

Laboratory №15

Kulikov Maksim

MAY-2021

RUDN University, Moscow, Russian Federation

Приобретение практических навыков работы с именованными каналами.

1. Ознакомиться с теоретическим материалом.
2. Выполнить работу.

1. Создаю в новом каталоге 4 скрипта (рис. -fig. 1)

```
b15$ cd prog  
b15/prog$ touch common.h server.c client.c Makefile  
b15/prog$ emacs &
```

Рис. 1: Создание файлов

2. Содержимое 4-х скриптов. В файл `common.h` добавил стандартные заголовочные файлы `unistd.h` и `time.h`, необходимые для работы кодов других файлов. `Common.h` предназначен для заголовочных файлов, чтобы в остальных программах их не прописывать каждый раз. В файл `server.c` добавил цикл `while` для контроля за временем работы сервера. Разница между текущим временем `time(NULL)` и временем начала работы `clock_t start=time(NULL)` (инициализация до цикла) не должна превышать 30 секунд. В файл `client.c` добавил цикл, который отвечает за количество сообщений о текущем времени (4 сообщения). `Makefile` я оставил без изменения (рис. -fig. 2)

```

client.c - реализация клиента
*
* чтобы запустить пример, необходимо:
* 1. запустить программу server на одной консоли;
* 2. запустить программу client на другой консоли.
*/

#include "common.h"

int
main()
{
    int writefd; /* дескриптор для записи в FIFO */
    int msglen;

    /* Banner */
    printf("FIFO Client...\n");

    for (int i=0; i<4; i++)
    {
        /* проверяем доступ к FIFO */
        if ((writefd = open(FIFO_NAME, O_WRONLY) < 0)
        {
            fprintf(stderr, "ks: невозможно открыть FIFO (%s)\n",
                    _FILE_, strerror(errno));
            exit(-1);
        }

        long int ttimeutime(NULL);
        char* textctime(&ttime);

        /* передаем сообщение серверу */
        msglen = strlen(text);
        if (write(writefd, text, msglen) != msglen)
        {
            fprintf(stderr, "ks: невозможно записать в FIFO (%s)\n",
                    _FILE_, strerror(errno));
            exit(-1);
        }
    }

    return 0;
}

server.c - реализация сервера
*
* чтобы запустить пример, необходимо:
* 1. запустить программу server на одной консоли;
* 2. запустить программу client на другой консоли.
*/

#include "common.h"

int
main()
{
    int readfd; /* дескриптор для чтения из FIFO */
    int i;
    char buff[MAX_BUFF]; /* буфер для чтения данных из FIFO */

    /* Banner */
    printf("FIFO Server...\n");

    /* создаем файл FIFO с открытием для всех
    * прав доступа на чтение и запись
    */
    if (mkfifo(FIFO_NAME, S_IFIFO | 0666, 0) < 0)
    {
        fprintf(stderr, "ks: невозможно создать FIFO (%s)\n",
                _FILE_, strerror(errno));
        exit(-1);
    }

    /* отключаем FIFO на чтение */
    if ((readfd = open(FIFO_NAME, O_RDONLY | O_NONBLOCK) < 0)
    {
        fprintf(stderr, "ks: невозможно открыть FIFO (%s)\n",
                _FILE_, strerror(errno));
        exit(-1);
    }

    while (1)
    {
        /* читаем данные из FIFO */
        if (read(readfd, buff, MAX_BUFF) < 0)
        {
            fprintf(stderr, "ks: невозможно прочитать из FIFO (%s)\n",
                    _FILE_, strerror(errno));
            exit(-1);
        }

        /* выводим данные */
        printf("FIFO Server: %s\n", buff);
    }

    return 0;
}

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>

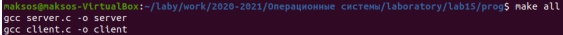
#define FIFO_NAME "/tmp/fifo"
#define MAX_BUFF 80

#endif /* __COMMON_H__ */

```

Рис. 2: Скрипты

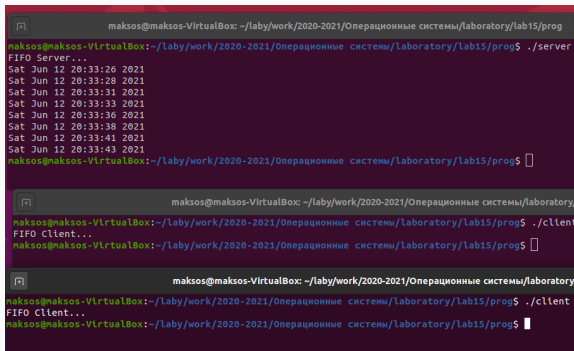
3. Выполняю компиляцию файлов с помощью Makefile. (рис. -fig. 3)

