

# Лабораторная работа №14: Презентация.

Именованные каналы.

---

Евдокимов Максим Михайлович. Группа - НФИбд-01-20.<sup>1</sup>

28 декабря, 2023, Москва, Россия

<sup>1</sup>Российский Университет Дружбы Народов

## Цели и задачи работы

---

Приобретение практических навыков работы с именованными каналами.

1. Научится создавать локальные сервера.
2. Научится создавать клиентов (ботов).
3. Изучить понятия именованные каналы, сигналы.

## Указание к работе

---

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: общедоступные (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.

## Выполнение лабораторной работы

---

Изучите приведённые в тексте программы server.c и client.c. Взяв данные примеры за образец, напишите аналогичные программы, внося следующие изменения:

```
max@evdokinov:~$ mkdir lab14; cd lab14
max@evdokinov:~/lab14$ touch common.h server.c client.c client2.c makefile
max@evdokinov:~/lab14$ gedit common.h

(gedit:14334): Gtk-WARNING **: 16:32:55.865: Calling org.freedesktop.portal.Inhibit.Inhibit failed: GDBus.Error:org.freedesktop.DBus.Error.UnknownMethod: Инт
ерфейс «org.freedesktop.portal.Inhibit» для пути /org/freedesktop/portal/desktop объекта не найден
max@evdokinov:~/lab14$ gedit server.c

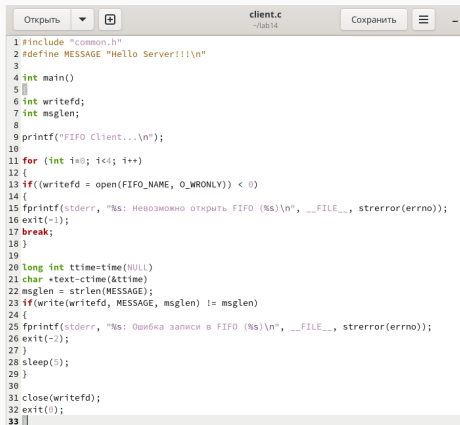
(gedit:14379): Gtk-WARNING **: 16:33:13.891: Calling org.freedesktop.portal.Inhibit.Inhibit failed: GDBus.Error:org.freedesktop.DBus.Error.UnknownMethod: Инт
ерфейс «org.freedesktop.portal.Inhibit» для пути /org/freedesktop/portal/desktop объекта не найден
max@evdokinov:~/lab14$ gedit client.c

(gedit:14427): Gtk-WARNING **: 16:33:42.751: Calling org.freedesktop.portal.Inhibit.Inhibit failed: GDBus.Error:org.freedesktop.DBus.Error.UnknownMethod: Инт
ерфейс «org.freedesktop.portal.Inhibit» для пути /org/freedesktop/portal/desktop объекта не найден
max@evdokinov:~/lab14$
```

Рис. 1: Создание директории подготовка файлов

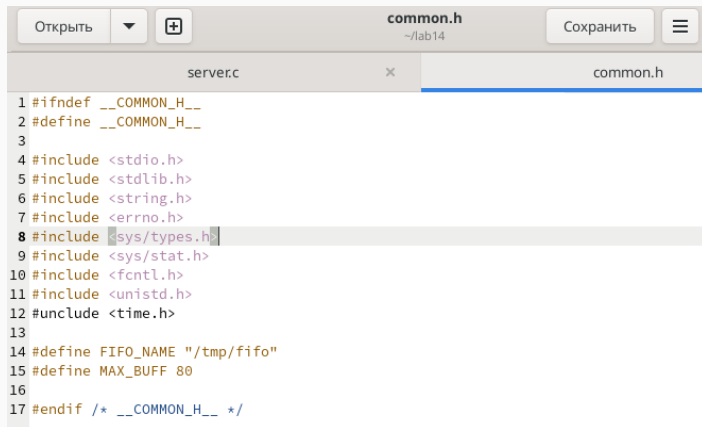


1. Работает не 1 клиент, а несколько (например, два).



