

Отчёт по лабораторной работе

Именованные каналы

Грузинова Елизавета Константиновна

Содержание

1	Цель работы	4
2	Задание	5
3	Теоретическое введение	6
4	Выполнение лабораторной работы	7
5	Выводы	14
6	Контрольные вопросы	15
	Список литературы	18

Список иллюстраций

4.1	Текст Makefile	7
4.2	Текст common.h	8
4.3	Текст будущей программы client	9
4.4	Работа программы client при запущенном сервере	9
4.5	Текст будущей программы client2	10
4.6	Работа программы client2 при запущенном сервере	11
4.7	Изменения в файле server.c(1)	12
4.8	Изменения в файле server.c(1)	12
4.9	Работа программы server	13

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?

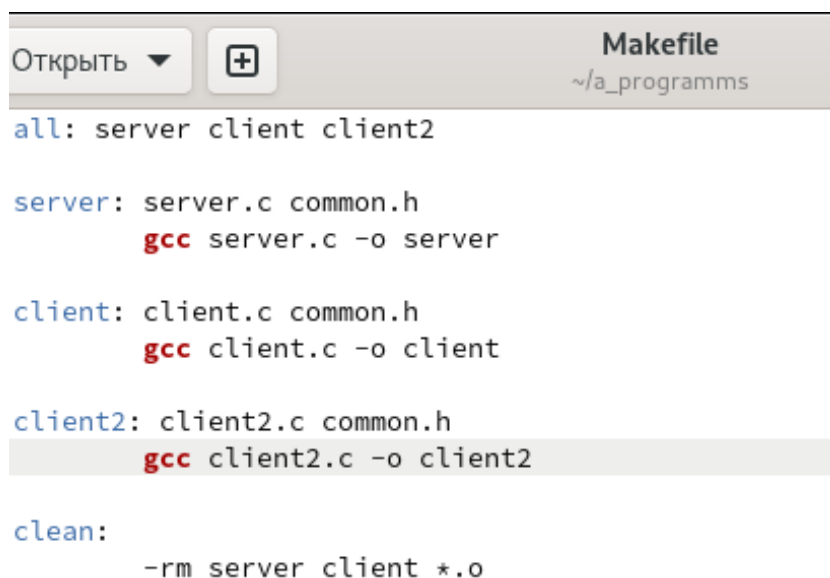
3 Теоретическое введение

Для передачи данных между неродственными процессами можно использовать механизм именованных каналов (named pipes). Данные передаются по принципу FIFO (First In First Out) (первым записан — первым прочитан), поэтому они называются также FIFO pipes или просто FIFO. Именованные каналы отличаются от неименованных наличием идентификатора канала, который представлен как специальный файл (соответственно имя именованного канала — это имя файла). Поскольку файл находится на локальной файловой системе, данное IPC используется внутри одной системы. [1]

4 Выполнение лабораторной работы

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). Также привести необходимые для работы тексты будущих командных файлов. (рис. 4.1, 4.2, 4.3, 4.4)



