

Лабораторная работа №14

Именованные каналы

Беличева Д.М.;НКНбд-01-21

Содержание

1	Цель работы	4
2	Задание	5
3	Теоретическое введение	6
4	Выполнение лабораторной работы	7
5	Выводы	10
6	Контрольные вопросы	11
	Список литературы	13

Список иллюстраций

4.1	Текст программы	7
4.2	Текст программы	8
4.3	Текст программы	8
4.4	Текст программы	9
4.5	Компиляция	9
4.6	Результат	9

1 Цель работы

Приобретение практических навыков работы с именованными каналами.

2 Задание

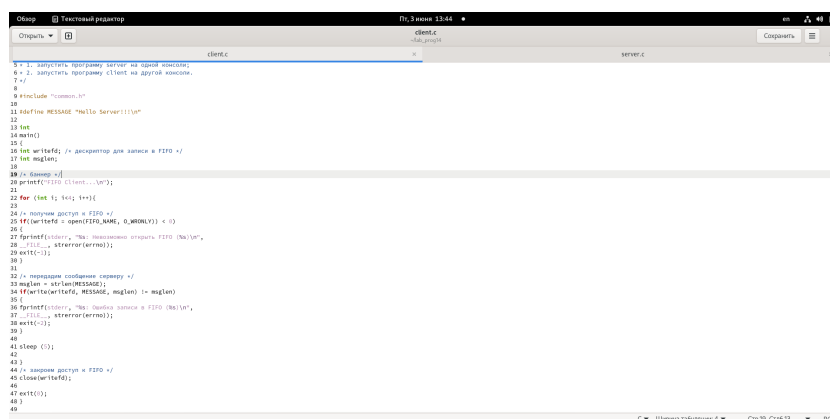
Изучите приведённые в тексте программы `server.c` и `client.c`. Взяв данные примеры за образец, напишите аналогичные программы, внося следующие изменения: 1. Работает не 1 клиент, а несколько (например, два). 2. Клиенты передают текущее время с некоторой периодичностью (например, раз в пять секунд).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

4 Выполнение лабораторной работы

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. 4.1, 4.2)



```
1 // 1. запустить программу server на одной машине.  
2 // 2. запустить программу client на другой машине.  
3 //  
4 #include "common.h"  
5  
6 #define MESSAGE "Hello Server!!!"  
7  
8 int  
9 main()  
10 {  
11     int sockfd; /* дескриптор для записи в FIFO */  
12     int n;  
13     /* Запуск */  
14     printf("FIFO client...\n");  
15  
16     for (n = 1; n <= 10; n++)  
17     {  
18         /* получить доступ к FIFO */  
19         if (waitfd < open(FIFO_NAME, O_WRONLY) < 0)  
20         {  
21             printf(stderr, "Ошибка: невозможно открыть FIFO (%d)\n",  
22                    errno);  
23             return 1;  
24         }  
25  
26         /* передать сообщение серверу */  
27         msglen = strlen(MESSAGE);  
28         if (write(sockfd, MESSAGE, msglen) <= msglen)  
29         {  
30             printf(stderr, "Ошибка: запись в FIFO (%d)\n",  
31                    errno);  
32             return 1;  
33         }  
34  
35         sleep(5);  
36  
37         /* закрыть доступ к FIFO */  
38         close(sockfd);  
39  
40         /* закрыть доступ к FIFO */  
41         close(sockfd);  
42  
43         return 0;  
44     }  
45  
46     return 0;  
47 }  
48  
49
```

