

# Именованные каналы

## Лабораторная работа №14

---

Нати Ф. Б.

Российский университет дружбы народов, Москва, Россия

## Информация

---

- Нати Франсиску Бунда
- студент 1 курса, группа НММбд-02-22
- Российский университет дружбы народов



## Вводная часть

---

- Приобретение практических навыков работы с именованными каналами.

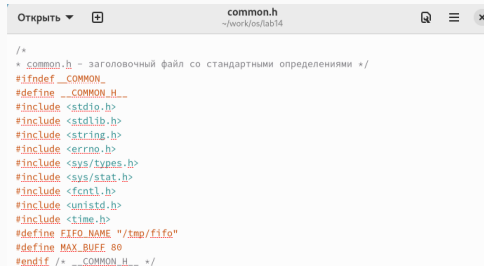
Изучить приведённые в тексте программы `server.c` и `client.c`. Взяв данные примеры за образец, написать аналогичные программы, внося следующие изменения: 1. Работает не 1 клиент, а несколько (например, два). 2. Клиенты передают текущее время с некоторой периодичностью (например, раз в пять секунд). 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?

## Выполнение лабораторной работы №14

---

# Внесение изменений в программы

```
[fbnati@fedora lab14]$ touch common.h  
[fbnati@fedora lab14]$ touch server.c  
[fbnati@fedora lab14]$ touch client.c  
[fbnati@fedora lab14]$ touch Makefile
```



The screenshot shows a code editor window titled "common.h" with the path "~/work/os/lab14". The editor contains the following C header file code:

```
/*  
 * common.h - заголовочный файл со стандартными определениями */  
#ifndef __COMMON_  
#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__ */
```



# Внесение изменений в программы

Открыть ▾ + server.c  
~/work/os/lab14

```
/*
 * server.c - реализация сервера
 * чтобы запустить пример, необходимо:
 * 1. запустить программу server на одной консоли;
 * 2. запустить программу client на другой консоли. */
#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-An");
    /* 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".
```

Открыть ▾ + client.c  
~/work/os/lab14

```
/*
 * client.c - реализация клиента
 * чтобы запустить пример, необходимо:
 * 1. запустить программу server на одной консоли;
 * 2. запустить программу client на другой консоли.
 */
#include "common.h"
#define MESSAGE "Hello Server!!!\n"
int main()
{
    int writefd;
    /* дескриптор для записи в FIFO */
    int msglen;
    /* баннер */
    printf("FIFO Client-An");
    /*цикл, отвечающий за отправку сообщения о текущем времени */
    for(int i=0; i<4; i++)
    {
```

# Внесение изменений в программы

Открыть  • Makefile  
~/work/os/lab14   

```
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
```

```
gcc server.c -o server  
gcc client.c -o client
```

# Внесение изменений в программы

```
fbnati@fedora: ~/work/os/lab14
[fbnati@fedora lab14]$ ./client
client.c: невозможно открыть FIFO (No such file or directory)
FIFO client_An[fbnati@fedora lab14]$ ./client
FIFO client_An[fbnati@fedora lab14]$

fbnati@fedora: ~/work/os/lab14
[fbnati@fedora lab14]$ touch common.h
[fbnati@fedora lab14]$ touch server.c
[fbnati@fedora lab14]$ touch client.c
[fbnati@fedora lab14]$ touch Makefile
[fbnati@fedora lab14]$ make all
gcc server.c -o server
gcc client.c -o client
[fbnati@fedora lab14]$ ./server
Hello Server!!!
Hello Server!!!
Hello Server!!!
Hello Server!!!
Hello Server!!!
Hello Server!!!
Hello Server!!!
Hello Server!!!
FIFO Server_An[fbnati@fedora lab14]$

fbnati@fedora: ~/work/os/lab14
[fbnati@fedora lab14]$ ./client
client_An[fbnati@fedora lab14]$
```

```
[fbnati@fedora lab14]$ ./server
server.c: невозможно создать FIFO (File exists)
FIFO Server_An[fbnati@fedora lab14]$
```

10. Прототип функции `strerror`: `char * strerror( int errnum );`. Функция `strerror` интерпретирует номер ошибки, передаваемый в функцию в качестве аргумента – `errnum`, 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

8. Количество процессов, которые могут параллельно присоединяться к любому концу канала, не ограничено. Однако если два или более процесса записывают в канал данные одновременно, каждый процесс за один раз может записать максимум PIPE BUF байтов данных. Предположим, процесс (назовем его А) пытается записать X байтов данных в канал, в котором имеется место для Y байтов данных. Если X больше, чем Y, только первые Y байтов данных записываются в канал, и процесс блокируется. Запускается другой процесс (например, В); в это время в канале появляется свободное пространство (благодаря третьему процессу, считывающему данные из канала). Процесс В записывает данные в канал. Затем, когда выполнение процесса А возобновляется, он записывает оставшиеся X-Y байтов данных в

## Результаты

---

В ходе выполнения были приобретены навыки работы с именованными каналами.