

Лабораторная работа №14. Именованные каналы.

СИССЕ МОХАМЕД ЛАМИН;НММбд-01-22

¹RUDN University, Moscow, Russian Federation

Приобретение практических навыков работы с именованными каналами.

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?

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: общенуксные (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)`.

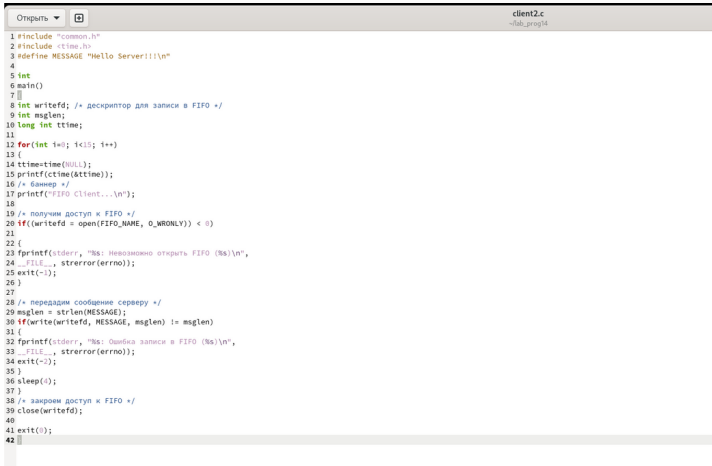
Изучите приведённые в тексте программы `server.c` и `client.c`. Взяв данные примеры за образец, напишите аналогичные программы, внося следующие изменения: 1. Работает не 1 клиент, а несколько (например, два). 2. Клиенты передают текущее время с некоторой периодичностью (например, раз в пять секунд). Используйте функцию `sleep()` для приостановки работы клиента. (рис. (fig:001?; fig:002?))

Выполнение лабораторной работы

```
client.c
5 * 1. запустить программу server на одной консоли;
6 * 2. запустить программу client на другой консоли.
7 */
8
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     /* баннер */
20     printf("FIFO Client...\n");
21
22     for (int i; i<4; 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(-1);
39         }
40
41         sleep (5);
42     }
43
44     /* закроем доступ к FIFO */
45     close(writefd);
46
47     exit(0);
48 }
49
```

Рис. 1: Текст программы

Выполнение лабораторной работы



```
client2.c
~lab_prog14

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 ttime;
11
12    for(int i=0; i<10; i++)
13    {
14        ttime=time(NULL);
15        printf("ctime(&ttime);");
16        /* бэннер */
17        printf("FIFO Client...\n");
18
19        /* получим доступ к FIFO */
20        if((writefd = open(FIFO_NAME, O_WRONLY)) < 0)
21        {
22            fprintf(stderr, "%s: Невозможно открыть FIFO (%s)\n",
23                _FILE_, strerror(errno));
24            exit(-1);
25        }
26
27        /* передадим сообщение серверу */
28        msglen = strlen(MESSAGE);
29        if(write(writefd, MESSAGE, msglen) != msglen)
30        {
31            fprintf(stderr, "%s: Ошибка записи в FIFO (%s)\n",
32                _FILE_, strerror(errno));
33            exit(-2);
34        }
35
36        sleep(1);
37    }
38    /* закроем доступ к FIFO */
39    close(writefd);
40
41    exit(0);
42 }
```

Рис. 2: Текст программы

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? (рис. (fig:003?; fig:004?; fig:005?; fig:006?))

