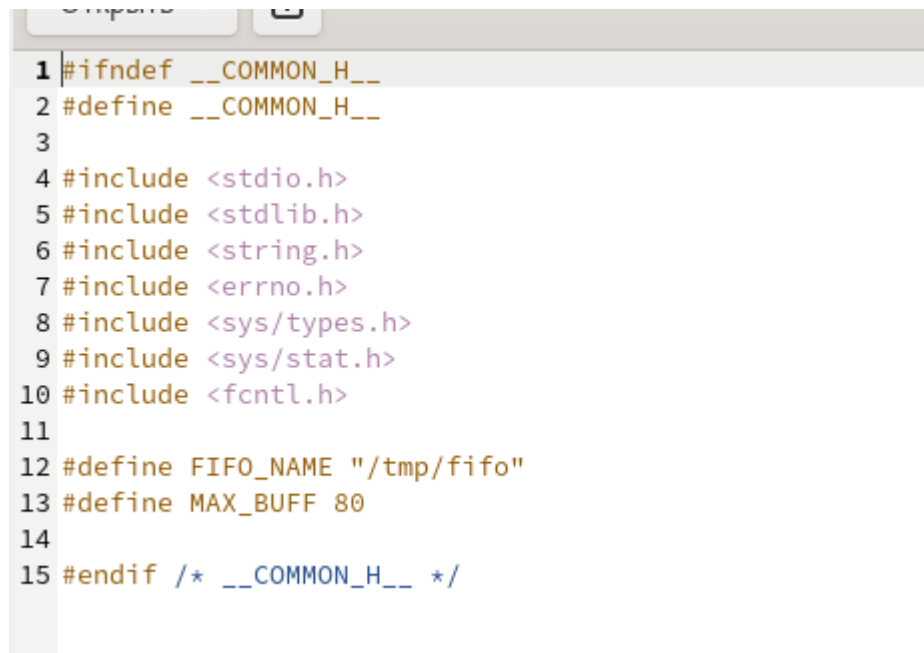
2022 г.

Цель работы:

Приобретение практических навыков работы с именованными каналами.

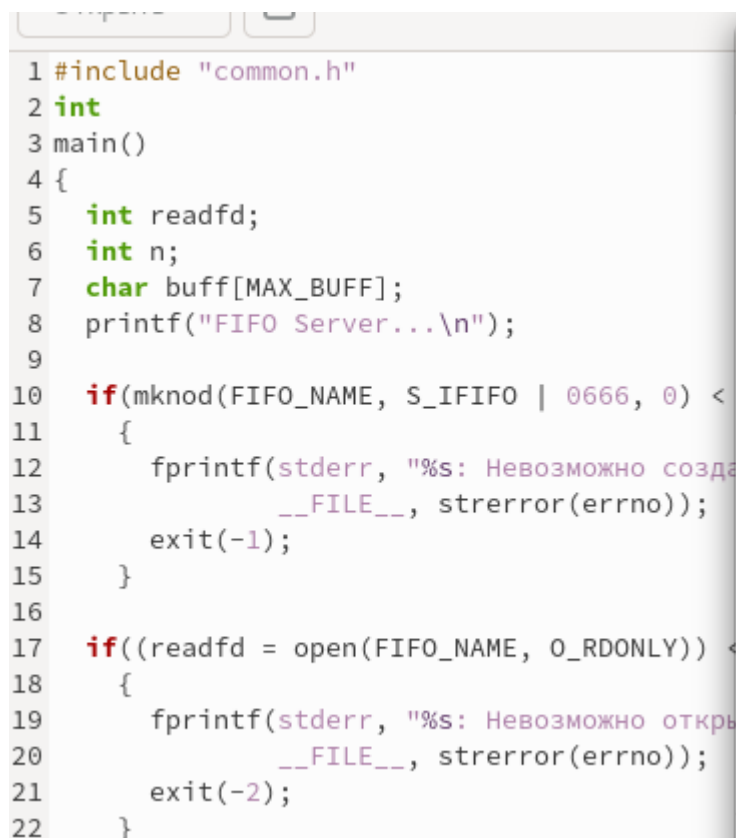
Основные этапы выполнения работы

1. Создали файлы common.h, server.c, client.c, client2.c. Скопировал основной код из теоретической части лабораторной работы и немного подкорректировал его



```
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
12 #define FIFO_NAME "/tmp/fifo"
13 #define MAX_BUFF 80
14
15 #endif /* __COMMON_H__ */
```

Рис.1 common.h



```
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) <
11        {
12            fprintf(stderr, "%s: Невозможно созда
13                __FILE__, strerror(errno));
14            exit(-1);
15        }
16
17    if((readfd = open(FIFO_NAME, O_RDONLY)) <
18        {
19            fprintf(stderr, "%s: Невозможно откр
20                __FILE__, strerror(errno));
21            exit(-2);
22        }
```