Рис. 4.1: Текст программы

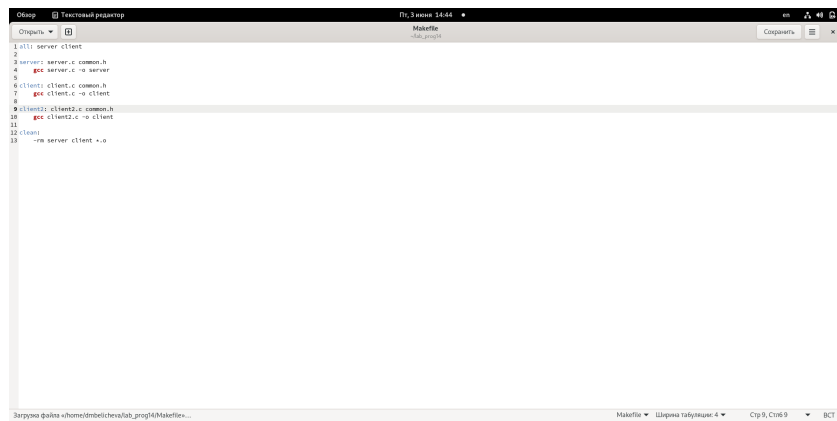
```
Обзор | Текстовый редактор | Пт, 3 июня 16:43 | client.c | lib, project | Сопоставить | X
1 #include "common.h"
2 #include <time.h>
3 #define MESSAGE "Hello Server!!!!!"
4 int
5 main()
6 {
7     int sockfd; /* дескриптор для sockets в FFD */
8     int msglen;
9     long int time;
10
11     for(int i=0; i<10; i++)
12     {
13         time=time(0);
14         printf("time: %d\n", time);
15         /* sleep */
16         printf("sleep... %d\n", 10);
17
18         /* получить доступ к FFD */
19         if((sockfd = open(FFD_NAME, O_WRONLY)) < 0)
20         {
21             printf("Error: %s\n", strerror(errno));
22             return 1;
23         }
24
25         /* создание сообщения */
26         msglen = strlen(MESSAGE);
27         if(write(sockfd, MESSAGE, msglen) != msglen)
28         {
29             printf("Error: %s\n", strerror(errno));
30             return 1;
31         }
32         sleep(1);
33     }
34     close(sockfd);
35     return 0;
36 }
```

Рис. 4.2: Текст программы

3. Сервер работает не бесконечно, а прекращает работу через некоторое время (напри- мер, 30 сек). Используйте функцию `clock()` для определения времени работы сервера. Что будет в случае, если сервер завершит работу, не закрыв канал? (рис. 4.3, 4.4, 4.5, 4.6)

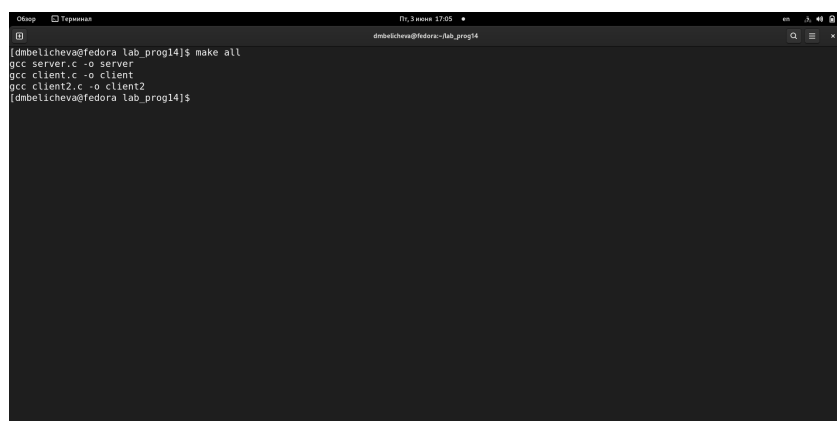
```
Обзор | Текстовый редактор | Пт, 3 июня 16:44 | server.c | lib, project | Сопоставить | X
1 /*
2  * server.c - реализация сервера
3  *
4  * работа: запустить сервер, запустить
5  * 1. запустить программу server на одной машине;
6  * 2. запустить программу client на другой машине.
7  */
8
9 #include "common.h"
10
11 int
12 main()
13 {
14     int sockfd; /* дескриптор для чтения из FFD */
15     int n;
16     char buff[BUFF_SIZE]; /* буфер для чтения данных из FFD */
17
18     /* sleep */
19     printf("Server... %d\n", 10);
20
21     /* создание файла FFD с открытыми для записи
22     * правами доступа на чтение и запись */
23     if((sockfd = open(FFD_NAME, O_RDWR)) < 0)
24     {
25         printf("Error: %s\n", strerror(errno));
26         return 1;
27     }
28
29     /* чтение FFD на чтение */
30     if(read(sockfd, buff, BUFF_SIZE) < 0)
31     {
32         printf("Error: %s\n", strerror(errno));
33         return 1;
34     }
35
36     clock_t begin=time(0);
37     while (begin<time(0))
38     {
39         /* чтение данных из FFD и вывод на экран */
40         while(n = read(sockfd, buff, BUFF_SIZE) > 0)
41         {
42             printf("%s\n", buff);
43         }
44     }
45 }
```

