

Лабораторная работа №14

Именованные каналы

Латыпова Диана. НФИбд-02-21

Содержание

1	Цель работы	5
2	Задание	6
3	Выполнение лабораторной работы	7
4	Контрольные вопросы	17
5	Выводы	20

Список иллюстраций

3.1	Код файла client.c	9
3.2	Код файла client2.c	11
3.3	Код файла common.h	12
3.4	Код файла makefile	13
3.5	Код файла server.c	15
3.6	Команда make	16
3.7	Запуск командных файлов	16

Список таблиц

1 Цель работы

Приобретение практических навыков работы с именованными каналами.

2 Задание

Изучите приведённые в тексте программы `server.c` и `client.c`. Взяв данные примеры за образец, напишите 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

Для начала я создала 5 файлов: server.c, client.c, client2.c, common.h, makefile. После чего, написала коды для каждого файла, используя примеры в работе.

- Листинг файла client.c(рис. 3.1):

```
#include "common.h"

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

int
main()
{
    int msg, len, i;
    long int t;

    for(i=0; i<20; i++)
    {
        sleep(3);
        t=time(NULL);
        printf("FIFO Client...\n");

        if((msg = open(FIFO_NAME, O_WRONLY)) < 0)
        {
```

```

        fprintf(stderr,"%s: Невозможно открыть FIFO (%s)\n",
        __FILE__, strerror(errno));
        exit(-1);
    }

    len = strlen(MESSAGE);

    if(write(msg, MESSAGE, len) != len)
    {
        fprintf(stderr,"%s: Ошибка записи в FIFO (%s)\n",
        __FILE__, strerror(errno));
        exit(-2);
    }
    close(msg);
}

exit(0);
}

```



```

home > dlatihpova > work > os > 14lab > C client.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
24         len = strlen(MESSAGE);
25
26         if(write(msg, MESSAGE, len) != len)
27         {
28             fprintf(stderr,"%s: Ошибка записи в FIFO (%s)\n",
29                 __FILE__, strerror(errno));

```

Рис. 3.1: Код файла client.c

- Листинг файла client2.c(рис. 3.2):

```

#include "common.h"

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

int
main()
{
    int writefd, msglen, count;
    long long int t;
    char message[10];

```

```

for(count=0; count<-5; ++count)
{
    sleep(5);
    t=(long long int) time(0);
    sprintf(message, "%lli", t);
    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);
}

```

```

home > dlatihpova > work > os > 14lab > C client2.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);
22         }
23
24         msglen = strlen(MESSAGE);
25         if(write(writefd, MESSAGE, msglen) != msglen)
26         {
27             fprintf(stderr,"%s: Ошибка записи в FIFO (%s)\n",
28                     __FILE__, strerror(errno));
29             exit(-2);
30         }

```

Рис. 3.2: Код файла client2.c

- Листинг файла common.h(рис. 3.3):

```

#ifndef __COMMON_H__
#define __COMMON_H__

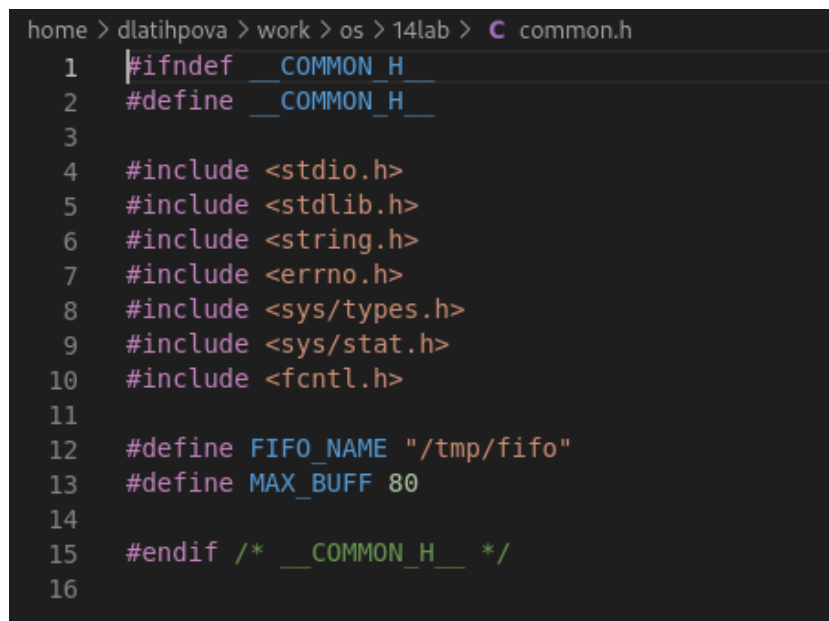
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <errno.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>

```

```
#define FIFO_NAME "/tmp/fifo"

#define MAX_BUFF 80

#endif /* __COMMON_H__ */
```

A screenshot of a terminal window with a dark background. The terminal shows the file path 'home > dlatihpova > work > os > 14lab > C common.h' at the top. Below the path, the code for 'common.h' is displayed, line by line, with line numbers 1 through 16 on the left. The code includes several standard headers and defines constants for a FIFO name and buffer size.

```
home > dlatihpova > work > os > 14lab > C common.h
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__ */
16
```

Рис. 3.3: Код файла common.h

- Листинг файла makefile(рис. 3.4):

```
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
```

```
home > dlatihpova > work > os > 14lab > M 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. 3.4: Код файла makefile

- Листинг файла server.c(рис. 3.5):

