

РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

Факультет физико-математических и естественных наук

Кафедра прикладной информатики и теории вероятностей

ОТЧЕТ

ПО ЛАБОРАТОРНОЙ РАБОТЕ № 15

дисциплина: Операционные системы

Студент: Понкратова Христина Анатольевна

Группа: НПМбд-02-20

МОСКВА

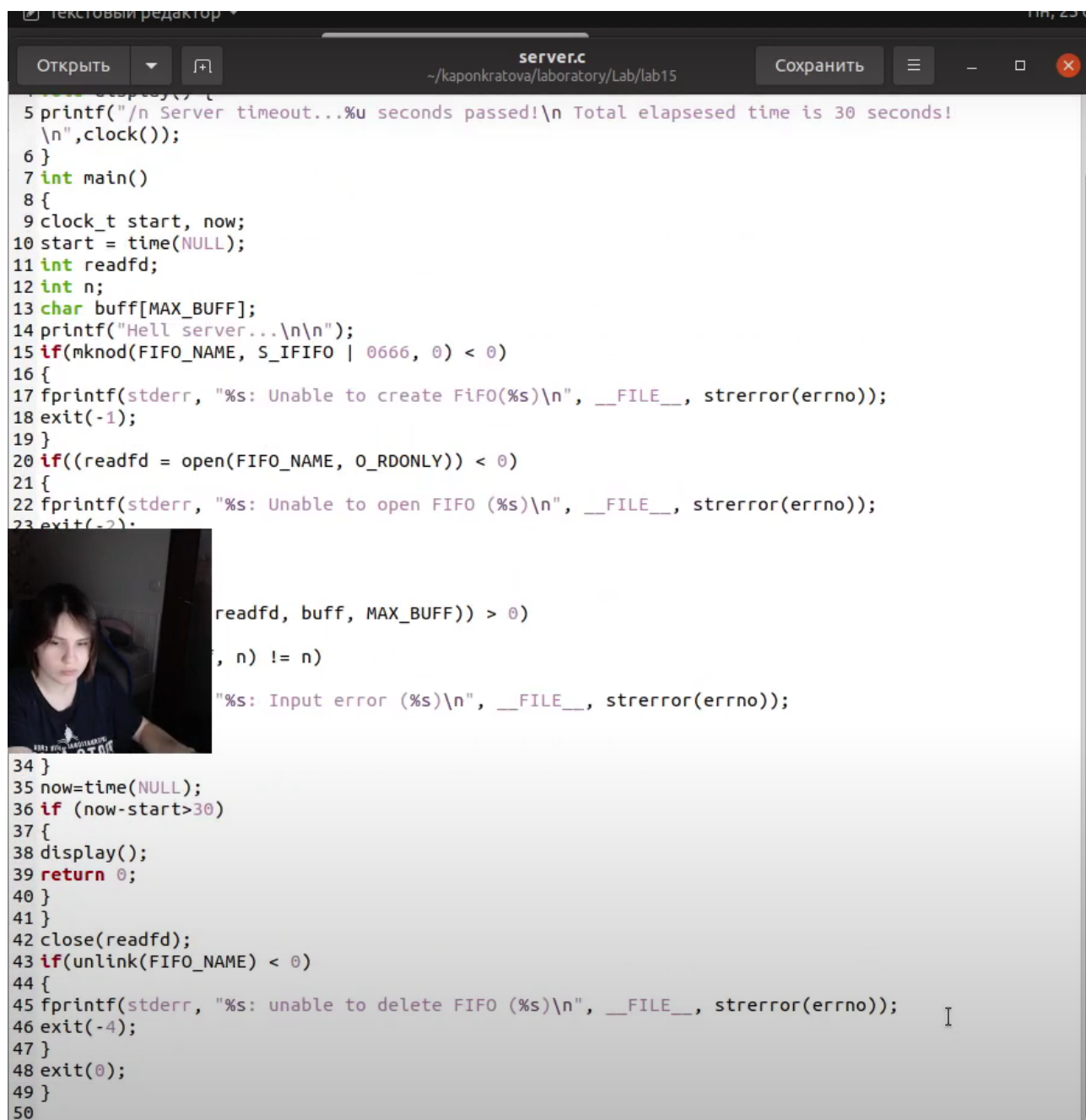
2021 г.