Рис. 4.3: Текст программы



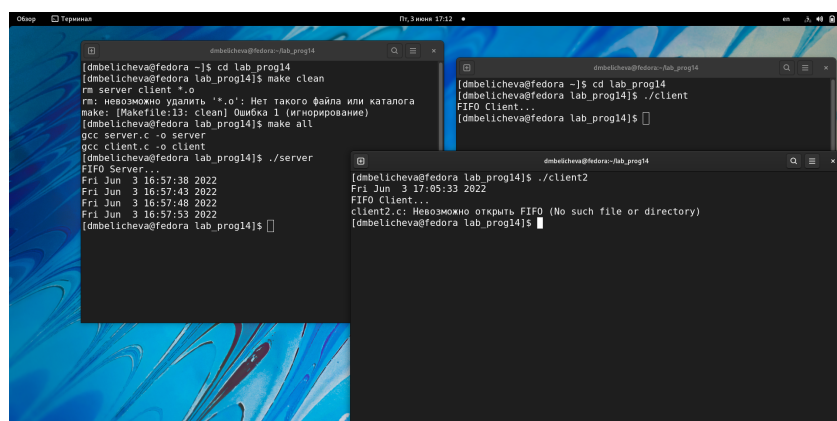
```
1 // server client
2
3 server server.c common.h
4 gcc server.c -o server
5
6 client client.c common.h
7 gcc client.c -o client
8
9 client1 client2.c common.h
10 gcc client2.c -o client1
11
12 clean
13 rm server client *.o
```

Рис. 4.4: Текст программы



```
[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. 4.5: Компиляция



```
[dmbelicheva@fedora ~]$ cd lab_prog14
[dmbelicheva@fedora lab_prog14]$ make clean
rm server client *.o
rm: невозможно удалить '*.o': Нет такого файла или каталога
make: [Makefile:13: clean] Ошибка 1 (игнорирование)
[dmbelicheva@fedora lab_prog14]$ make all
gcc server.c -o server
gcc client.c -o client
[dmbelicheva@fedora lab_prog14]$ ./server
FIFO Server...
Fri Jun 3 16:57:38 2022
Fri Jun 3 16:57:42 2022
Fri Jun 3 16:57:48 2022
Fri Jun 3 16:57:53 2022
[dmbelicheva@fedora lab_prog14]$

[dmbelicheva@fedora ~]$ cd lab_prog14
[dmbelicheva@fedora lab_prog14]$ ./client1
FIFO client...
[dmbelicheva@fedora lab_prog14]$

[dmbelicheva@fedora lab_prog14]$ ./client2
client2.c: Невозможно открыть FIFO (No such file or directory)
[dmbelicheva@fedora lab_prog14]$
```

Рис. 4.6: Результат

5 Выводы

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

6 Контрольные вопросы

1. Именованные каналы отличаются от неименованных наличием идентификатора канала, который представлен как специальный файл (соответственно имя именованного канала — это имя файла).
2. Создание неименованного канала из командной строки возможно командой `pipe`.
3. Создание именованного канала из командной строки возможно с помощью `mkfifo`.
4. 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. 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. 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).

7. Два и более процессов могут читать и записывать в канал.
8. 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`.
9. 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

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