

Презентация по лабораторной работе №14

Операционные системы

Сячинова Ксения Ивановна

14 апреля 2023

Российский университет дружбы народов, Москва, Россия

Цель работы

Приобретение практических навыков работы с именованными каналами

Задание

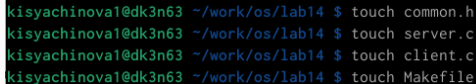
Изучить приведённые в тексте программы `server.c` и `client.c`. Взяв данные примеры за образец, напишите аналогичные программы, внося следующие изменения:

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?

Выполнение лабораторной работы

Для начала изучили материал лабораторной работы. Далее на основе примеров напишем аналогичные программы, но с изменениями.

1. Для начала создадим необходимые файлы для работы.



```
kisyachinova1@dk3n63 ~/work/os/lab14 $ touch common.h
kisyachinova1@dk3n63 ~/work/os/lab14 $ touch server.c
kisyachinova1@dk3n63 ~/work/os/lab14 $ touch client.c
kisyachinova1@dk3n63 ~/work/os/lab14 $ touch Makefile
```

Рис. 1: Создание файлов

Затем изменим коды программ, данных в лабораторной работе. В файл *common.h* добавим стандартные заголовочные файлы: “unistd.h”, “time.h”. Это необходимо для работы других файлов. Этот файл является заголовочным, чтобы в остальных программах не прописывать одно и то же каждый раз.

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

Рис. 2: Файл “common.h”

2. Затем в файл `server.c` добавляем цикл “while” для контроля за временем работы сервера. Разница между текущим временем и началом работы не должна превышать 30 секунд.

```
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     /* создаем 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     /* откроем 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(-2);
37     }
```

Рис. 3: Файл “server.c”

```

36     exit(-2);
37 }
38 /*начало отсчёта времени*/
39 clock_t start = time(NULL);
40
41 /*цикл работат пока с момента начала отсчёта времени прошло меньше 30 секунд*/
42 while(time(NULL)-start < 30)
43 {
44     /* читаем данные из FIFO и выводим на экран */
45     while((n = read(readfd, buff, MAX_BUFF)) > 0)
46     {
47         if(write(1, buff, n) != n)
48         {
49             fprintf(stderr, "%s: Ошибка вывода (%s)\n",
50                 __FILE__, strerror(errno));
51             exit(-3);
52         }
53     }
54 }
55 close(readfd); /* закроем FIFO */
56
57 /* удалим FIFO из системы */
58 if(unlink(FIFO_NAME) < 0)
59 {
60     fprintf(stderr, "%s: Невозможно удалить FIFO (%s)\n",
61         __FILE__, strerror(errno));
62     exit(-4);
63 }
64 exit(0);
65 }

```

Рис. 4: Файл "server.c"

3. В файл *client.c* добавим цикл, который отвечает за количество сообщений о текущем времени (4 сообщения). С помощью команды “sleep” приостановим работу клиента на 5 секунд.рис.

```
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
9 #include "common.h"
10
11 #define MESSAGE "Hello Server!!!\n"
12
13 int main()
14 {
15     int writefd; /* дескриптор для записи в FIFO */
16     int nsiglen;
17
18     /* баннер */
19     printf("FIFO Client...\n");
20
21     /* цикл, отвечающий за отправку сообщения о текущем времени */
22     for(int i=0; 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             break;
31         }
```

Рис. 5: Файл “client.c”

```

31     }
32
33     /*текущее время */
34     long int ttime=time(NULL);
35     char* text=ctime(&ttime);
36
37     /* передадим сообщение серверу */
38     msglen = strlen(MESSAGE);
39     if(write(writefd, MESSAGE, msglen) != msglen)
40     {
41         fprintf(stderr, "%s: Ошибка записи в FIFO (%s)\n",
42             __FILE__, strerror(errno));
43         exit(-2);
44     }
45     /*приостановка работы клиента на 5 секунд*/
46     sleep(5);
47 }
48
49 /* закроем доступ к FIFO */
50 close(writefd);
51 exit(0);
52 }

```

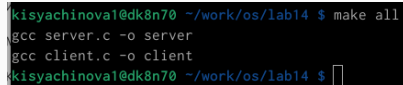
Рис. 6: Файл “client.c”

Makefile оставили без изменений.

```
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 clean:
10        -rm server client *.o
```

Рис. 7: Файл “Makefile”

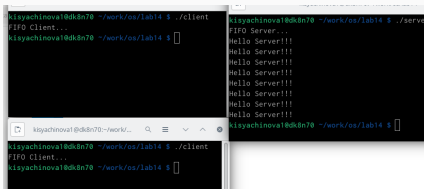
Далее делаем компиляцию файлов с помощью команды “make all”.

A terminal window with a black background and green text. The prompt is 'kisyachinova1@dk8n70 ~/work/os/lab14 \$'. The command 'make all' has been entered and executed. The output shows two lines: 'gcc server.c -o server' and 'gcc client.c -o client'. The prompt is now followed by a cursor.

```
kisyachinova1@dk8n70 ~/work/os/lab14 $ make all
gcc server.c -o server
gcc client.c -o client
kisyachinova1@dk8n70 ~/work/os/lab14 $
```

Рис. 8: Компиляция

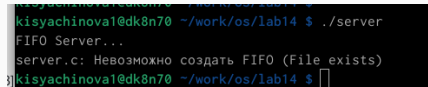
Затем открываем три терминала для проверки работы наших файлов. В первом пишем “./server”, а в остальных “./client”. В результате каждый терминал вывел по 4 сообщения, а по истечению 30 секунд работа сервера была завершена. Всё работает верно.



The image shows three terminal windows from the user kisyachinova1@dk8n70. The top-left window runs './client' and outputs 'FIFO Client...'. The top-right window runs './server' and outputs 'FIFO Server...' followed by eight 'Hello Server!!!' messages. The bottom-left window also runs './client' and outputs 'FIFO Client...'. The terminal windows are arranged in a grid-like fashion, with the server window at the top right and two client windows below it.

Рис. 9: Проверка

Проверим длительность работы сервера. Вводим команду “./server” в одном терминале. Он завершил свою работу через 30 секунд. Если сервер завершит свою работу, не закрывая канал, то при повторном запуске появится ошибка “Невозможно создать FIFO”, так как у нас уже есть один канал.



```
kisyachinova1@dk8n70 ~/work/os/lab14 $ ./server
FIFO Server...
server.c: Невозможно создать FIFO (File exists)
kisyachinova1@dk8n70 ~/work/os/lab14 $
```

Рис. 10: Проверка

Выводы

В ходе выполнения данной лабораторной работы я приобрела навыки работы с очередями сообщений.