

Отчет по лабораторной работе №14

Дисциплина: Операционные системы

Шошина Евгения Александровна (НКАбд-03-22)

Содержание

1	Цель работы	5
2	Задание	6
3	Теоретическое введение	7
4	Выполнение лабораторной работы	8
5	Выводы	11
6	Контрольные вопросы	12

Список иллюстраций

4.1	текст программы	8
4.2	Текст программы	9
4.3	Текст программы	9
4.4	Текст программы	10
4.5	Компиляция	10

Список таблиц

1 Цель работы

Приобретение практических навыков работы с именованными каналами.

2 Задание

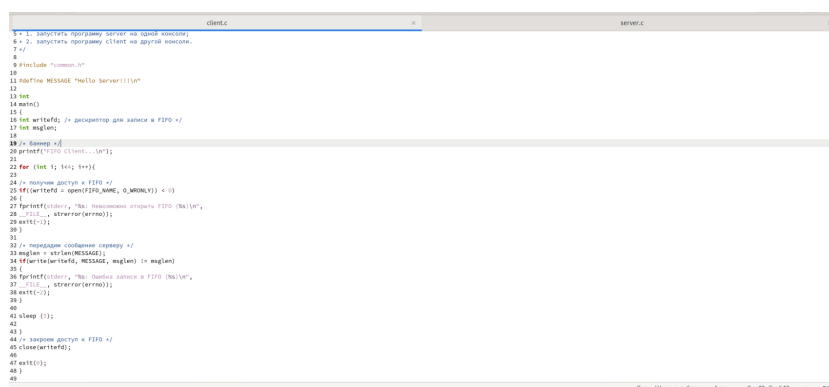
Изучите приведённые в тексте программы `server.c` и `client.c`. Взяв данные примеры за образец, напишите аналогичные программы, внося следующие изменения: 1. Работает не 1 клиент, а несколько (например, два). 2. Клиенты передают текущее время с некоторой периодичностью (например, раз в пять секунд).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

4 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. 4.1).



```
client.c
1 // 1. запустить программу server на своей машине.
2 // 2. запустить программу client на другой машине.
3
4
5 #include "common.h"
6
7 // Define MESSAGE "Hello Server!!!"
8
9 int
10 int
11 int
12 int
13 int
14 int
15 int
16 int
17 int
18 int
19 int
20 int
21 int
22 int
23 int
24 int
25 int
26 int
27 int
28 int
29 int
30 int
31 int
32 int
33 int
34 int
35 int
36 int
37 int
38 int
39 int
40 int
41 int
42 int
43 int
44 int
45 int
46 int
47 int
48 int
49 int
```

Рис. 4.1: текст программы

2. Клиенты передают текущее время с некоторой периодичностью (например, раз в пять секунд). Используйте функцию sleep() для приостановки работы клиента.(рис. 4.2).



```
Открыть  client.c Сохранить
1 #include <unistd.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 mglen;
10    long int ttime;
11
12    for(int i=0; i<10; i++)
13    {
14        gettimeofday(&ttime);
15        printf("ttime\n");
16        /* sleep */
17        printf("FIFO Client...\n");
18
19        /* получаем дескриптор FIFO */
20        if(writefd = open(FIFO_NAME, O_WRONLY) < 0)
21        {
22            printf(stderr, "Невозможно открыть FIFO (%s)\n",
23                  strerror(errno));
24            exit(-1);
25        }
26
27        /* передаем сообщение серверу */
28        mglen = strlen(MESSAGE);
29        if(write(writefd, MESSAGE, mglen) != mglen)
30        {
31            printf(stderr, "Не удалось записать в FIFO (%s)\n",
32                  strerror(errno));
33            exit(-1);
34        }
35        sleep(1);
36        close(writefd);
37        /* закрываем дескриптор FIFO */
38        close(writefd);
39    }
40    exit(0);
41 }
```

Рис. 4.2: 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. 4.3). (рис. 4.4).(рис. 4.5).

```
client.c x server.c
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 <unistd.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     /* sleep */
19     printf("FIFO Server...\n");
20
21     /* создаем файл FIFO с открытием для всех
22     * с правами доступа на чтение и запись.
23     */
24     if(mkfifo(FIFO_NAME, S_IFIFO | 0666, 0) < 0)
25     {
26         printf(stderr, "Невозможно создать FIFO (%s)\n",
27               strerror(errno));
28         exit(-1);
29     }
30
31     /* открываем FIFO на чтение */
32     if(readfd = open(FIFO_NAME, O_RDONLY) < 0)
33     {
34         printf(stderr, "Невозможно открыть FIFO (%s)\n",
35               strerror(errno));
36         exit(-1);
37     }
38
39     clock_t beginning_time(0); clock_t now_time(0);
40     while (beginning_time < now_time)
41     {
42         /* чтение данных из FIFO и выводе на экран */
43         while(n = read(readfd, buff, MAX_BUFF) > 0)
44         {
45             if(write(1, buff, n) < n)
46             {
47                 printf(stderr, "Невозможно записать в stdout (%s)\n",
48                       strerror(errno));
49                 exit(-1);
50             }
51         }
52     }
53 }
```

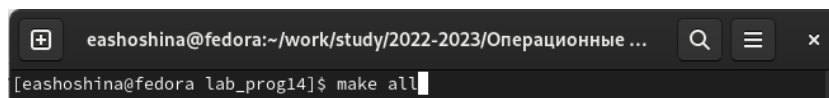
Рис. 4.3: Текст программы



The image shows a text editor window with a title bar containing "Открыть" (Open) and a file icon. The file name is "Makefile" and the path is "~/lab_prog14". The content of the Makefile is as follows:

```
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. 4.4: Текст программы



The image shows a terminal window with a title bar containing a plus icon, the user and host information "eashoshina@fedora:~/work/study/2022-2023/Операционные ...", and search, menu, and close icons. The terminal prompt is "[eashoshina@fedora lab_prog14]\$". The command "make all" has been entered, and the cursor is at the end of the line.

Рис. 4.5: Компиляция

5 Выводы

Приобрели 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, пор-

ции 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 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 сообщении об ошибке, понятном человеку.