

# Презентация по лабораторной работе №14

---

Сячинова Ксения Ивановна, НПМбд-02-21

Российский университет дружбы народов

## Цель работы

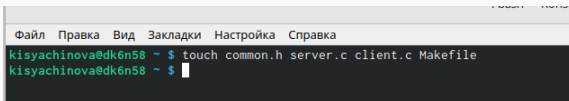
---

Приобретение практических навыков работы с именованными каналами.

## Выполнение лабораторной работы

---

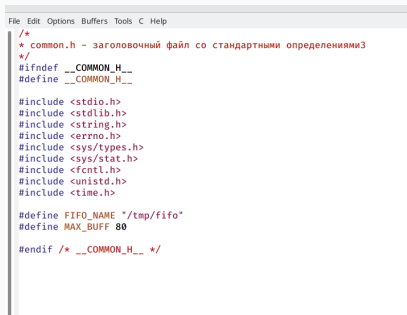
1. Для начала изучим приведённые в тексте программы `server.c` и `client.c`. Взяв данные примеры за образец, напишем аналогичные программы, с некоторыми изменениями. Первое изменение: работает не 1 клиент, а несколько (например, два). Сперва создадим все необходимые файлы для лабораторной работы.(рис. 1)



```
Файл  Правка  Вид  Закладки  Настройка  Справка
kisyachinova@dk6n58 ~ $ touch common.h server.c client.c Makefile
kisyachinova@dk6n58 ~ $
```

Figure 1: Создание файлов

Изменим коды программ, которые даны нам в лабораторной работе. В файл `common.h` добавим стандартные заголовочные файлы: `"unistd.h"`, `"time.h"`. Они необходимы для работы других кодов файлов. Данный файл создан в качестве заголовочного файла, чтобы в остальных программах их не прописывать каждый раз.(рис. 2)

A screenshot of a code editor window with a menu bar (File, Edit, Options, Buffers, Tools, C, Help) and a text area containing C header file code. The code defines \_\_COMMON\_H\_\_ and includes various standard headers like <stdio.h>, <stdlib.h>, <string.h>, <errno.h>, <sys/types.h>, <sys/stat.h>, <fcntl.h>, <unistd.h>, and <time.h>. It also defines FIFO\_NAME and MAX\_BUFF.

```
File Edit Options Buffers Tools C Help

/*
 * common.h - заголовочный файл со стандартными определениями
 */
#ifndef __COMMON_H__
#define __COMMON_H__

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <errno.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <unistd.h>
#include <time.h>

#define FIFO_NAME "/tmp/fifo"
#define MAX_BUFF 80

#endif /* __COMMON_H__ */
```

Figure 2: Файл `"common.h"`

2. 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 (time(NULL) и clock\_t start=time(NULL)).(рис. 3), (рис. 4)

```
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 main()
12 {
13     int readfd; /* дескриптор для чтения из FIFO */
14     int n;
15     char buff[MAX_BUFF]; /* буфер для чтения данных из FIFO */
16
17     /* баннер */
18     printf("FIFO Server...\n");
19
20     /* создаем файл FIFO с открытыми для всех
21      * правами доступа на чтение и запись
22      */
23     if(mknod(FIFO_NAME, S_IFIFO | 0666, 0) < 0)
24     {
25         fprintf(stderr, "Невозможно создать FIFO (%s)\n",
26                 _FILE_, strerror(errno));
27         exit(-1);
28     }
29
30     /* откром FIFO на чтение */
31     if((readfd = open(FIFO_NAME, O_RDONLY)) < 0)
32     {
33         fprintf(stderr, "Невозможно открыть FIFO (%s)\n",
34                 _FILE_, strerror(errno));
35         exit(-1);
36     }
```

Figure 3: Файл “server.c”

```

35     exit(-1);
36 }
37
38 /*начало отсчёта времени*/
39 clock_t start = time(NULL);
40
41 /*чекя работает, 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(-1);
52         }
53     }
54 }
55
56 close(readfd); /* закроем FIFO */
57
58 /* удалим FIFO из системы */
59 if(unlink(FIFO_NAME) < 0)
60 {
61     fprintf(stderr, "%s: Невозможно удалить FIFO (%s)\n",
62             _FILE_, strerror(errno));
63     exit(-1);
64 }
65 exit(0);
66 }