Цель работы:

Приобретение практических навыков работы с сокетами.

Ход работы:

1. Изучили приведенные в тексте работы программы server.c и client.c, разобрались в них. Скомпилировали, исправили ошибки. Запустили.



```
server.c
~/kaponkratova/laboratory/Lab/lab15
Сохранить

5 printf("/n Server timeout...%u seconds passed!\n Total elapsedes time is 30 seconds!
  \n",clock());
6 }
7 int main()
8 {
9 clock_t start, now;
10 start = time(NULL);
11 int readfd;
12 int n;
13 char buff[MAX_BUFF];
14 printf("Hell server...\n\n");
15 if(mknod(FIFO_NAME, S_IFIFO | 0666, 0) < 0)
16 {
17 fprintf(stderr, "%s: Unable to create FiFO(%s)\n", __FILE__, strerror(errno));
18 exit(-1);
19 }
20 if((readfd = open(FIFO_NAME, O_RDONLY)) < 0)
21 {
22 fprintf(stderr, "%s: Unable to open FIFO (%s)\n", __FILE__, strerror(errno));
23 exit(-2);
24 }
25 while((n = read(readfd, buff, MAX_BUFF)) > 0)
26 {
27 if(n != n)
28 {
29 fprintf(stderr, "%s: Input error (%s)\n", __FILE__, strerror(errno));
30 }
31 }
32 }
33 now=time(NULL);
34 if (now-start>30)
35 {
36 display();
37 return 0;
38 }
39 close(readfd);
40 if(unlink(FIFO_NAME) < 0)
41 {
42 fprintf(stderr, "%s: unable to delete FIFO (%s)\n", __FILE__, strerror(errno));
43 exit(-4);
44 }
45 exit(0);
46 }
47 }
48 }
49 }
50 }
```

И

Открыть

server.c

Сохранить

~/kaponkratova/laboratory/Lab/lab15

```
1 //////////////////////////////////////////////////
2 // server.c
3 #include "common.h"
4 void display() {
5     printf("/n Server timeout...%u seconds passed!\n Total elapsedesed time is 30 seconds!
6     \n",clock());
7 }
8 int main()
9 {
10     clock_t start, now;
11     start = time(NULL);
12     int readfd;
13     int n;
14     char buff[MAX_BUFF];
15     printf("Hell server...\n\n");
16     if(mknod(FIFO_NAME, S_IFIFO | 0666, 0) < 0)
17     {
18         fprintf(stderr, "%s: Unable to create FIFO(%s)\n", __FILE__, strerror(errno));
19         exit(-1);
20     }
21     if(open(FIFO_NAME, O_RDONLY)) < 0)
22     {
23         fprintf(stderr, "%s: Unable to open FIFO (%s)\n", __FILE__, strerror(errno));
24         exit(-1);
25     }
26     readfd = open(FIFO_NAME, O_RDONLY);
27     if(readfd, buff, MAX_BUFF) > 0)
28     {
29         if(read(readfd, buff, MAX_BUFF) != n)
30         {
31             fprintf(stderr, "%s: Input error (%s)\n", __FILE__, strerror(errno));
32         }
33         sleep(5);
34     }
35     now=time(NULL);
36     if (now-start>30)
37     {
38         display();
39     }
40 }
```

Загрузка файла «/home/khristina/kaponkratova/laboratory... С Ширина табуляции: 8 Стр 1, Стлб 1 ВСТ

Открыть

common.h

~/kaponkratova/laboratory/Lab/lab15

```
1 //////////////////////////////////////////////////
2 // common.h
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 #define FIFO_NAME "/tmp/fifo"
13 #define MAX_BUFF 80
14 #endif /* __COMMON_H__ */
15
```



The screenshot shows a code editor window with the title 'makefile' and a path '~/\kaponkratova/laboratory/Lab/lab15'. The editor contains a Makefile script with the following content:

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

2. Скомпилировали

3. Запустили несколько (3) клиентов. При работе с третьим клиентом вышло сообщение об ошибке.

