

Презентация по лабораторной работе №13

Дисциплина: Операционные системы

Шошина Е.А.

9 мая 2023

Российский университет дружбы народов, Москва, Россия

Информация

- Шошина Евгения Александровна
- группа: НКАбд-03-22
- студент факультета физико-математических и естественных наук
- Российский университет дружбы народов
- 1132229532@pfur.ru
- <https://EAShoshina.github.io/ru/>



Приобретение практических навыков работы с именованными каналами.

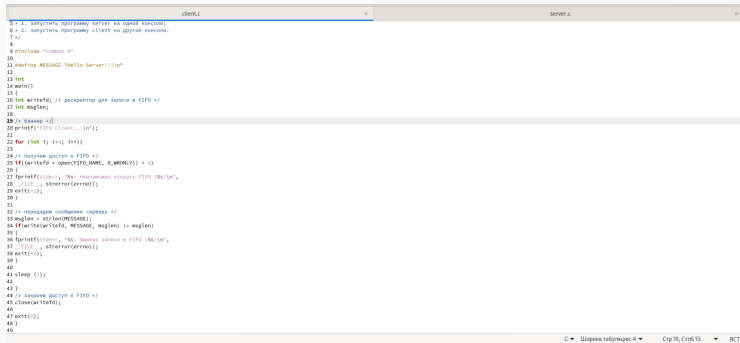
Задание: Изучите приведённые в тексте программы `server.c` и `client.c`. Взяв данные примеры за образец, напишите аналогичные программы, внося следующие изменения:

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?

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: общедюкские (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)`.

Выполнение лабораторной работы

Изучили приведённые в тексте программы server.c и client.c. Взяв данные примеры за образец, написали аналогичные программы, внося следующие изменения: 1. Работает не 1 клиент, а несколько (например, два).(рис. (fig:001?)).



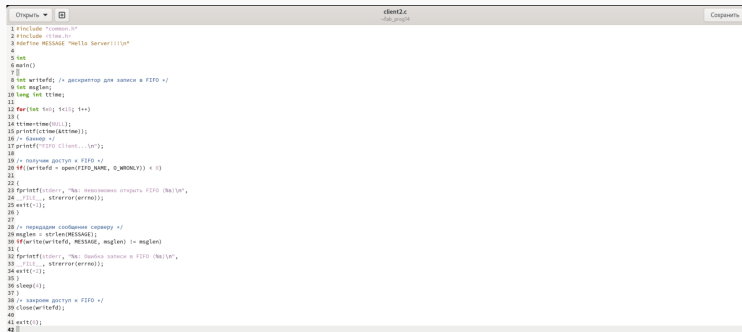
```
client.c
5 * 1. запустить программу server на одной консоли;
6 * 2. запустить программу client на другой консоли.
7 */
8
9 #include "common.h"
10
11 #define MESSAGE "Hello Server!!!"
12
13 int
14 main()
15 {
16     int writefd; /* дескриптор для записи в FIFO */
17     int maglen;
18
19     /* Завершаем */
20     printf("FIFO client...\n");
21
22     for (int i; i < 10; i++){
23
24         /* получаем доступ к FIFO */
25         if((writefd = open(FIFO_NAME, O_WRONLY)) < 0)
26         {
27             perror("Ошибка: невозможно открыть FIFO (%s)\n",
28                  _FILE_, strerror(errno));
29             exit(-1);
30         }
31
32         /* передаем сообщение серверу */
33         maglen = strlen(MESSAGE);
34         if(write(writefd, MESSAGE, maglen) != maglen)
35         {
36             perror("Ошибка: запись в FIFO (%s)\n",
37                  _FILE_, strerror(errno));
38             exit(-1);
39         }
40
41         sleep (1);
42     }
43
44     /* закрываем доступ к FIFO */
45     close(writefd);
46
47     exit(0);
48 }
49
```

server.c

Стр 19, Стр 13 ВСТ

Рис. 1: текст программы

2. Клиенты передают текущее время с некоторой периодичностью (например, раз в пять секунд). Используйте функцию `sleep()` для приостановки работы клиента.(рис. (fig:002?)).



```
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<5; i++)
13     {
14         time_t(ttime);
15         printf("ctime(%ttime)\n");
16         /* Sleep */
17         printf("FIFO Client...\n");
18
19         /* получить доступ к FIFO */
20         if((writefd = open(FIFO_NAME, O_WRONLY)) < 0)
21         {
22             fprintf(stderr, "Ошибка: 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)
30         {
31             fprintf(stderr, "Ошибка: запись в FIFO (%s)\n",
32                     _FILE_, strerror(errno));
33             exit(-1);
34         }
35
36         sleep(5);
37     }
38     /* закрыть 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?)). (рис. (fig:004?)).(рис. (fig:005?)).

client.c	server.c
<pre>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 /* Запуск */ 19 printf("FIFO Server...\n"); 20 21 /* создаем файл FIFO с открытыми для всех 22 * правами доступа на чтение и запись 23 */ 24 if(mkfifo(FIFO_NAME, S_IFIFO 0666, 0) < 0) 25 { 26 fprintf(stderr, "На: Невозможно создать FIFO (%s)\n", 27 _FILE_, strerror(errno)); 28 exit(-1); 29 } 30 31 /* открываем FIFO на чтение */ 32 if((readfd = open(FIFO_NAME, O_RDONLY)) < 0) 33 { 34 fprintf(stderr, "На: 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_time < now_time) 41 { 42 /* чтение данных из FIFO и выводим на экран */ 43 while((n = read(readfd, buff, MAX_BUFF)) > 0) 44 { 45 if(write(1, buff, n) < n)</pre>	

Рис. 3: 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?)). (рис. (fig:004?)).(рис. (fig:005?)).

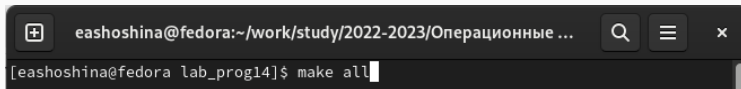


The image shows a screenshot of a text editor window titled "Makefile" with the path "~/lab_prog14". The window contains the following Makefile content:

```
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

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?)). (рис. (fig:004?)).(рис. (fig:005?)).

A terminal window with a dark background. The title bar shows the user 'eashoshina@fedora' and the current directory '~/work/study/2022-2023/Операционные ...'. The terminal prompt is '[eashoshina@fedora lab_prog14]\$' and the command 'make all' has been entered, with a cursor at the end of the line.

```
eashoshina@fedora:~/work/study/2022-2023/Операционные ...  
[eashoshina@fedora lab_prog14]$ make all
```

Рис. 5: Компиляция

Приобрели практические навыки работы с именованными каналами.

Контрольные вопросы

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);`

Контрольные вопросы

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=EP1PE`) (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