

Отчёт по лабораторной работе №15

Именованные каналы

Дисциплина: Операционные системы

Студент: Оразгелдиева Огулнур

Группа: НПИбд-02-20

Студ. номер: 1032205431

2021, Москва

Лабораторная работа №15

Именованные каналы

Цель:

Приобрести практические навыки работы с именованными каналами

Задачи:

1. Ознакомиться с теоретическим материалом
 2. Изучить приведённые в тексте программы `server.c` и `client.c`. Взяв данные примеры за образец, написать аналогичные программы, внося следующие изменения:
 - Работает не 1 клиент, а несколько (например, два).
 - Клиенты передают текущее время с некоторой периодичностью (например, раз в пять секунд). Использовать функцию `sleep()` для приостановки работы клиента.
 - Сервер работает не бесконечно, а прекращает работу через некоторое время (например, 30сек). Использовать функцию `clock()` для определения времени работы сервера.
-

Теоретические сведения [\[1\]](#) [\[2\]](#)

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

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)`.

```
include
include
int mkfifo(const char * pathname, mode_t mode);
```

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

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

```
include
include
include <fcntl.h>
include <unistd.h>
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. [\[1\]](#)

Дополнительно:

Функция `clock()` возвращает количество временных тактов, прошедших с начала запуска программы.

С помощью макроса `CLOCKS_PER_SEC` функция получает количество пройденных тактов за 1 секунду. Таким образом, зная сколько выполняется тактов в секунду, зная время запуска программы можно посчитать время работы всей программы или отдельного её фрагмента, что и делает данная функция.

Возвращаемое значение - число тактов прошедшее с момента запуска программы.

Возвращаемое значение функции `clock` имеет тип данных `clock_t`. `clock_t` способный представлять временные такты, а также поддерживает арифметические операции. [2]

Ход работы

1. Ознакомившись с теоретическим материалом, рассмотрим тексты программ файлов `server.`, `client.c`.
2. Будем пользоваться редактором `vi`. Создадим с помощью этого редактора файл `common.h`. (см. рис. 1)

```
oorazgeldiyeva@oorazgeldiyeva:~  
Файл  Правка  Вид  Поиск  Терминал  Справка  
[oorazgeldiyeva@oorazgeldiyeva ~]$ vi common.h
```

Рисунок 1. Создание файла `common.h`

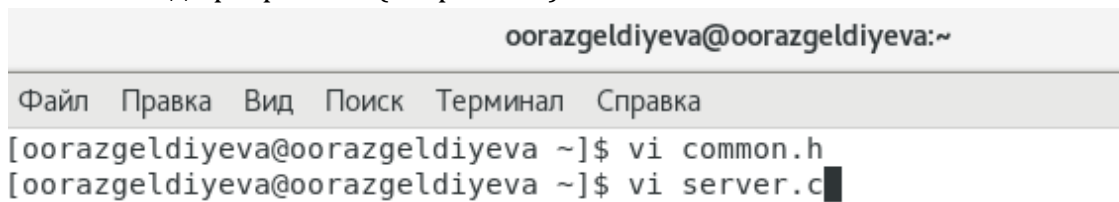
Напишем текст программы согласно образцу. В этом файле будут библиотеки, которые будут включены в программу.

Добавим библиотеку `<time.h>`, которая позволит работать с функцией `clock()`. (см. рис. 2)

```
oorazgeldiyeva@oorazgeldiyeva:~  
Файл  Правка  Вид  Поиск  Терминал  Справка  
#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 <time.h>  
#define FIFO_NAME "/tmp/fifo"  
#define MAX_BUFF 80  
  