Выполнение лабораторной работы

```
1 /*
2 * server.c - реализация сервера
3 *
4 * чтобы запустить пример, необходимо:
5 * 1. запустить программу server на одной консоли;
6 * 2. запустить программу client на другой консоли.
7 */
8
9 #include "common.h"
10
11 int
12 main()
13 {
14     int readfd; /* дескриптор для чтения из FIFO */
15     int n;
16     char buff[MAX_BUFF]; /* буфер для чтения данных из FIFO */
17
18     /* баннер */
19     printf("FIFO Server...\n");
20
21     /* создан файл FIFO с открытыми для всех
22      * правами доступа на чтение и запись
23      */
24     if(mknod(FIFO_NAME, S_IFIFO | 0666, 0) < 0)
25     {
26         fprintf(stderr, "%s: Невозможно создать FIFO (%s)\n",
27             __FILE__, strerror(errno));
28         exit(-1);
29     }
30
31     /* открыт FIFO на чтение */
32     if((readfd = open(FIFO_NAME, O_RDONLY)) < 0)
33     {
34         fprintf(stderr, "%s: Невозможно открыть FIFO (%s)\n",
35             __FILE__, strerror(errno));
36         exit(-1);
37     }
38
39     clock_t beginning=time(NULL), clock_t now=time(NULL);
40     while (beginning-now<30)
41     {
42         /* читаем данные из FIFO и выводим на экран */
43         while((n = read(readfd, buff, MAX_BUFF)) > 0)
44         {
45             if(write(1, buff, n) != n)
```

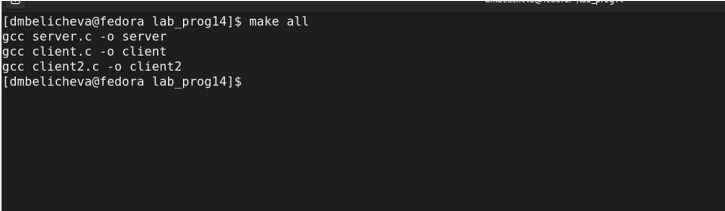
Рис. 3: Текст программы



The image shows a screenshot of a text editor window titled "Makefile" with the path "~lab_prog14". The editor contains a Makefile with the following content:

```
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 client2: client2.c common.h
10    gcc client2.c -o client
11
12 clean:
13     -rm server client *.o
```

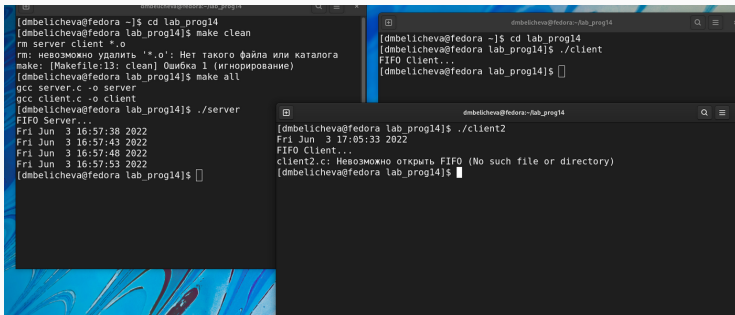
Рис. 4: Текст программы

A terminal window with a dark background and light text. The prompt is [dmbelicheva@fedora lab_prog14]\$. The user enters 'make all', followed by three lines of gcc commands: 'gcc server.c -o server', 'gcc client.c -o client', and 'gcc client2.c -o client2'. The prompt returns after the last command.

```
[dmbelicheva@fedora lab_prog14]$ make all
gcc server.c -o server
gcc client.c -o client
gcc client2.c -o client2
[dmbelicheva@fedora lab_prog14]$
```

Рис. 5: Компиляция

Выполнение лабораторной работы



The image displays three terminal windows from a Fedora system, showing the steps to compile and run a C program that uses FIFO (Named Pipes).

Terminal 1 (Left): Shows the compilation process. The user navigates to the `lab_prog14` directory and runs `make clean`, which fails with an error: `rm: невозможно удалить '*.o': Нет такого файла или каталога`. Then, `make all` is run, which successfully compiles `server.c` into `server` and `client.c` into `client`. The user then runs `./server`, which outputs `FIFO Server...`.

Terminal 2 (Top Right): Shows the user running `./client`, which outputs `FIFO Client...`.

Terminal 3 (Bottom Right): Shows the user running `./client2`, which outputs `FIFO Client...`. Then, the user runs `client2.c`, which results in an error: `client2.c: Невозможно открыть FIFO (No such file or directory)`.

Рис. 6: Результат

В процессе выполнения лабораторной работы я приобрела практические навыки работы с именованными каналами.

1. Лабораторная работа № 14. Именованные каналы [Электронный ресурс]. URL: <https://esystem.rudn.ru/>.

Спасибо за внимание!