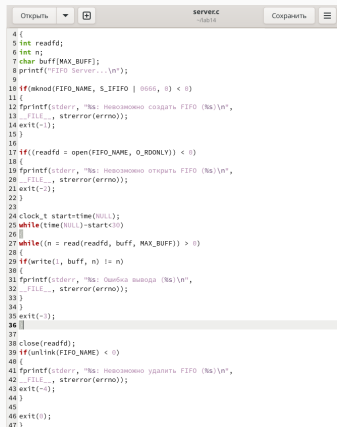
```
1 #include "common.h"
2 #define MESSAGE "Hello Server!!!\n"
3
4 int main()
5 {
6     int writefd;
7     int msglen;
8
9     printf("FIFO client...\n");
10
11     for (int i=0; i<4; i++)
12     {
13         if((writefd = open(FIFO_NAME, O_WRONLY)) < 0)
14         {
15             fprintf(stderr, "%s: Невозможно открыть FIFO (%s)\n", __FILE__, strerror(errno));
16             exit(-1);
17             break;
18         }
19
20         long int ttime=time(NULL)
21         char *text=ctime(&ttime)
22         msglen = strlen(MESSAGE);
23         if(write(writefd, MESSAGE, msglen) != msglen)
24         {
25             fprintf(stderr, "%s: Ошибка записи в FIFO (%s)\n", __FILE__, strerror(errno));
26             exit(-2);
27         }
28         sleep(5);
29     }
30
31     close(writefd);
32     exit(0);
33 }
```

Рис. 2: Код client.c



```
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 #include <unistd.h>
12 #unclude <time.h>
13
14 #define FIFO_NAME "/tmp/fifo"
15 #define MAX_BUFF 80
16
17 #endif /* __COMMON_H__ */
```

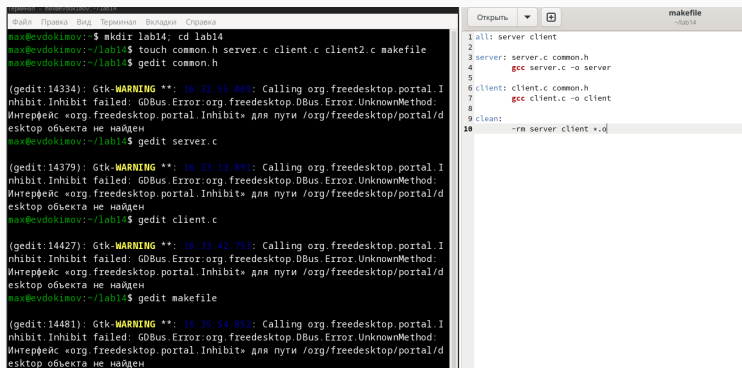
Рис. 3: Код common.h



```
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
24 clock_t start=time(NULL);
25 while(time(NULL)-start<30)
26 {
27 while((n = read(readfd, buff, MAX_BUFF)) > 0)
28 {
29 if(write(1, buff, n) != n)
30 {
31 fprintf(stderr, "%s: Ошибка вывода (%s)\n",
32 __FILE__, strerror(errno));
33 }
34 }
35 exit(-3);
36 }
37
38 close(readfd);
39 if(unlink(FIFO_NAME) < 0)
40 {
41 fprintf(stderr, "%s: Невозможно удалить FIFO (%s)\n",
42 __FILE__, strerror(errno));
43 exit(-4);
44 }
45
46 exit(0);
47 }
```

Рис. 4: Код server.c

2. Клиенты передают текущее время с некоторой периодичностью (например, раз в пять секунд). Используйте функцию `sleep()` для приостановки работы клиента.



The image shows a terminal window on the left and a makefile editor on the right. The terminal window displays the following commands and output:

```
max@evdokimov:~$ mkdir lab14; cd lab14
max@evdokimov:~/lab14$ touch common.h server.c client.c client2.c makefile
max@evdokimov:~/lab14$ gedit common.h

(gedit:14334): Gtk-WARNING **: 16:32:55.069: Calling org.freedesktop.portal.Inhibit.Inhibit failed: GDBus.Error:org.freedesktop.DBus.Error.UnknownMethod: Интерфейс «org.freedesktop.portal.Inhibit» для пути /org/freedesktop/portal/desktop объекта не найден
max@evdokimov:~/lab14$ gedit server.c

(gedit:14379): Gtk-WARNING **: 16:33:13.891: Calling org.freedesktop.portal.Inhibit.Inhibit failed: GDBus.Error:org.freedesktop.DBus.Error.UnknownMethod: Интерфейс «org.freedesktop.portal.Inhibit» для пути /org/freedesktop/portal/desktop объекта не найден
max@evdokimov:~/lab14$ gedit client.c

(gedit:14427): Gtk-WARNING **: 16:33:42.751: Calling org.freedesktop.portal.Inhibit.Inhibit failed: GDBus.Error:org.freedesktop.DBus.Error.UnknownMethod: Интерфейс «org.freedesktop.portal.Inhibit» для пути /org/freedesktop/portal/desktop объекта не найден
max@evdokimov:~/lab14$ gedit makefile

(gedit:14481): Gtk-WARNING **: 16:35:54.852: Calling org.freedesktop.portal.Inhibit.Inhibit failed: GDBus.Error:org.freedesktop.DBus.Error.UnknownMethod: Интерфейс «org.freedesktop.portal.Inhibit» для пути /org/freedesktop/portal/desktop объекта не найден
```