#endif /* __COMMON_H__ */  
~
```

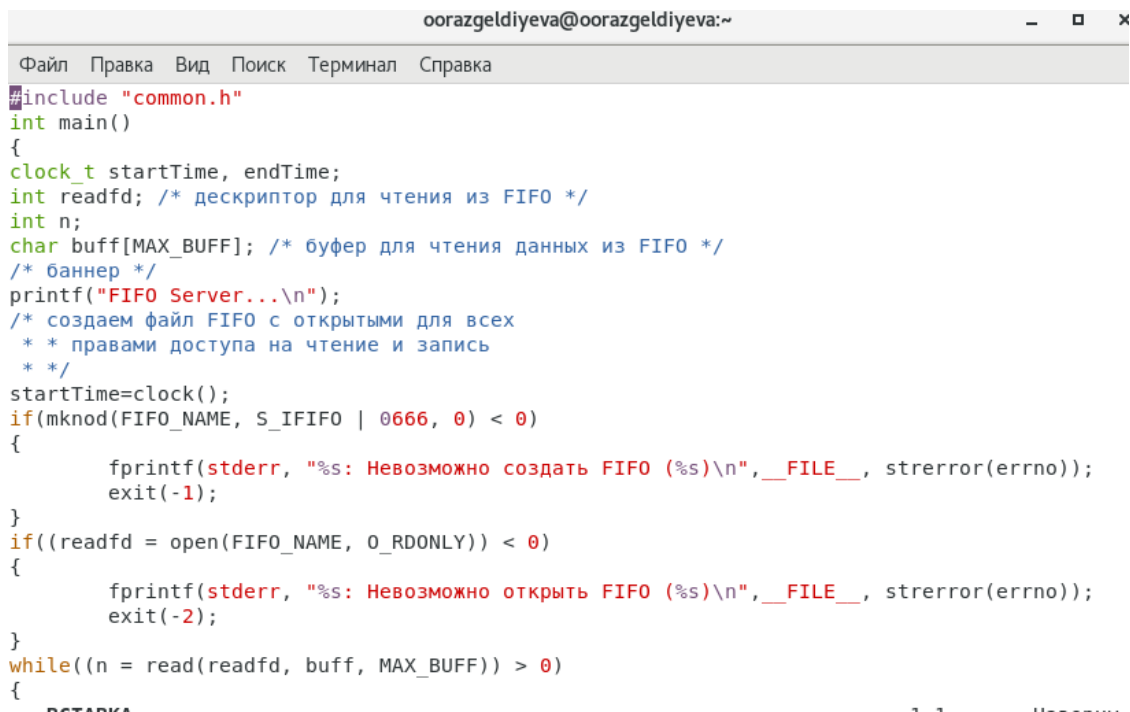
Рисунок 2. `common.h`

3. Создадим и откроем файл `server.c` с помощью редактора `vi` (см. рис. 3), напишем в нем код программы. (см. рис. 4-5)



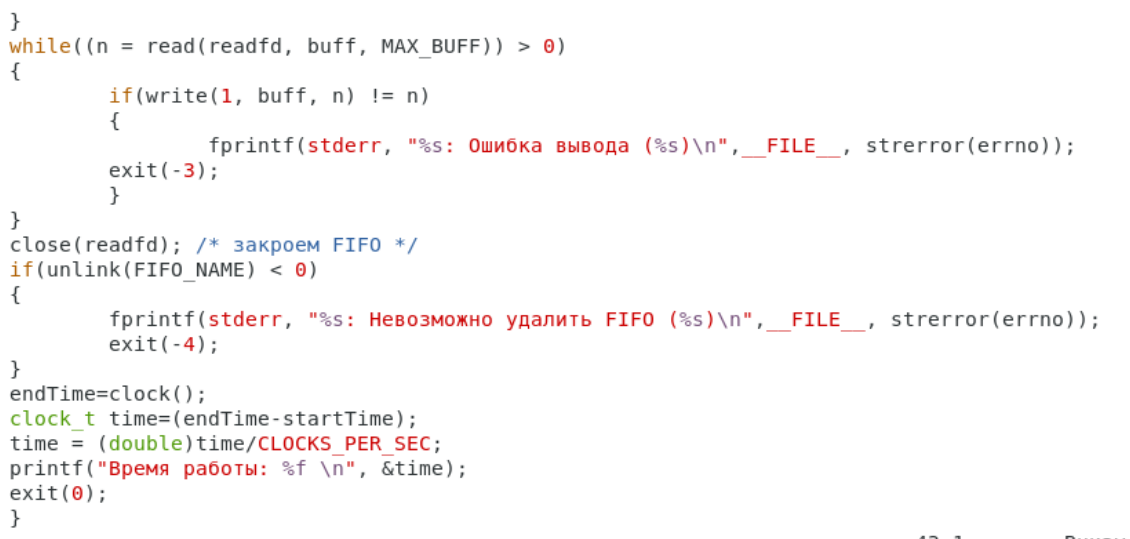
```
oorazgeldiyeva@oorazgeldiyeva:~  
Файл  Правка  Вид  Поиск  Терминал  Справка  
[oorazgeldiyeva@oorazgeldiyeva ~]$ vi common.h  
[oorazgeldiyeva@oorazgeldiyeva ~]$ vi server.c
```

Рисунок 3. Создание файла `server.c`



