

# **Лабораторная работа No 14.**

**Именованные каналы**

Сарасбати Брасалес

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>5</b>
<b>2</b>	<b>Задание</b>	<b>6</b>
<b>3</b>	<b>Теоретическое введение</b>	<b>7</b>
<b>4</b>	<b>Выполнение лабораторной работы</b>	<b>8</b>
<b>5</b>	<b>Выводы</b>	<b>11</b>
<b>6</b>	<b>Контрольные вопросы</b>	<b>12</b>
	<b>Список литературы</b>	<b>14</b>

# Список иллюстраций

4.1	.....	8
4.2	.....	8
4.3	.....	9
4.4	.....	9
4.5	.....	9
4.6	.....	10

## Список таблиц

# **1 Цель работы**

Приобретение практических навыков работы с именованными каналами.

## 2 Задание

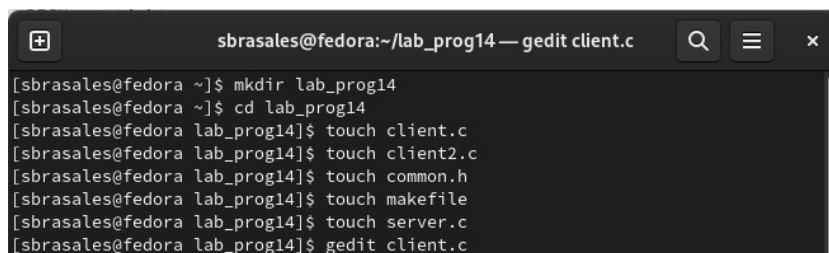
Изучите приведённые в тексте программы `server.c` и `client.c`. Взяв данные примеры за образец, напишите аналогичные программы, внося следующие изменения: 1. Работает не 1 клиент, а несколько (например, два). 2. Клиенты передают текущее время с некоторой периодичностью (например, раз в пять секунд).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

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: общепонимание (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)`

## 4 Выполнение лабораторной работы

### Создание файлов



```
sbrasales@fedora: ~/lab_prog14 — gedit client.c
[sbrasales@fedora ~]$ mkdir lab_prog14
[sbrasales@fedora ~]$ cd lab_prog14
[sbrasales@fedora lab_prog14]$ touch client.c
[sbrasales@fedora lab_prog14]$ touch client2.c
[sbrasales@fedora lab_prog14]$ touch common.h
[sbrasales@fedora lab_prog14]$ touch makefile
[sbrasales@fedora lab_prog14]$ touch server.c
[sbrasales@fedora lab_prog14]$ gedit client.c
```

Рис. 4.1:

### client.c



```
9 #include "common.h"
10
11 #define MESSAGE "Hello Server!!\n"
12
13 int
14 main()
15 {
16     int writefd; /* дескриптор для записи в FIFO */
17     int msglen;
18
19     /* Бомбард */
20     printf("FIFO client...\n");
21
22     for (int i = 1; i <= 10; i++){
23
24         /* получим доступ к FIFO */
25         if((writefd = open(FIFO_NAME, O_WRONLY)) < 0)
26         {
27             fprintf(stderr, "%s: Невозможно открыть FIFO (%s)\n",
28                 _FILE_, strerror(errno));
29             exit(-1);
30         }
31
32         /* передадим сообщение серверу */
33         msglen = strlen(MESSAGE);
34         if(write(writefd, MESSAGE, msglen) != msglen)
35         {
36             fprintf(stderr, "%s: Ошибка записи в FIFO (%s)\n",
37                 _FILE_, strerror(errno));
38             exit(-2);
39         }
40     }
41 }
```

Рис. 4.2:

### client2.c



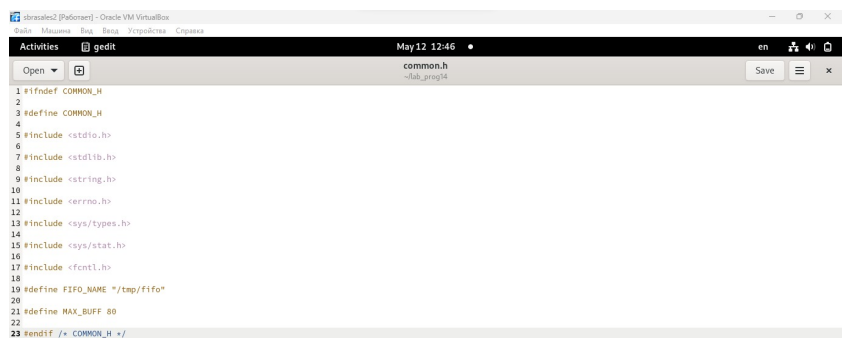


The screenshot shows a gedit editor window titled "client2.c" with a file path of "~lab\_prog14". The code is in C and includes a header file "common.h". It defines a message "Hello Server!!\n" and a loop that writes this message to a FIFO named "fifo" 15 times. The code also includes error handling for opening the FIFO and printing the error message.

```
1 #include "common.h"
2 #include <time.h>
3 #define MESSAGE "Hello Server!!\n"
4
5 int
6 main()
7 {
8     int writefd; /* дескриптор для записи в FIFO */
9     int msglen;
10    long int time;
11
12    for(int i=0; i<15; i++)
13    {
14        time=time(NULL);
15        printf(ctime(&time));
16        /* Sleep */
17        printf("FIFO client...\n");
18
19        /* получим доступ к FIFO */
20        if((writefd = open(FIFO_NAME, O_WRONLY)) < 0)
21        {
22            printf(stderr, "%s: Невозможно открыть FIFO (%s)\n",
23                _FILE_, strerror(errno));
24            exit(-1);
25        }
26
27        /* передаем сообщение серверу */
28        msglen = strlen(MESSAGE);
29        if(write(writefd, MESSAGE, msglen) != msglen)
```

Рис. 4.3:

## common.h

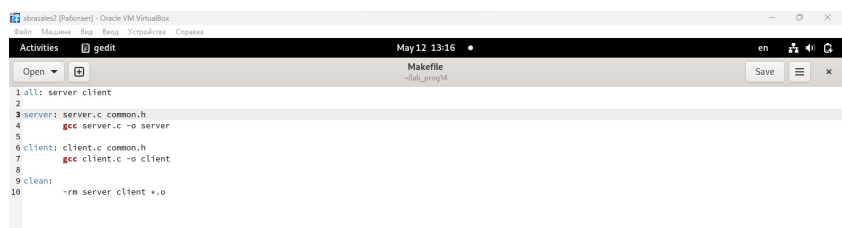


The screenshot shows a gedit editor window titled "common.h" with a file path of "~lab\_prog14". The code defines a macro "COMMON\_H" and includes several standard C headers: "stdio.h", "stdlib.h", "string.h", "errno.h", "sys/types.h", "sys/stat.h", and "fcntl.h". It also defines "FIFO\_NAME" as "/tmp/fifo" and "MAX\_BUFF" as 80.

```
1 #ifndef COMMON_H
2
3 #define COMMON_H
4
5 #include <stdio.h>
6
7 #include <stdlib.h>
8
9 #include <string.h>
10
11 #include <errno.h>
12
13 #include <sys/types.h>
14
15 #include <sys/stat.h>
16
17 #include <fcntl.h>
18
19 #define FIFO_NAME "/tmp/fifo"
20
21 #define MAX_BUFF 80
22
23 #endif /* COMMON_H */
```

Рис. 4.4:

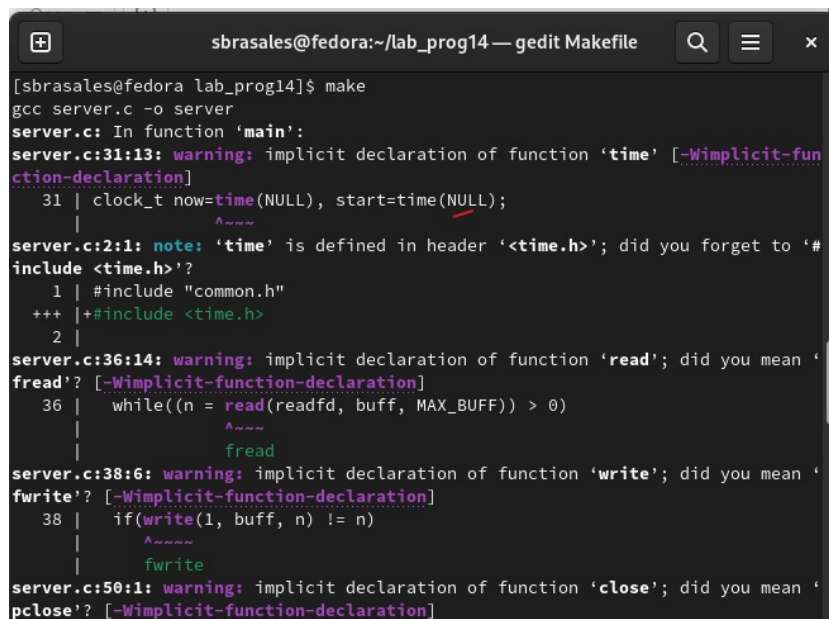
## Makefile



The screenshot shows a gedit editor window titled "Makefile" with a file path of "~lab\_prog14". The code defines two targets: "server" and "client". The "server" target depends on "server.c" and "common.h", and is compiled with "gcc" to produce "server". The "client" target depends on "client.c" and "common.h", and is compiled with "gcc" to produce "client". The "clean" target removes the "server" and "client" files.

```
1 all: server client
2
3 server: server.c common.h
4     gcc server.c -o server
5
6 client: client.c common.h
7     gcc client.c -o client
8
9 clean:
10     ~rm server client *.o
```

Рис. 4.5:



```
sbrasales@fedora:~/lab_prog14 — gedit Makefile
[sbrasales@fedora lab_prog14]$ make
gcc server.c -o server
server.c: In function 'main':
server.c:31:13: warning: implicit declaration of function 'time' [-Wimplicit-function-declaration]
   31 | clock_t now=time(NULL), start=time(NULL);
      |             ^
server.c:2:1: note: 'time' is defined in header '<time.h>'; did you forget to '#include <time.h>'?
   1 | #include "common.h"
+++ |+#include <time.h>
   2 |
server.c:36:14: warning: implicit declaration of function 'read'; did you mean 'fread'? [-Wimplicit-function-declaration]
   36 | while((n = read(readfd, buff, MAX_BUFF)) > 0)
      |             ^
      |             fread
server.c:38:6: warning: implicit declaration of function 'write'; did you mean 'fwrite'? [-Wimplicit-function-declaration]
   38 | if(write(1, buff, n) != n)
      |     ^
      |     fwrite
server.c:50:1: warning: implicit declaration of function 'close'; did you mean 'pclose'? [-Wimplicit-function-declaration]
```

Рис. 4.6:

## **5 Выводы**

Мы приобрели 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? Создание неименованного канала из командной строки возможно командой `pipe`. 3. Возможно ли создание именованного канала из командной строки? Создание именованного канала из командной строки возможно с помощью `mkfifo`. 4. Опишите функцию языка C, создающую неименованный канал. Функция языка C, создающая 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. Функция языка C, создающая 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, возвращается требуемое число байтов, остаток сохраняется для следующих чтений. При чтении большего числа байтов, возвращается доступное число байтов 7. 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=EPipe`) (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 `length` байтов из буфера `buffer` в файл, определенный дескриптором файла `fd`. Эта операция чисто 'двоичная' и без буферизации. При единице возвращает действительное число байтов. Функция `write` возвращает число действительно записанных в файл байтов или -1 при ошибке, устанавливая при этом `errno`.

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.

## **Список литературы**