The makefile editor shows the following content:

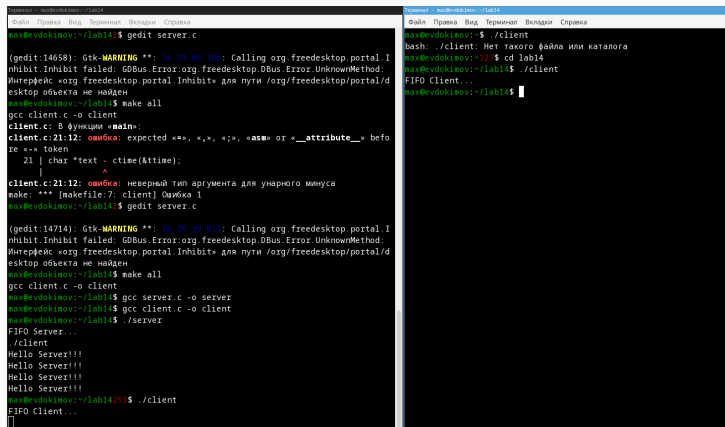
```
all: server client
server: server.c common.h
    gcc server.c -o server
client: client.c common.h
    gcc client.c -o client
clean:
    rm server client *.o
```

Рис. 5: Makefile

```
max@evdokimov:~/lab14$ make all
gcc client.c -o client
max@evdokimov:~/lab14$ gcc server.c -o server
max@evdokimov:~/lab14$ gcc client.c -o client
max@evdokimov:~/lab14$ ./server
```

Рис. 6: Компиляция

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?



```
Terminal - max@evdokinov:~/lab14
Файл Правка Вид Терминал Вкладки Справка
max@evdokinov:~/lab14$ gedit server.c

(gedit:14658): Gtk-WARNING **: 16:39:06.166: Calling org.freedesktop.portal.Inhibit.Inhibit failed: GDBus.Error:org.freedesktop.DBus.Error.UnknownMethod: Интерфейс «org.freedesktop.portal.Inhibit» для пути /org/freedesktop/portal/desktop объекта не найден
max@evdokinov:~/lab14$ make all
gcc client.c -o client
client.c: В функции «main»:
client.c:21:12: ошибка: expected «=», «,», «;», «asm» or «__attribute__» before «-» token
   21 | char *text = ctime(&time);
       |             ^
client.c:21:12: ошибка: неверный тип аргумента для унарного минуса
make: *** [makefile:7: client] Ошибка 1
max@evdokinov:~/lab14$ gedit server.c

(gedit:14714): Gtk-WARNING **: 16:39:39.811: Calling org.freedesktop.portal.Inhibit.Inhibit failed: GDBus.Error:org.freedesktop.DBus.Error.UnknownMethod: Интерфейс «org.freedesktop.portal.Inhibit» для пути /org/freedesktop/portal/desktop объекта не найден
max@evdokinov:~/lab14$ make all
gcc client.c -o client
max@evdokinov:~/lab14$ gcc server.c -o server
max@evdokinov:~/lab14$ gcc client.c -o client
max@evdokinov:~/lab14$ ./server
FIFO Server...
./client
Hello Server!!!
Hello Server!!!
Hello Server!!!
Hello Server!!!
max@evdokinov:~/lab14$ ./client
FIFO Client...

```

```
Terminal - max@evdokinov:~/lab14
Файл Правка Вид Терминал Вкладки Справка
max@evdokinov:~/lab14$ ./client
bash: ./client: Нет такого файла или каталога
max@evdokinov:~/lab14$ cd lab14
max@evdokinov:~/lab14$ ./client
FIFO Client...
max@evdokinov:~/lab14$
```

Рис. 7: Результат работы сервера

client.c:

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

int main()
{
    int writefd;
    int msglen;

    printf("FIFO Client...\n");
```

```
for (int i=0; i<4; i++)
{
    if((writefd = open(FIFO_NAME, O_WRONLY)) < 0)
    {
        fprintf(stderr, "%s: Невозможно открыть FIFO (%s)\n", __FILE__, strerror(errno));
        exit(-1);
    }
    break;
}
```



```
long int ttime=time(NULL);
char *text=ctime(&ttime);
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);
}

close(writefd);
exit(0);
}
```

common.h:

```
#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_BUFSIZE 80
```

server.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 start=time(NULL);
while(time(NULL)-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);
}
```

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

Makefile:

```
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
```

## 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). Поскольку файл находится на локальной файловой системе, данное IPC используется внутри одной системы.