Рис.2 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     }

```

Рис.3 client.c

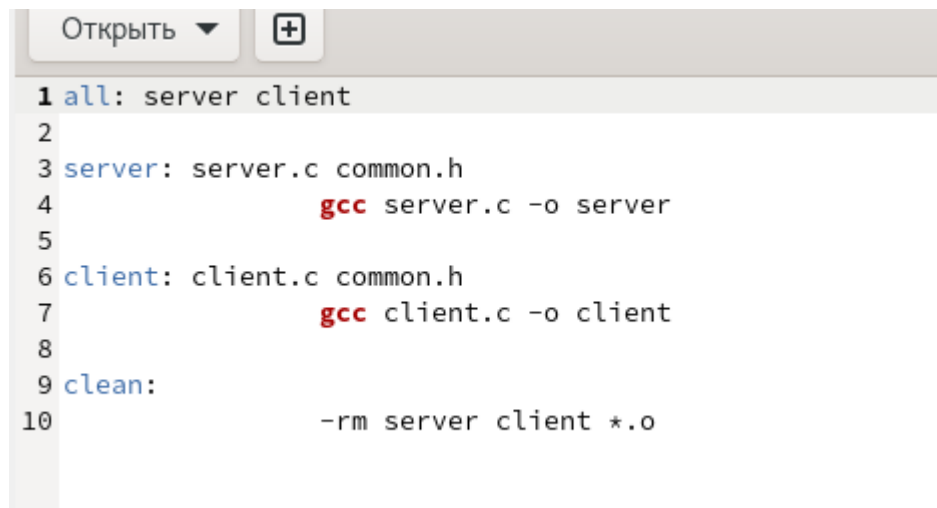
```

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);

```

Рис.4 client2.c


2. Создали makefile



```
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
9 clean:
10        -rm server client *.o
```

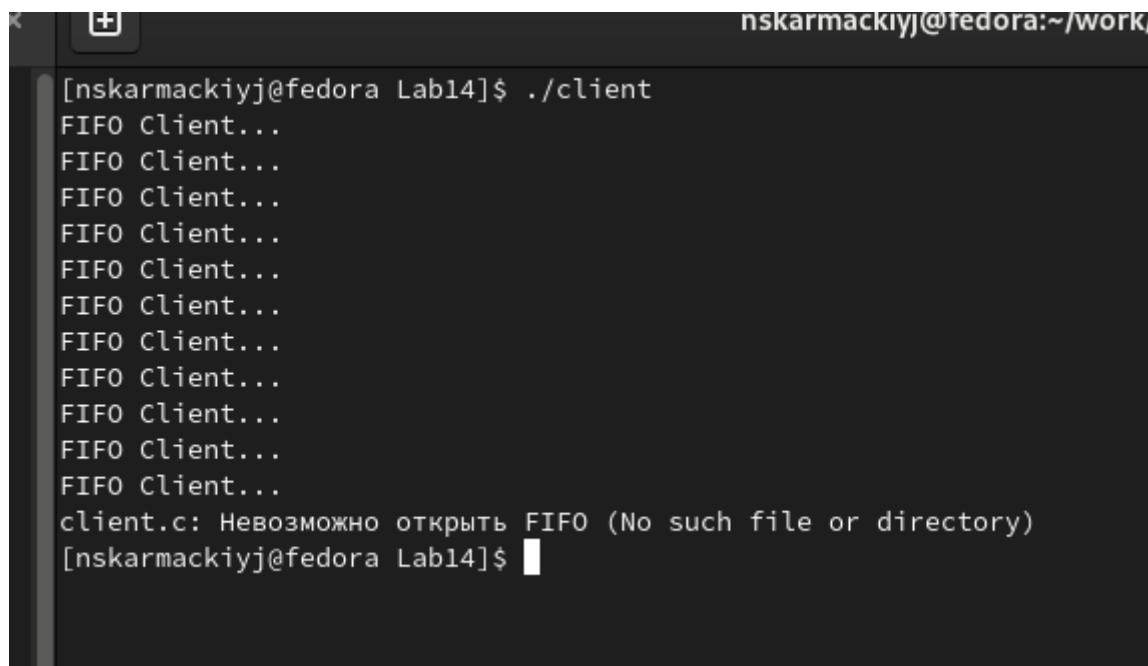
Рис.5 makefile

3. Запустили makefile. Затем запустили server и запустили client в отдельном окне терминала



```
[nskarmackiyj@fedora Lab14]$ ./server
FIFO Server...
Hello Server!!!
Hello Server!!!
Hello Server!!!
Hello Server!!!
Hello Server!!!
Hello Server!!!
Hello Server!!!
Hello Server!!!
Hello Server!!!
Hello Server!!!
server timeout, 30 - seconds passed
[nskarmackiyj@fedora Lab14]$
```

Рис.6 Работа командного файла файла server



```
nskarmackiyj@fedora:~/work/
[nskarmackiyj@fedora Lab14]$ ./client
FIFO Client...
FIFO Client...
FIFO Client...
FIFO Client...
FIFO Client...
FIFO Client...
FIFO Client...
FIFO Client...
FIFO Client...
FIFO Client...
FIFO Client...
FIFO Client...
client.c: Невозможно открыть FIFO (No such file or directory)
[nskarmackiyj@fedora Lab14]$
```

Рис.7 Работа командного файла файла client

Вывод

Мы научились пользоваться именованными каналами.

Ответы на контрольные вопросы

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.