Цель работы

Приобретение практических навыков работы с именованными каналами.

Выполнение лабораторной работы

Пишем и редактируем программы на С, так чтобы на одном сервере можно было запускать сначала один клиент. Далее напишем и отредактируем программы, так чтобы на одном сервере можно было запускать больше серверов, чем один, интервал между клиентами будет 5 секунд, сервер завершится через 30 секунд. Мы имеем 4 файла (программы) это заголовочный файл (common.h) клиент (client.c), сервер (server.c), и Makefile.

```
////////////////////////////////////
// common.h
#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>
#define FIFO_NAME "/tmp/fifo"
#define MAX_BUFF 80
#endif /* __COMMON_H__ */

////////////////////////////////////
// client.c
#include "common.h"
#define MESSAGE "Hello Server!!! \n"
int main ()
{
    int writefd;
    int msglen;
    printf("FIFO Client... \n");
    if ((writefd = open(FIFO_NAME, O_WRONLY)) < 0)
    {
        fprintf(stderr, "%s: Невозможно открыть FIFO (%s)\n",
            __FILE__, strerror(errno));
        exit(-1);
    }
    msglen = strlen(MESSAGE);
    if (write(writefd, MESSAGE, msglen) != msglen)
    {
        fprintf(stderr, "%s: Ошибка записи в FIFO (%s)\n",
            __FILE__, strerror(errno));
        exit(-2);
    }
    close (writefd);
    exit(0);
}

////////////////////////////////////
// server.c
```

```

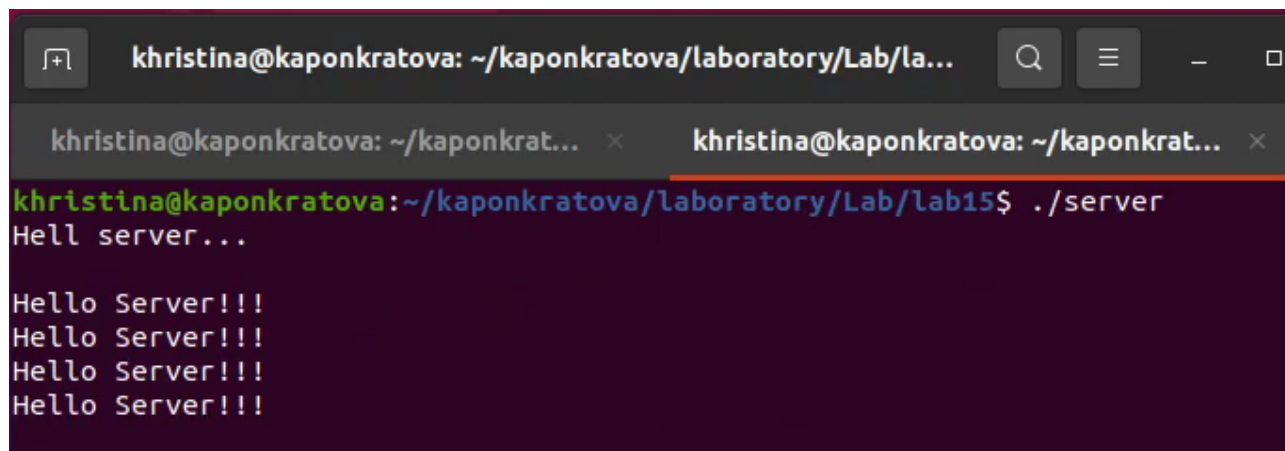
#include "common.h"
void display() {
    printf("/n Server timeout...%u seconds passed!\n Total elapsed time is 30
seconds!\n", clock());
}
int main()
{
    clock_t start, now;
    start = time(NULL);
    int readfd;
    int n;
    char buff[MAX_BUFF];
    printf("Hel I server...\n\n");
    if(mknod(FIFO_NAME, S_IFIFO | 0666, 0) < 0)
    {
        fprintf(stderr, "%s: Unable to create FiFO(%s)\n", __FILE__, strerror(errno));
        exit(-1);
    }
    if((readfd = open(FIFO_NAME, O_RDONLY)) < 0)
    {
        fprintf(stderr, "%s: Unable to open FiFO (%s)\n", __FILE__, strerror(errno));
        exit(-2);
    }
    for(;;)
    {
        while((n = read(readfd, buff, MAX_BUFF)) > 0)
        {
            if(write(1, buff, n) != n)
            {
                fprintf(stderr, "%s: Input error (%s)\n", __FILE__, strerror(errno));
            }
            sleep(5);
        }
        now=time(NULL);
        if (now-start>30)
        {
            display();
            return 0;
        }
    }
    close(readfd);
    if(unlink(FIFO_NAME) < 0)
    {
        fprintf(stderr, "%s: unable to delete FiFO (%s)\n", __FILE__, strerror(errno));
        exit(-4);
    }
    exit(0);
}

////////////////////////////////////
// makefile
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 *.o
```

