

Отчет по лабораторной работе №14

Именованные каналы

Кочкарев “sakochkarev” Станислав

Содержание

1	Цель работы	3
2	Задание	4
3	Выполнение лабораторной работы	5
4	Выводы	12
5	Контрольные вопросы	13

1 Цель работы

Приобретение практических навыков работы с именованными каналами.

2 Задание

Написать программы по примеру приведенных со следующими изменениями:

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 Выполнение лабораторной работы

Первым делом мы взяли за основу приведенные тексты программ и создали файлы с данными текстами (рис. 3.1, 3.2, 3.3).

```
common.h x server.c x client.c x
1 //
2 // #ifndef LAB14_COMMON_H
3 // #define LAB14_COMMON_H
4 //
5 // #endif // LAB14_COMMON_H
6 //
7 /*
8  * common.h - заголовочный файл со стандартными определениями
9  */
10 #ifndef __COMMON_H__
11 #define __COMMON_H__
12
13 #include <stdio.h>
14 #include <stdlib.h>
15 #include <string.h>
16 #include <errno.h>
17 #include <sys/types.h>
18 #include <sys/stat.h>
19 #include <fcntl.h>
20 #include <unistd.h>
21
22 #define FIFO_NAME "/Users/squidass/tmp/fifo"
23 #define MAX_BUFF 80
```

Рис. 3.1: Файл программы common.h

```

common.h x server.c x client.c x
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  #include "common.h"
9  #include "time.h"
10
11 int main() {
12     int readfd; /* дескриптор для чтения из FIFO */
13     int n;
14     char buff[MAX_BUFF]; /* буфер для чтения данных из FIFO */
15     /* баннер */
16     printf("FIFO Server...\n");
17     /* создаем файл FIFO с открытыми для всех
18     * правами доступа на чтение и запись
19     */
20
21     if (mkfifo(FIFO_NAME, 0666) < 0) {
22         fprintf(stderr, "%s: Невозможно создать FIFO (%s)\n", __FILE__, strerror( errno));
23         exit(-1);
24     }
25
26     /* откроем FIFO на чтение */
27     if ((readfd = open(FIFO_NAME, O_RDONLY)) < 0) {
28         fprintf(stderr, "%s: Невозможно открыть FIFO (%s)\n", __FILE__, strerror( errno));
29         exit(-2);
30     }
31
32     clock_t start_time = time(NULL);
33
34     /* читаем данные из FIFO и выводим на экран */
35     while ((n = read(readfd, buff, MAX_BUFF)) > 0 && time(NULL) - start_time < 30) {

```

Рис. 3.2: Файл программы server.c

```

common.h x server.c x client.c x
1  /*
2  * client.c -- реализация клиента
3  *
4  * Чтобы запустить пример, необходимо:
5  * 1. запустить программу server на одной консоли;
6  * 2. запустить программу client на другой консоли.
7  */
8  #include "common.h"
9  #include "time.h"
10
11 int main() {
12     int writefd; /* дескриптор для записи в FIFO */ int msglen;
13     /* баннер */
14     printf("FIFO Client...\n");
15     /* получим доступ к FIFO */
16     if ((writefd = open(FIFO_NAME, O_WRONLY)) < 0) {
17         fprintf(stderr, "%s: Невозможно открыть FIFO (%s)\n",
18             __FILE__, strerror( errno: errno));
19         exit(-1);
20     }
21
22     while (1) {
23         /* передадим сообщение серверу */
24         time_t rawtime;
25         struct tm *timeinfo;
26         time(&rawtime);
27         timeinfo = localtime(&rawtime);
28         char *message = asctime(timeinfo);
29         msglen = strlen( s: message);
30         if (write( fd: writefd, buf: message, nbyte: msglen) != msglen) {
31             fprintf(stderr, "%s: Ошибка записи в FIFO (%s)\n",
32                 __FILE__, strerror( errno: errno));
33             exit(-2);
34         }
35         // fprintf(stdout, "test %ld", clock());

```

Рис. 3.3: Файл программы client.c

После этого мы начали работать с программой client.c.

Там мы добавили второе приведенное изменение – отправку текущего времени

(timestamp) каждые 5 секунд (рис. 3.4). Для приостановки работы клиента была использована функция `sleep()`.

```
11 int main() {
12     int writefd; /* дескриптор для записи в FIFO */ int msglen;
13     /* баннер */
14     printf("FIFO-Client...\n");
15     /* получим доступ к FIFO */
16     if ((writefd = open(FIFO_NAME, O_WRONLY)) < 0) {
17         fprintf(stderr, "%s: Невозможно открыть FIFO (%s)\n",
18             __FILE__, strerror( errno: errno));
19         exit(-1);
20     }
21
22     while (1) {
23         /* передадим сообщение серверу */
24         time_t rawtime;
25         struct tm *timeinfo;
26         time(&rawtime);
27         timeinfo = localtime(&rawtime);
28         char *message = asctime(timeinfo);
29         msglen = strlen( s: message);
30         if (write( fd: writefd, buf: message, nbyte: msglen) != msglen) {
31             fprintf(stderr, "%s: Ошибка записи в FIFO (%s)\n",
32                 __FILE__, strerror( errno: errno));
33             exit(-2);
34         }
35         // fprintf(stdout, "test %ld", clock());
36         sleep(5);
37     }
38     /* закроем доступ к FIFO */
39     close(writefd);
40     exit(0);
41 }
```