## 2. Возможно ли создание неименованного канала из командной строки?

Чтобы создать неименованный канал из командной строки нужно использовать символ `|`, служащий для объединения двух и более процессов: `process_1 | process_2 | process_3 |` и так далее.

3. Возможно ли создание именованного канала из командной строки?

Чтобы создать именованный канал из командной строки нужно использовать либо команду «mknod», либо команду «mkfifo».

#### 4. Опишите функцию языка C, создающую неименованный канал.

Неименованный канал является средством взаимодействия между связанными процессами –родительским и дочерним. Родительский процесс создает канал при помощи системного вызова: «`int pipe(int fd[2]);`». Массив из двух целых чисел является выходным параметром этого системного вызова. Если вызов выполнен нормально, то этот массив содержит два файловых дескриптора. `fd[0]` является дескриптором для чтения из канала, `fd[1]` –дескриптором для записи в канал. Когда процесс порождает другой процесс, дескрипторы родительского процесса наследуются дочерним процессом, и, таким образом, прокладывается трубопровод между двумя процессами. Естественно, что один из процессов использует канал только для чтения, а другой –только для записи. Поэтому, если, например, через канал должны передаваться данные из родительского процесса в дочерний, родительский процесс сразу после запуска дочернего процесса закрывает дескриптор канала для чтения, а дочерний процесс закрывает дескриптор для записи. Двухнаправленный обмен просто подразумевает 2 канала.

5. Опишите функцию языка C, создающую именованный канал.

«`int mkfifo(const char pathname, mode_t mode);`», где первый параметр – путь, где будет располагаться FIFO (имя файла, идентифицирующего канал), второй параметр определяет режим работы с FIFO (маска прав доступа к файлу), «`mknod (namefile, IFIFO | 0666, 0)`», где `namefile` – имя канала, `0666` – к каналу разрешен доступ на запись и на чтение любому запросившему процессу), «`int mknod(const char pathname, mode_t mode, dev_t dev);`». Функция `mkfifo()` создает канал и файл соответствующего типа. Если указанный файл канала уже существует, `mkfifo()` возвращает `-1`. После создания файла канала процессы, участвующие в обмене данными, должны открыть этот файл либо для записи, либо для чтения. 6). При чтении меньшего числа байтов, чем находится в канале или FIFO, возвращается требуемое число байтов, остаток сохраняется для последующих чтений. При чтении большего числа байтов, чем находится в канале или FIFO, возвращается доступное число байтов. Процесс, читающий из канала, должен соответствующим образом обработать ситуацию, когда прочитано меньше, чем заказано.

6. Что будет в случае прочтения из `fifo` меньшего числа байтов, чем находится в канале?  
Большого числа 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=ERRPIPE`) (если процесс не установил обработки сигнала `SIGPIPE`, производится обработка по умолчанию – процесс завершается).

7. Аналогично, что будет в случае записи в fifo меньшего числа байтов, чем позволяет буфер? Большого числа 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=ERRPIPE`) (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 байты `count` из буфера `buffer` в файл, связанный с `handle`. Операции `write` начинаются с текущей позиции указателя на файл (указатель ассоциирован с заданным файлом). Если файл открыт для добавления, операции выполняются в конец файла. После осуществления операций записи указатель на файл(если он есть) увеличивается на количество действительно записанных байтов. Функция `write` возвращает число действительно записанных байтов. Возвращаемое значение должно быть положительным, но меньше числа `count` (например, когда размер для записи `count` байтов выходит за пределы пространства на диске). Возвращаемое значение -1 указывает на ошибку; `errno` устанавливается в одно из следующих значений: `EACCES` – файл открыт для чтения или закрыт для записи, `EBADF` – неверный `handle`-р файла, `ENOSPC` – на устройстве нет свободного места. Единица в вызове функции `write` в программе `server.c` означает идентификатор (дескриптор потока) стандартного потока вывода.

## 10. Опишите функцию `strerror`.

Прототип функции `strerror`: `char * strerror( int errornum );`. Функция `strerror` интерпретирует номер ошибки, передаваемый в функцию в качестве аргумента `-errornum`, в понятное для человека текстовое сообщение (строку). Откуда берутся эти ошибки? Ошибки эти возникают при вызове функций стандартных Сибиблиотек. 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 различаться, это зависит от платформы и компилятора.

## Выводы по проделанной работе

---

В ходе выполнения лабораторной работы были изучены и проработаны на практике именованные каналы.