```
oorazgeldiyeva@oorazgeldiyeva:~  
Файл  Правка  Вид  Поиск  Терминал  Справка  
#include "common.h"  
int main()  
{  
    clock_t startTime, endTime;  
    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  
     */  
    startTime=clock();  
    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);  
    }  
    while((n = read(readfd, buff, MAX_BUFF)) > 0)  
    {  
        /* чтение */  
        printf("%s\n", buff);  
    }  
}
```

Рисунок 4. `server.c`

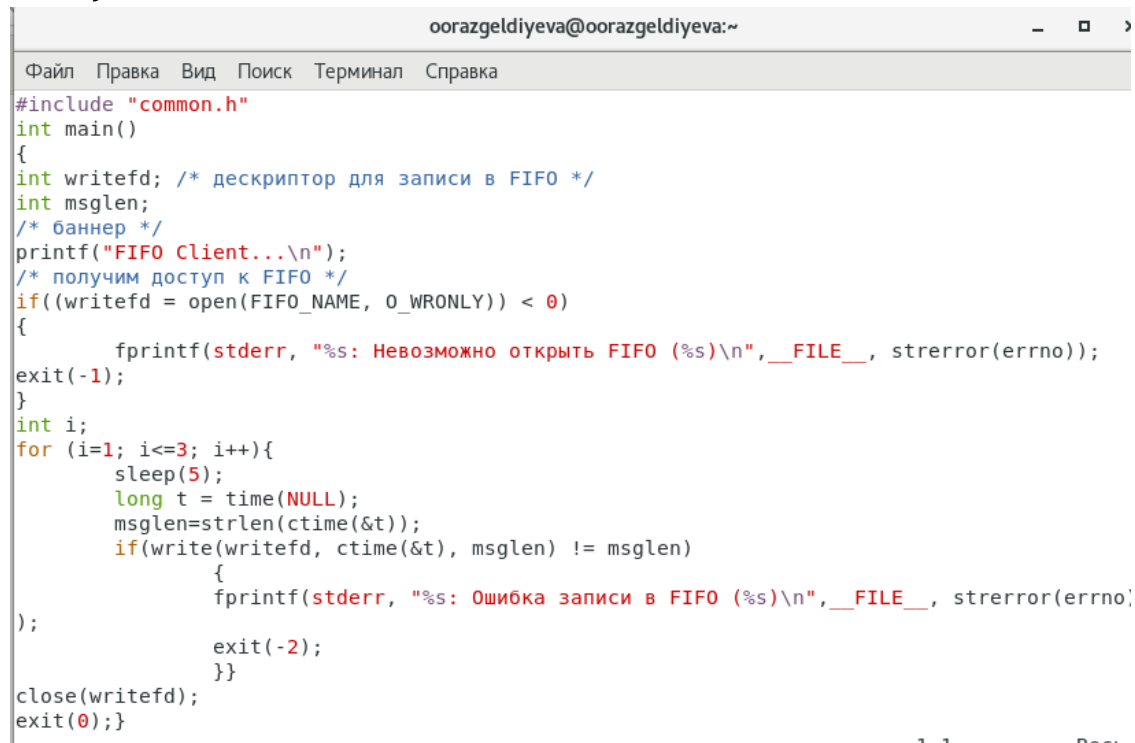