The image shows a screenshot of a text editor window titled "Makefile" with the path "~/a_programms". The window contains the following text:

```
Открыть ▼  +  Makefile
~/a_programms

all: server client client2

server: server.c common.h
    gcc server.c -o server

client: client.c common.h
    gcc client.c -o client

client2: client2.c common.h
    gcc client2.c -o client2

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

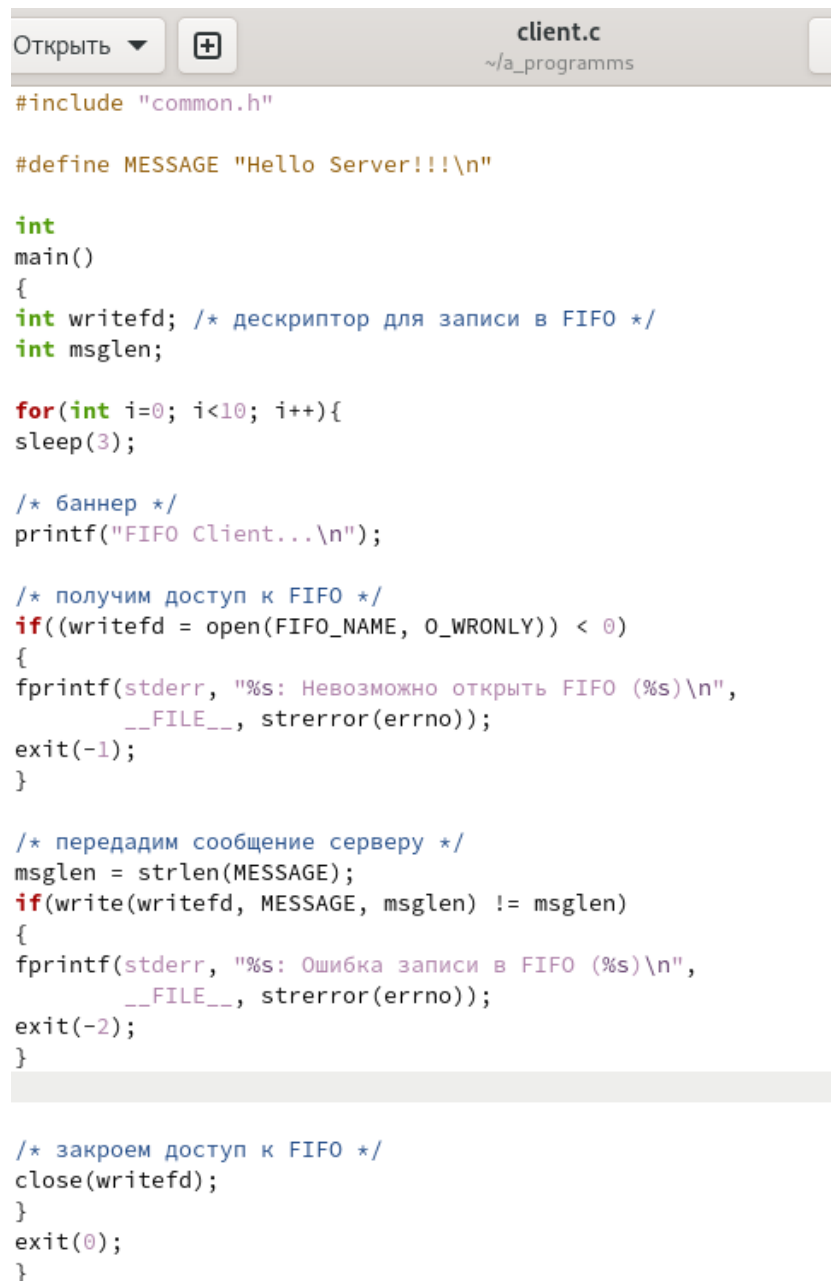
Рис. 4.1: Текст Makefile

```
#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>
#include <unistd.h>
#include <time.h>
#define FIFO_NAME "/tmp/fifo"
#define MAX_BUFF 80

#endif /* __COMMON_H__ */
```

Рис. 4.2: Текст common.h



```

Открыть + client.c
~/_a_programms

#include "common.h"

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

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

    for(int i=0; i<10; i++){
        sleep(3);

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

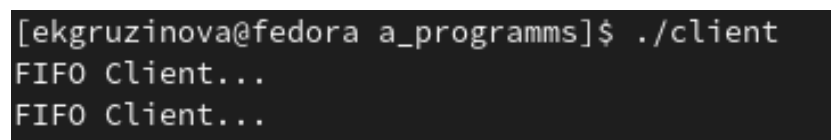
        /* получим доступ к FIFO */
        if((writefd = open(FIFO_NAME, O_WRONLY)) < 0)
        {
            fprintf(stderr, "%s: Невозможно открыть FIFO (%s)\n",
                __FILE__, strerror(errno));
            exit(-1);
        }

        /* передадим сообщение серверу */
        msglen = strlen(MESSAGE);
        if(write(writefd, MESSAGE, msglen) != msglen)
        {
            fprintf(stderr, "%s: Ошибка записи в FIFO (%s)\n",
                __FILE__, strerror(errno));
            exit(-2);
        }

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

```

Рис. 4.3: Текст будущей программы client



```

[ekgruzinova@fedora _a_programms]$ ./client
FIFO Client...
FIFO Client...

```

Рис. 4.4: Работа программы client при запущенном сервере

2. Клиенты передают текущее время с некоторой периодичностью (например,

раз в пять секунд). Используйте функцию `sleep()` для приостановки работы клиента. (рис. 4.5, 4.6)

```
#include "common.h"
#include <time.h>
#define MESSAGE "Hello Server!!!\n"