A terminal window with a dark purple background. The prompt is 'naksos@naksos-VirtualBox:~/laby/work/2020-2021/Операционные системы/Laboratory/Lab15/prog\$'. The user has entered the command 'make all'. The output shows two compilation commands: 'gcc server.c -o server' and 'gcc client.c -o client'.

```
naksos@naksos-VirtualBox:~/laby/work/2020-2021/Операционные системы/Laboratory/Lab15/prog$ make all
gcc server.c -o server
gcc client.c -o client
```

Рис. 3: Компиляция

4. Запускаю сервер в 1 терминале. Запускаю 2 клиента из других терминалов. Они работают верно (каждый клиент выводит в терминале, где запустили сервер, по 4 сообщения) (рис. -fig. 4)



The image displays three terminal windows from a VirtualBox environment, illustrating the execution of a FIFO-based program. The top window shows the server process, which outputs a series of timestamps. The middle and bottom windows show two separate client processes, each outputting a single line indicating they have connected to the server.

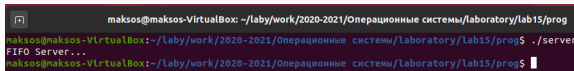
```
maksos@maksos-VirtualBox: ~/laby/work/2020-2021/Операционные системы/laboratory/lab15/prog
maksos@maksos-VirtualBox:~/laby/work/2020-2021/Операционные системы/laboratory/lab15/prog$ ./server
FIFO Server...
Sat Jun 12 20:33:26 2021
Sat Jun 12 20:33:28 2021
Sat Jun 12 20:33:31 2021
Sat Jun 12 20:33:33 2021
Sat Jun 12 20:33:36 2021
Sat Jun 12 20:33:38 2021
Sat Jun 12 20:33:41 2021
Sat Jun 12 20:33:43 2021
maksos@maksos-VirtualBox:~/laby/work/2020-2021/Операционные системы/laboratory/lab15/prog$

maksos@maksos-VirtualBox: ~/laby/work/2020-2021/Операционные системы/laboratory/
maksos@maksos-VirtualBox:~/laby/work/2020-2021/Операционные системы/laboratory/lab15/prog$ ./client
FIFO Client...
maksos@maksos-VirtualBox:~/laby/work/2020-2021/Операционные системы/laboratory/lab15/prog$

maksos@maksos-VirtualBox: ~/laby/work/2020-2021/Операционные системы/laboratory/
maksos@maksos-VirtualBox:~/laby/work/2020-2021/Операционные системы/laboratory/lab15/prog$ ./client
FIFO Client...
maksos@maksos-VirtualBox:~/laby/work/2020-2021/Операционные системы/laboratory/lab15/prog$
```

Рис. 4: Проверка работы программы

5. Проверка работы сервера. Он должен закрываться автоматически через 30 секунд. Работает верно (рис. -fig. 5)

A terminal window with a dark background. The title bar shows 'maksos@maksos-VirtualBox: ~/laby/work/2020-2021/Операционные системы/laboratory/lab15/prog'. The prompt is 'maksos@maksos-VirtualBox: ~/laby/work/2020-2021/Операционные системы/laboratory/lab15/prog\$'. The user has entered './server' and the output is 'FIFO Server...'. The prompt is now 'maksos@maksos-VirtualBox: ~/laby/work/2020-2021/Операционные системы/laboratory/lab15/prog\$' followed by a cursor.

```
maksos@maksos-VirtualBox: ~/laby/work/2020-2021/Операционные системы/laboratory/lab15/prog
maksos@maksos-VirtualBox: ~/laby/work/2020-2021/Операционные системы/laboratory/lab15/prog$ ./server
FIFO Server...
maksos@maksos-VirtualBox: ~/laby/work/2020-2021/Операционные системы/laboratory/lab15/prog$
```

Рис. 5: Проверка работы сервера

Приобрёл практические навыки работы с именованными каналами.

1. Именованные каналы отличаются от неименованных наличием идентификатора канала, который представлен как специальный файл (соответственно имя именованного канала — это имя файла).
2. Для создания неименованного канала используется системный вызов `pipe`. Массив из двух целых чисел является выходным параметром этого системного вызова.
3. Вы можете создавать именованные каналы из командной строки и внутри программы. С давних времен программой создания их в командной строке была команда: `mknod - $ mknod имя_файла`, однако команды `mknod` нет в списке команд `X/Open`, поэтому она включена не во все UNIX-подобные системы. Предпочтительнее применять в командной строке - `$ mkfifo имя_файла`.
4. `int read(int pipe_fd, void *area, int cnt);`