```
#include "common.h"

int
main()
{
    int readfd;
    int n;
    char buff[MAX_BUFF];
    printf("FIFO Server...\n");

    if(mknod(FIFO_NAME, S_IFIFO | 0666, 0) < 0)
    {
        fprintf(stderr, "%s: Невозможно создать FIFO (%s)\n",
            __FILE__, strerror(errno));
        exit(-1);
    }
}
```

```

    }

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

    clock_t now=time(NULL), start=time(NULL);
    while(now-start<30)
    {
        while((n = read(readfd, buff, MAX_BUFF)) > 0)
        {
            if(write(1, buff, n) != n)
            {
                fprintf(stderr, "%s: Ошибка вывода (%s)\n",
                    __FILE__, strerror(errno));
            }
        }
        now=time(NULL);
    }

    printf("server timeout, %li - seconds passed\n", (now-start));
    close(readfd);

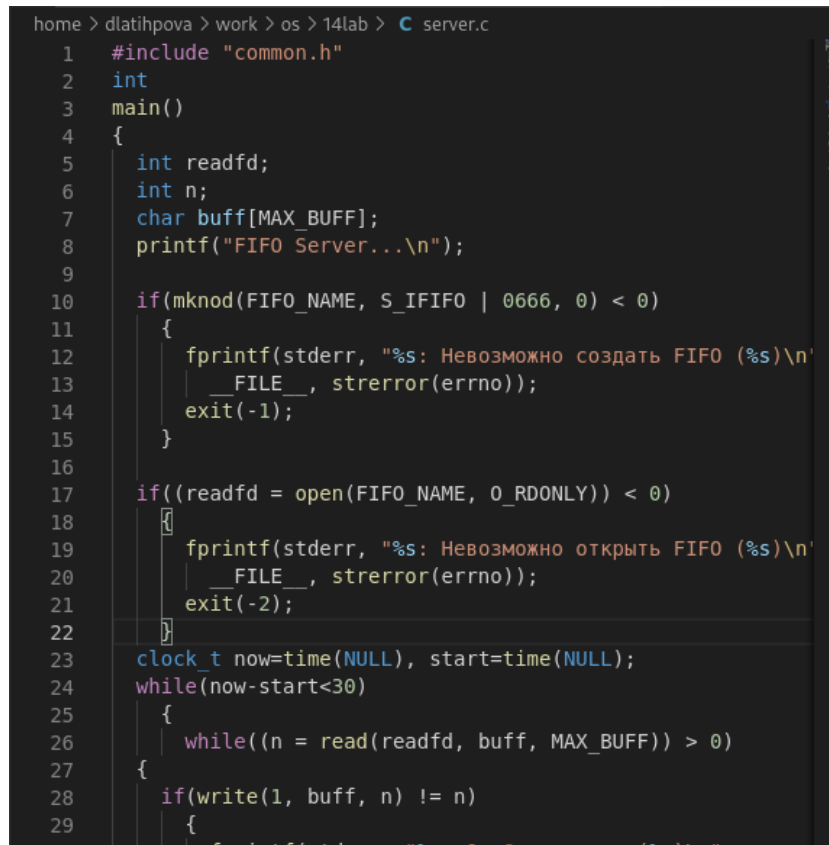
    if(unlink(FIFO_NAME) < 0)
    {
        fprintf(stderr, "%s: Невозможно удалить FIFO (%s)\n",
            __FILE__, strerror(errno));
        exit(-4);
    }

```

```

    }
    exit(0);
}

```



```

home > dlatihpova > work > os > 14lab > C server.c
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) < 0)
11     {
12         fprintf(stderr, "%s: Невозможно создать FIFO (%s)\n",
13             __FILE__, strerror(errno));
14         exit(-1);
15     }
16
17     if((readfd = open(FIFO_NAME, O_RDONLY)) < 0)
18     {
19         fprintf(stderr, "%s: Невозможно открыть FIFO (%s)\n",
20             __FILE__, strerror(errno));
21         exit(-2);
22     }
23     clock_t now=time(NULL), start=time(NULL);
24     while(now-start<30)
25     {
26         while((n = read(readfd, buff, MAX_BUFF)) > 0)
27         {
28             if(write(1, buff, n) != n)
29             {
30                 fprintf(stderr, "%s: Ошибка вывода (%s)\n",

```

Рис. 3.5: Код файла server.c

Запустила 2 терминала в папке с данными файлами. Выполнила команду make, создалось 2 командных файла(рис. 3.6):

make

4 Контрольные вопросы

1. В чем ключевое отличие именованных каналов от неименованных?

Именованные каналы, в отличие от неименованных, могут использоваться неродственными процессами. Они дают вам, по сути, те же возможности, что и неименованные каналы, но с некоторыми преимуществами, присущими обычным файлам. Именованные каналы используют специальную запись в директории для управления правами доступа.

2. Возможно ли создание неименованного канала из командной строки?

Создание неименованного канала из командной строки возможно командой `pipe`.

3. Возможно ли создание именованного канала из командной строки?

Создание именованного канала из командной строки возможно с помощью `mkfifo`.

4. Опишите функцию языка C, создающую неименованный канал.

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. Опишите функцию языка C, создающую именованный канал.

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)`;

6-7. 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. 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).

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

Два и более процессов могут читать и записывать в канал.

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 `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`.

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.

5 Выводы

Я приобрела практические навыки работы с именованными каналами.