Запуск пректа



```
khristina@kaponkratova: ~/kaponkratova/laboratory/Lab/lab15$ ./server
Hell server...

Hello Server!!!
Hello Server!!!
Hello Server!!!
Hello Server!!!
```

Вывод

В данной работе мы приобрели практические навыки работы с именованными каналами по типу клиент-сервер.

Контрольные вопросы

1. В чем ключевое отличие именованных каналов от неименованных? Ответ: У именованных каналов есть идентификатор канала, а у неименованных его нет.
2. Возможно ли создание неименованного канала из командной строки? Ответ: Возможно создание неименованного канала из командной строки, но только с созданием временного канала с индикатором.
3. Возможно ли создание именованного канала из командной строки? Ответ: Да. При помощи `mknod`.
4. Опишите функцию языка C, создающую неименованный канал. Ответ:

```
#include int fd[2];
pipe(fd);
/* возвращает 0 в случае успешного завершения, -1 - в случае ошибки;*/
Это значит, что функция возвращает два файловых дескриптора: fd[0] и fd[1], при этом первый открыт для чтения, а второй – для записи.
```

5. Опишите функцию языка C, создающую именованный канал. Ответ:

```
#include <sys/types.h>
#include <sys/stat.h>
int mkfifo(const char *pathname, mode_t mode);
```

6. . Что будет в случае прочтения из fifo меньшего числа байтов, чем находится в канале? Большого числа байтов? Ответ: При чтении меньшего числа байтов, чем находится в канале или FIFO, возвращается требуемое число байтов, остаток сохраняется для последующих чтений. При чтении большего числа байтов, чем находится в канале или FIFO, возвращается доступное число байтов. Процесс, читающий из канала, должен соответствующим образом обработать ситуацию, когда прочитано меньше, чем требуется в программе.

7. Аналогично, что будет в случае записи в `fifo` меньшего числа байтов, чем позволяет буфер? Большого числа байтов? Ответ: Запись числа байтов, меньшего числа битов у канала или `FIFO`, 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 занятой нами до этого памяти.
8. Могут ли два и более процессов читать или записывать в канал? Ответ: Да. Если у `buff` достаточное количество памяти.
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)? Ответ: Функция записывает `length` памяти из буфера `buffer` в файл, определенный дескриптором файла `fd`. Эта операция чисто 'двоичная' и без буферизации. При единице возвращает действительное число байтов. При -1 сообщение об ошибке.
10. Опишите функцию `strerror` Ответ: Интерпретирует номер ошибки, передаваемый в функцию в качестве аргумента — `errno`, в понятное для человека текстовое сообщение (строку). Откуда берутся эти ошибки? Ошибки эти возникают при вызове функций стандартных Си-библиотек. То есть хорошим тоном программирования будет — использование этой функции в паре с другой, и если возникнет ошибка, то пользователь или программист поймет, как исправить ошибку, прочитав сообщение функции `strerror`. Возвращенный указатель ссылается на статическую строку с ошибкой, которая не должна быть изменена программой. Дальнейшие вызовы функции `strerror` перезапишут содержание этой строки. Интерпретированные сообщения об ошибках могут различаться, это зависит от платформы и компилятора.