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

    for(int i=0; i<15; i++){
        ttime=time(NULL);
        printf(ctime(&ttime));

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

        /* получим доступ к FIFO */
        if((writefd = open(FIFO_NAME, O_WRONLY)) < 0)
        {
            fprintf(stderr, "%s: Невозможно открыть FIFO (%s)\n",
                __FILE__, strerror(errno));
            exit(-1);
        }

        /* передадим сообщение серверу */
        msglen = strlen(MESSAGE);
        if(write(writefd, MESSAGE, msglen) != msglen)
        {
            fprintf(stderr, "%s: Ошибка записи в FIFO (%s)\n",
                __FILE__, strerror(errno));
            exit(-2);
        }
        sleep(5);
    }
    /* закроем доступ к FIFO */
    close(writefd);

    exit(0);
}
```

Рис. 4.5: Текст будущей программы client2

```
[ekgruzinova@fedora a_programms]$ ./client2
Sat Jun  4 21:14:10 2022
FIFO Client...
Sat Jun  4 21:14:15 2022
FIFO Client...
Sat Jun  4 21:14:20 2022
FIFO Client...
Sat Jun  4 21:14:25 2022
FIFO Client...
Sat Jun  4 21:14:30 2022
```

Рис. 4.6: Работа программы client2 при запущенном сервере

3. Сервер работает не бесконечно, а прекращает работу через некоторое время (например, 30 сек). Используйте функцию `clock()` для определения времени работы сервера. (рис. 4.7, 4.8, 4.9)

```

#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 запись
     */
    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);
    }
    clock_t now=time(NULL), start=time(NULL);
    while(now-start<30)
    /* 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);
        }
    }
    now=time(NULL);
    printf("Время работы сервера завершено спустя %li - секунд\n", (now-start));
}

```

Рис. 4.7: Изменения в файле server.c(1)

```

close(readfd); /* закроем FIFO */

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

```

Рис. 4.8: Изменения в файле server.c(1)

```
tekgruzinova@fedora a_programms]$ ./server
FIFO Server...
Hello Server!!!
Hello Server!!!
Hello Server!!!
Hello Server!!!
Hello Server!!!
Hello Server!!!
Hello Server!!!
Hello Server!!!
Hello Server!!!
Hello Server!!!
Время работы сервера завершено спустя 30 - секунд
```

Рис. 4.9: Работа программы server

Если сервер завершит работу, не закрыв канал, то файл FIFO не исчезнет, вследствие чего его нельзя будет запустить во второй раз, что приведет к недееспособности сервера.

5 Выводы

В процессе выполнения лабораторной работы я приобрела практические навыки работы с именованными каналами.

6 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?

Для создания файла FIFO можно использовать более общую функцию `mknod(2)`, пред- назначенную для создания специальных файлов различных типов (FIFO, сокет, файлы устройств и обычные файлы для хранения данных).
1 `#include <sys/types.h>` 2 `#include <sys/stat.h>` 3 `#include <fcntl.h>` 4 `#include <unistd.h>` 5 6
`int mknod(const char *pathname, mode_t mode, dev_t dev);`

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, чем находится в канале, возвращается требуемое число байтов, остаток сохраняется для последующих чтений. При чтении большего числа байтов, чем находится в канале или FIFO 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, 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. 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

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

В общем случае возможна многонаправленная работа процессов с каналом, т.е. возможна ситуация, когда с одним и тем же каналом взаимодействуют два и более процесса, и каждый из взаимодействующих каналов пишет и читает информацию в канал. Но традиционной схемой организации работы с каналом

является однонаправленная организация, когда канал связывает два, в большинстве случаев, или несколько взаимодействующих процесса, каждый из которых может либо читать, либо писать 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)?

`Write` - Функция записывает `length` байтов из буфера `buffer` в файл, определенный дескриптором файла `fd`. Эта операция чисто 'двоичная' и без буферизации. Реализуется как непосредственный вызов `DOS`. С помощью функции `write` мы посылаем сообщение 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. Ошибки эти возникают при вызове функций стандартных Си-библиотек.

Список литературы

1. Лабораторная работа No 14. Именованные каналы [Электронный ресурс].
URL: https://esystem.rudn.ru/pluginfile.php/1383193/mod_resource/content/3/014-ipc-fifo.pdf.