```

Figure 4: Файл “server.c”



3. В файл `client.c` добавляем цикл, который отвечает за количество сообщений о текущем времени (4 сообщения). С помощью команды `“sleep(5)”` приостанавливаем работу клиента на 5 секунд. (рис. 5), (рис. 6)

```
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
14 main()
15 {
16     int writefd; /* дескриптор для записи в FIFO */
17     int readfd;
18
19     /* баннер */
20     printf("FIFO Client...\n");
21
22     /* цикл, отвечающий за отправку сообщений о текущем времени */
23     for(int i=0; i<4; i++)
24     {
25         /* получаем доступ к FIFO */
26         if((writefd = open(FIFO_NAME, O_WRONLY)) < 0)
27         {
28             fprintf(stderr, "%s: невозможно открыть FIFO (%s)\n",
29                     __FILE__, strerror(errno));
30
31             exit(-1);
32             break;
33         }
34     }
35
36     /* текущее время */
37     long int ttime(time(NULL));
38     char* testtime(ttime);
39 }
```

Figure 5: Файл “client.c”

```

36      /* текущее время */
37      long int time=time(NULL);
38      char* text=time(&time);
39
40      /* передаем сообщение серверу */
41      msglen = strlen(MESSAGE);
42      if(write(writefd, MESSAGE, msglen) != msglen)
43      {
44          fprintf(stderr, "%s: Ошибка записи в FIFO (%s)\n",
45                  __FILE__, strerror(errno));
46          exit(-1);
47      }
48      /* приостановка работы клиента на 5 секунд */
49      sleep(5);
50  }
51  /* закрываем доступ к FIFO */
52  close(writefd);
53  exit(0);
54  }

```

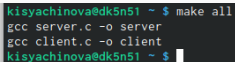
Figure 6: Файл “client.c”

Makefile оставили без изменений. (рис. 7)

```
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
```

Figure 7: Makefile

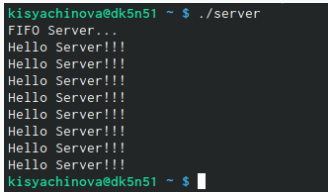
Используем команду “make all” для компиляции необходимых файлов.(рис. 8)

A terminal window with a dark background and light green text. The prompt is 'kisyachinova@dk5n51 ~ \$'. The command 'make all' has been entered and executed. The output shows two compilation commands: 'gcc server.c -o server' and 'gcc client.c -o client'. The prompt is now 'kisyachinova@dk5n51 ~ \$' followed by a white cursor bar.

```
kisyachinova@dk5n51 ~ $ make all
gcc server.c -o server
gcc client.c -o client
kisyachinova@dk5n51 ~ $
```

Figure 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. (рис. 9), (рис. 10), (рис. 11)

A terminal window with a dark background and green text. The prompt is 'kisyachinova@dk5n51 ~ \$'. The command './server' has been executed. The output consists of eight lines: 'FIFO Server...', followed by seven 'Hello Server!!!' messages. The prompt is now 'kisyachinova@dk5n51 ~ \$' with a cursor.

```
kisyachinova@dk5n51 ~ $ ./server
FIFO Server...
Hello Server!!!
Hello Server!!!
Hello Server!!!
Hello Server!!!
Hello Server!!!
Hello Server!!!
Hello Server!!!
kisyachinova@dk5n51 ~ $
```

Figure 9: Server

```
kisyachinova@dk5n51 ~ $ ./client  
FIFO Client...  
kisyachinova@dk5n51 ~ $ █
```

Figure 10: Client 1

```
kisyachinova@dk5n51 ~ $ ./client  
FIFO Client...  
kisyachinova@dk5n51 ~ $ █
```

Figure 11: Client 2

## Выводы

---

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