

Лабораторная работа №15

Бондарь Алексей Олегович

Май-2021

RUDN University, Moscow, Russian Federation

Приобретение практических навыков работы с именованными каналами.

1. Ознакомиться с теоретическим материалом.
2. Изучить основы программирования в оболочке ОС UNIX/Linux.
3. Выполнить упражнения.
4. Ответить на контрольные вопросы.

- 1) Для начала я создал необходимые файлы с помощью команды «touch common.h server.c client.c Makefile» и открыл редактор emacs для их редактирования.

```
aabondarj@dk8n67 ~ $ touch common.h server.c client.c Makefile  
aabondarj@dk8n67 ~ $ emacs &
```

Figure 1: Создание файлов

2) Далее я изменил коды программ, представленных в тексте лабораторной работы.

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.

```
#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"
```

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.

```
#include "common.h"

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

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

    /* создаем файл FIFO с открытыми для всех
    * правами доступа на чтение и запись
    */
    if(mknod(FIFO_NAME, S_IFIFO | 0666, 0 ) < 0)
    {
        fprintf(stderr, "%s: Невозможно создать FIFO (%s)\n",
            __FILE__, strerror(errno));
        exit(-1);
    }
    /* откроем FIFO на чтение */
    if((readfd = open(FIFO_NAME, O_RDONLY | O_NONBLOCK)) < 0)
    {
        fprintf(stderr, "%s: Невозможно открыть FIFO (%s)\n",
            __FILE__, strerror(errno));
        exit(-2);
    }
}
```

Figure 3: Скрипт server.c

```

/*начало отсчета времени */
clock_t start = time(NULL);

/*цикл работает, пока с момента начала отсчета времени прошло менее 30 сек */
while(time(NULL) - start < 30)
{
    /* читаем данные из FIFO и выводим на экран */
    while((n = read(readfd, buff, MAX_BUFF)) > 0)
    {
        if(write(1, buff, n) != n)
        {
            fprintf(stderr, "%s: Ошибка вывода (%s)\n",
                __FILE__, strerror(errno));
            exit(-3);
        }
    }
}
close(readfd); /* закроем FIFO */

/* удалим FIFO из системы */
if(unlink(FIFO_NAME) < 0)
{
    fprintf(stderr, "%s: Невозможно удалить FIFO (%s)\n",
        __FILE__, strerror(errno));
    exit(-4);
}

exit (0);
}

```

Figure 4: Скрипт server.c продолжение

В файл client.c добавил цикл, который отвечает за количество сообщений о текущем времени (4 сообщения), которое получается в результате выполнения команд (/* текущее время */) и команду sleep(5) для приостановки работы клиента на 5 секунд.

```
#include "common.h"

#define MESSAGE "Hello Server!!!\n"
int
main()
{
    int writefd; /* дескриптор для записи в FIFO */
    int msglen;

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

    /*цикл отвечающий за отправку сообщение о текущем времени*/
    for(int i=0;i<4;i++)
    {

        /* получим доступ к FIFO */
        if((writefd = open(FIFO_NAME, O_WRONLY)) < 0)
        {
            fprintf(stderr, "%s: Невозможно открыть FIFO (%s)\n",
                    FILE, strerror(errno));
            exit(-1);
            break;
        }
    }
}
```

Figure 5: Скрипт client.c


```

/*
/*текущее время*/
long int ttime=time(NULL);
char* text=ctime(&ttime);

/* передадим сообщение серверу */
msglen = strlen(text);
if(write(writefd, text, msglen) != msglen)
{
    fprintf(stderr, "%s: Ошибка записи в FIFO (%s)\n",
            FILE, strerror(errno));
    exit(-2);
}
/*приостановка работы клиента на 5 сек*/
sleep(5);

/* закроем доступ к FIFO */
close(writefd);

exit(0);
}

```

Figure 6: Скрипт client.c продолжение

Makefile (файл для сборки) не изменял.

```
all: server client

server: server.c common.h
    gcc server.c -o server

client: client.c common.h
    gcc client.c -o client

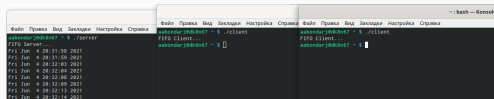
clean:
    -rm server client *.o
```

Figure 7: Скрипт Makefile

3) После написания кодов я, используя команду «make all», скомпилировал необходимые файлы.

Далее я проверил работу написанного кода. Отрыл 3 консоли (терминала) и запустил: в первом терминале – «./server», в остальных двух – «./client».

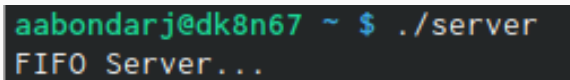
В результате каждый терминал - клиент вывел по 4 сообщения. Спустя 30 секунд работа сервера была прекращена. Программа работает корректно.



The image shows three terminal windows side-by-side. The leftmost window is titled 'xshenker@ubuntu: ~\$' and shows the command 'FIFO Server...' followed by a list of timestamps: 'Fri Jun 4 20:31:50 2021', 'Fri Jun 4 20:31:50 2021', 'Fri Jun 4 20:32:03 2021', 'Fri Jun 4 20:32:04 2021', 'Fri Jun 4 20:32:06 2021', 'Fri Jun 4 20:32:06 2021', 'Fri Jun 4 20:32:13 2021', and 'Fri Jun 4 20:32:14 2021'. The middle window is titled 'xshenker@ubuntu: ~\$' and shows the command 'FIFO Client...' followed by a prompt 'xshenker@ubuntu: ~\$'. The rightmost window is titled 'xshenker@ubuntu: ~\$' and shows the command 'FIFO Client...' followed by a prompt 'xshenker@ubuntu: ~\$'.

Figure 8: Консоли

Также я отдельно проверил длительность работы сервера, введя команду «./server» в одном терминале. Он завершил свою работу через 30 секунд.

A terminal window with a dark background. The prompt 'aabondarj@dk8n67 ~ \$' is in green. The command './server' is in blue. The output 'FIFO Server...' is in yellow.

```
aabondarj@dk8n67 ~ $ ./server
FIFO Server...
```

Figure 9: Команда «./server»

Если сервер завершит свою работу, не закрыв канал, то, когда мы будем запускать этот сервер снова, появится ошибка «Невозможно создать FIFO», так как у нас уже есть один канал.

В ходе выполнения данной лабораторной работы я приобрел практические навыки работы с именованными каналами.