Рис. 3.4: Измененная главная функция

Далее были произведены изменения в файле `server.c`. В нем были добавлены изменения из 3-его задания – прекращение работы по истечению 30 секунд. Для этого была использована функция `time()`. Итоговый файл приведен на (рис. 3.5).

```

21 if (mkfifo(FIFO_NAME, 0666) < 0) {
22     fprintf(stderr, "%s: Невозможно создать FIFO (%s)\n", __FILE__, strerror( errno: errno));
23     exit(-1);
24 }
25
26 /* откроем FIFO на чтение */
27 if ((readfd = open(FIFO_NAME, O_RDONLY)) < 0) {
28     fprintf(stderr, "%s: Невозможно открыть FIFO (%s)\n", __FILE__, strerror( errno: errno));
29     exit(-2);
30 }
31
32 clock_t start_time = time(NULL);
33
34 /* читаем данные из FIFO и выводим на экран */
35 while ((n = read(readfd, buff, MAX_BUFF)) > 0 & time(NULL) - start_time < 30) {
36     if (write(fd: 1, buf: buff, nbyte: n) != n) {
37         fprintf(stderr, "%s: Ошибка вывода (%s)\n", __FILE__, strerror( errno: errno));
38         exit(-3);
39     }
40 }
41 close(readfd); /* закроем FIFO */
42 /* удалим FIFO из системы */
43 if (unlink(FIFO_NAME) < 0) {
44     fprintf(stderr, "%s: Невозможно удалить FIFO (%s)\n", __FILE__, strerror( errno: errno));
45     exit(-4);
46 }
47 exit(0);
48 }

```

Рис. 3.5: Измененная часть server.c

Во время работы над файлами изменения были протестированы.

В результате этих тестов было выяснено, что если сервер завершит работу не закрыв канал, то файл канала останется и при следующем запуске сервера он выдаст ошибку (рис. 3.6).

```

sakochkarev@sakochkarev [21:07:10] [~/work/study/2021-2022/Операционные системы/os-intro/labs/lab14] [master *]
-> % ./server
FIFO Server...
server.c: Невозможно создать FIFO (File exists)

```

Рис. 3.6: Ошибка при некорректном завершении

После выполнения всех заданий были произведены финальные тесты, которые подтвердили работоспособность и корректность выполнения всех заданий (рис. 3.7) (рис. 3.8).

```
sakochkarev@sakochkarev [21:14:04] [~/work/study/2021-2022/Операционные системы/os-intro/labs/lab14] [master *]
-> % ./server
FIFO Server...
Mon May 30 21:16:41 2022
Mon May 30 21:16:43 2022
Mon May 30 21:16:46 2022
Mon May 30 21:16:48 2022
Mon May 30 21:16:51 2022
Mon May 30 21:16:53 2022
Mon May 30 21:16:56 2022
Mon May 30 21:16:58 2022
Mon May 30 21:17:01 2022
Mon May 30 21:17:03 2022
Mon May 30 21:17:06 2022
Mon May 30 21:17:08 2022
sakochkarev@sakochkarev [13:38:35] [~/work/study/2021-2022/Операционные системы/os-intro/labs/lab14] [master *]
```

Рис. 3.7: Запуск и вывод сервера

```
sakochkarev@sakochkarev [21:14:09] [~/work/study/2021-2022/Операционные системы/os-intro/labs/lab14] [master *]
-> % ./client
FIFO Client...
sakochkarev@sakochkarev [13:39:16] [~/work/study/2021-2022/Операционные системы/os-intro/labs/lab14] [master *]
```

Рис. 3.8: Запуск и вывод одного из клиентов

4 Выводы

По выполнении данной лабораторной работы мы приобрели практические навыки работы с именованными каналами.

5 Контрольные вопросы

1. Именованные каналы, в отличие от неименованных, могут использоваться неродственными процессами.
2. Нет.
3. Да, используя `mkfifo`.
4. `int pipe(int pipefd[2]);` создаёт однонаправленный канал данных, который можно использовать для взаимодействия между процессами.
5. Функция `int mkfifo(const char *pathname, mode_t mode);` создает специальный FIFO файл с именем `pathname`, а параметр `mode` задает права доступа к файлу.
6. При прочтении меньшего числа байтов из `fifo` сообщение просто выводится не полностью.

Прочтение большего числа байтов дает уже более заметный эффект – не находя достаточного количества информации в `fifo`, программа обращается к другим участкам памяти и выводит уже их, тем самым печатая информацию из других файлов.
7. Запись некорректного числа байтов непозволительна, поэтому вызов `write()` блокируется до выполнения корректной последовательности.
8. Да, два или более процессов могут одновременно записывать в один канал.
- 9.

```
ssize_t write(int fd, const void *buf, size_t count);
```

ssize_t – тип возвращаемого значения
int fd, const void *buf, size_t count – аргументы
Записывает байты в указанный буффер. 1 на месте аргумента fd – указание файлового дескриптора.

10. Функция stderror находит сообщение об ошибке, соответствующее приведенному номеру ошибки.