```
}  
while((n = read(readfd, buff, MAX_BUFF)) > 0)  
{  
    if(write(1, buff, n) != n)  
    {  
        fprintf(stderr, "%s: Ошибка вывода (%s)\n", __FILE__, strerror(errno));  
        exit(-3);  
    }  
}  
close(readfd); /* закроем FIFO */  
if(unlink(FIFO_NAME) < 0)  
{  
    fprintf(stderr, "%s: Невозможно удалить FIFO (%s)\n", __FILE__, strerror(errno));  
    exit(-4);  
}  
endTime=clock();  
clock_t time=(endTime-startTime);  
time = (double)time/CLOCKS_PER_SEC;  
printf("Время работы: %f \n", &time);  
exit(0);  
}
```

Рисунок 5. `server.c`

Пояснения: с помощью `clock_t` зададим переменные для начала и конца работы процесса (`startTime`, `endTime`); используя функцию `clock()` устанавливаем начальное и конечное время (`startTime=clock()`, `endTime=clock()`). Записываем и выводим промежуток времени на экран.

4. Создадим и откроем файл `client.c` внесем в него код программы по образцу. (см. рис. 6)



```
#include "common.h"
int main()
{
    int writefd; /* дескриптор для записи в FIFO */
    int msglen;
    /* 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);
    }
    int i;
    for (i=1; i<=3; i++){
        sleep(5);
        long t = time(NULL);
        msglen=strlen(ctime(&t));
        if(write(writefd, ctime(&t), msglen) != msglen)
        {
            fprintf(stderr, "%s: Ошибка записи в FIFO (%s)\n", __FILE__, strerror(errno));
            exit(-2);
        }
    }
    close(writefd);
    exit(0);}
```

Рисунок 6. *client.c*

В текст программы заносим изменения.

Пояснения: задаем цикл `for`, который будет каждые 5 секунд выводить сообщение со временем и датой. Выполнятся это действие будет по 3 раза для каждого клиента.

5. Создадим и откроем с помощью редактора `vi` файл `Makefile`. Запишем без изменений приведенный в образце текст. (см. рис. 7)

```
oorazgeldiyeva@oorazgeldiyeva:~  
Файл  Правка  Вид  Поиск  Терминал  Справка  
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  
~  
~  
~
```

Рисунок 7. Makefile

Пояснения: в этом файле заданы цели all, server, client и clean. Цель all будет выполнять цели server и client, которые в свою очередь компилируют файлы server.c и client.c соответственно; цель clean удаляет созданные после компиляции файлы.

- Используя команду make скомпилируем файлы server.c и client.c. (использовали цель all). (см. рис. 8)

```
oorazgeldiyeva@oorazgeldiyeva:~  
Файл  Правка  Вид  Поиск  Терминал  Справка  
[oorazgeldiyeva@oorazgeldiyeva ~]$ vi common.h  
[oorazgeldiyeva@oorazgeldiyeva ~]$ vi server.c  
[oorazgeldiyeva@oorazgeldiyeva ~]$ vi client.c  
[oorazgeldiyeva@oorazgeldiyeva ~]$ vi Makefile  
[oorazgeldiyeva@oorazgeldiyeva ~]$ make all  
gcc server.c -o server  
gcc client.c -o client  
[oorazgeldiyeva@oorazgeldiyeva ~]$ █
```

Рисунок 8. Компиляция

- Запустим в одной консоли программу server, в других двух - client. (см. рис. 9)

[2] - Функция `clock()`, язык программирования C/C++