# Отчет по лабораторной работе 15

**Именованные каналы**

Шалыгин Георгий Эдуардович, НФИбд-02-20

# Содержание

# List of Figures

# 1 Цель работы

Приобретение практических навыков работы с именованными каналами

Объект исследования: система UNIX.

Предмет исследования: работа с именованными каналами в UNIX.

# 2 Техническое обеспечение:

- Характеристики техники: AMD Ryzen 5 3500U 2.1 GHz, 8 GB оперативной памяти, 50 GB свободного места на жёстком диске;
- ОС Windows 10 Home
- Git 2.31.1
- Google Chrome 91.0.4472.19
- VirtualBox 2.0
- CentOS 7

# 3 Условные обозначения и термины:

**Именованный канал** или **именованный конвейер** (англ. *named pipe*) — один из методов межпроцессного взаимодействия, расширение понятия конвейера в Unix и подобных ОС.[1]

**Сокет домена Unix** или **IPC-сокет** (сокет межпроцессного взаимодействия) — конечная точка обмена данными, подобная Интернет-сокету, но не использующая сетевого протокола для взаимодействия (обмена данными)

**Командный язык** - это язык, на котором пользователь взаимодействует с системой в интерактивном режиме.

**Командный интерпретатор**, интерпретатор командной строки - компьютерная программа, часть операционной системы, обеспечивающая базовые возможности управления компьютером посредством интерактивного ввода команд через интерфейс командной строки или последовательного исполнения пакетных командных файлов.[3]

Подробнее в [2] и [3].

# 4 Теоретическое введение:

Одним из видов взаимодействия между процессами в операционных системах является обмен сообщениями. Под сообщением понимается последовательность байтов, передаваемая от одного процесса другому.

В операционных системах типа UNIX есть 3 вида межпроцессорных взаимодействий: общеюниксные (именованные каналы, сигналы), System V Interface Definition (SVID — разделяемая память, очередь сообщений, семафоры) и BSD (сокеты).

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.

Файлы именованных каналов создаются функцией mkfifo(3).

Первый параметр — имя файла, идентифицирующего канал, второй параметр — маска прав доступа к файлу.

После создания файла канала процессы, участвующие в обмене данными, должны открыть этот файл либо для записи, либо для чтения. При закрытии файла сам канал продолжает существовать. Для того чтобы закрыть сам канал, нужно удалить его файл, например с помощью вызова unlink(2).

Рассмотрим работу именованного канала на примере системы клиент–сервер.

Сервер создаёт канал, читает из него текст, посылаемый клиентом, и выводит его на терминал.

Вызов функции mkfifo() создаёт файл канала (с именем, заданным макросом FIFO_NAME):

mkfifo(FIFO_NAME, 0600);

В качестве маски доступа используется восьмеричное значение 0600, разрешающее процессу с аналогичными реквизитами пользователя чтение и запись. Можно также установить права доступа 0666. Открываем созданный файл для чтения:

f = fopen(FIFO_NAME, O_RDONLY);

Ждём сообщение от клиента. Сообщение читаем с помощью функции read() и печатаем на экран. После этого удаляется файл FIFO_NAME и сервер прекращает работу. Клиент открывает FIFO для записи как обычный файл:

f = fopen(FIFO_NAME, O_WRONLY);

Посылаем сообщение серверу с помощью функции write().

Для создания файла FIFO можно использовать более общую функцию mknod(2), предназначенную для создания специальных файлов различных типов (FIFO, сокеты, файлы устройств и обычные файлы для хранения данных).

int mknod(const char *pathname, mode_t mode, dev_t dev);

Тогда, вместо mkfifo(FIFO_NAME, 0600); пишем mknod(FIFO_NAME, S_IFIFO | 0600, 0);

Каналы представляют собой простое и удобное средство передачи данных, которое, однако, подходит не во всех ситуациях. Например, с помощью каналов довольно трудно организовать обмен асинхронными сообщениями между процессами.

# 5 Выполнение лабораторной работы

1. Создадим файлы: common.h, server.c, client.c, client2.c для работы с именованными каналами со следующими критериями:

    - Работает не 1 клиент, а несколько (например, два).
    - Клиенты передают текущее время с некоторой периодичностью (например, раз в пять секунд). Используйте функцию sleep() для приостановки работы клиента.
    - Сервер работает не бесконечно, а прекращает работу через некоторое время (например, 30 сек). Используйте функцию clock() для определения времени работы сервера.

    В случае, если сервер завершит работу, не закрыв канал, в дальнейшем его не удастся создать, т.к. не удалятся временные файлы.

    (рис. 5.1, 5.2, 5.3, 5.4 ).

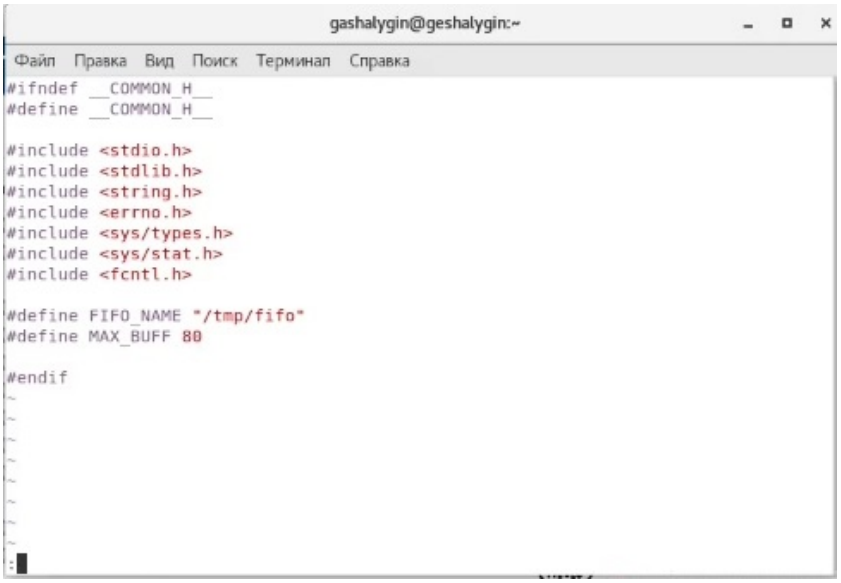


Figure 5.1: Текст commoh.h

```c
#include "common.h"
#define MESSAGE "Hello Server!!!\n"

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

        for (int i=0; i<=5; i++)
        {
                sleep(5);
                t=(long long) time(0);
                sprintf(message, "%lli", t);
                message[9] = '\n';
                printf("FIFO client2\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);
}
```

Figure 5.2: Текст client2.c

```c
#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);
}
~
~
```

Figure 5.3: Текст client.c

```c
#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));                                                exit(-3);
                        }
                }
                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);
}
```

Figure 5.4: Текст client.c

2. Выполним компиляцию программы посредством gcc. (рис. 5.5).

   Запустим программы в разных терминалах и увидим взаимодействие между ними на рис. 5.6.

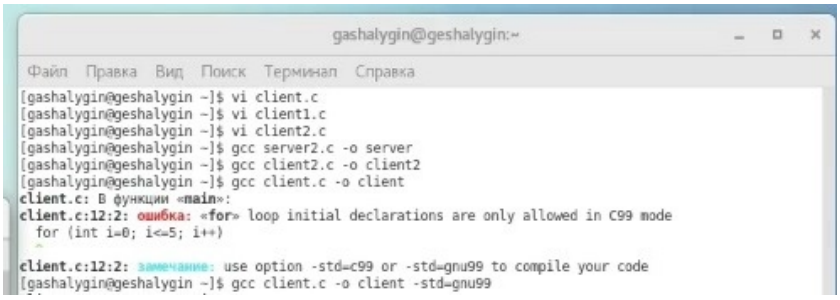


Figure 5.5: Компиляция

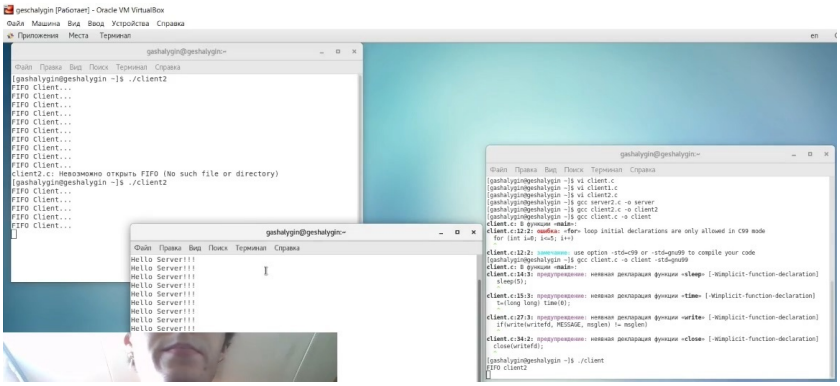


Figure 5.6: Запуск

# 6 Выводы

В процессе работы над лабораторной работы были изучены основы программирования в оболочке ОС UNIX, получен опыт работы с именованными каналами.

# 7 Библиография


1. https://parallel.uran.ru/book/export/html/464
2. https://www.opennet.ru/docs/RUS/linux_parallel/node17.html
3. Д.С. Кулябов, А.В. Королькова / Администрирование локальных систем. Лабораторные работы. — М.: Российский университет дружбы народов, 2017. — 119 с.