

Отчёт по лабораторной работе №15

Именованные каналы

Николаев Дмитрий Иванович

Содержание

1	Цель работы	3
2	Выполнение лабораторной работы	4
2.1	Контрольные вопросы	9
3	Выводы	11

1 Цель работы

Приобретение практических навыков работы с именованными каналами.

2 Выполнение лабораторной работы

- 1) Создал файлы под будущую программу (common.h, server.c client1.c, client2.c, Makefile), позднее переместив их в другую директорию для удобства.

```
dinikolaev@dk6n65 ~ $ mkdir lab15
dinikolaev@dk6n65 ~ $ ls lab15
dinikolaev@dk6n65 ~ $ touch common.h
dinikolaev@dk6n65 ~ $ touch server.c
dinikolaev@dk6n65 ~ $ touch client1.c
dinikolaev@dk6n65 ~ $ touch client2.c
dinikolaev@dk6n65 ~ $ touch Makefile
```

- Создал файлы для сервера и

клиентов

- 2) Изучил приведённые в тексте программы и взял их за образец, сделав некоторые изменения:

2.1) В заголовочный файл со стандартными определениями добавил библиотеку time.h

```

#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<time.h>

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

#endif

```

- Заголовочный файл common.h

2.2) В программе с сервером изменил условие выхода - поставив ограничение по времени существования сервера.

```

#include "common.h"

int main()
{
    int readfd;
    int n;
    char buff[MAX_BUFF];
    printf("FIFO Server...\n");
    if(mknod(FIFO_NAME, S_IFIFO | 0666, 0) < 0)
    {
        fprintf(stderr, "%s: Impossible to create (%s)\n", __FILE__, strerror(errno));
        exit(-1);
    }
    if((readfd = open(FIFO_NAME, O_RDONLY)) < 0)
    {
        fprintf(stderr, "%s: Impossible to open FIFO (%s)\n", __FILE__, strerror(errno));
        exit(-2);
    }

    clock_t now = time(NULL), start=time(NULL);
    while(now-start < 30)
    {
        if ((n=read(readfd, buff, MAX_BUFF)) == 0)
        {
            fprintf(stderr, "%s: Output error (%s)\n", __FILE__, strerror(errno));
            exit(-3);
        }
        now=time(NULL);
    }
    printf("\nServer is out of time \n%u seconds passed", (now-start));
    close(readfd);
    if(unlink(FIFO_NAME) < 0)
    {
        fprintf(stderr, "%s: Impossible to delete FIFO (%s)\n", __FILE__, strerror(errno));
        exit(-4);
    }
    exit(0);
}

```

- Программа, реализующая сервер

2.3) В программах с клиентами поставил цикл, для ограничения по времени, где первый клиент останавливает работу на 4 секунды (sleep(4)), а второй - на 5 (sleep(5)).

```

#include "common.h"

#define MESSAGE "Hello Server!!!\n"
int main()
{
    for (int i = 0; i < 15; i++)
    {
        int writefd;
        int msglen;
        long int time_;
        time_ = time(NULL);

        #define MESSAGE ctime(&time_)
        printf("FIFO Client...\n");
        if((writefd = open(FIFO_NAME, O_WRONLY)) < 0)
        {
            fprintf(stderr, "%s: Impossible to open FIFO (%s)\n", __FILE__, strerror(errno));
            exit(-1);
        }
        msglen = strlen(MESSAGE);
        if(write(writefd, MESSAGE, msglen) != msglen)
        {
            fprintf(stderr, "%s: Error writing to FIFO (%s)\n", __FILE__, strerror(errno));
            exit(-2);
        }
        sleep(4);
        close(writefd);
    }
    exit(0);
}

```

- Программа, реализующая первый клиент

```

#include "common.h"

#define MESSAGE "Hello Server!!!\n"
int main()
{
    for (int i = 0; i < 15; i++)
    {
        int writefd;
        int msglen;
        long int time_;
        time_ = time(NULL);

        #define MESSAGE ctime(&time_)
        printf("FIFO Client...\n");
        if((writefd = open(FIFO_NAME, O_WRONLY)) < 0)
        {
            fprintf(stderr, "%s: Impossible to open FIFO (%s)\n", __FILE__, strerror(errno));
            exit(-1);
        }
        msglen = strlen(MESSAGE);
        if(write(writefd, MESSAGE, msglen) != msglen)
        {
            fprintf(stderr, "%s: Error writing to FIFO (%s)\n", __FILE__, strerror(errno));
            exit(-2);
        }
        sleep(5);
        close(writefd);
    }
    exit(0);
}

```

- Программа, реализующая второй клиент

2.4) В make-файле изменил названия для компилирующихся файлов (сделав

аналог для второго клиента), добавив где необходимо файлы, связанные со вторым клиентом.

```
Makefile      [-M--]  0 L:[ 1+10
all: server client1 client2

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

client1: client1.c common.h
<----->gcc client1.c -o client1

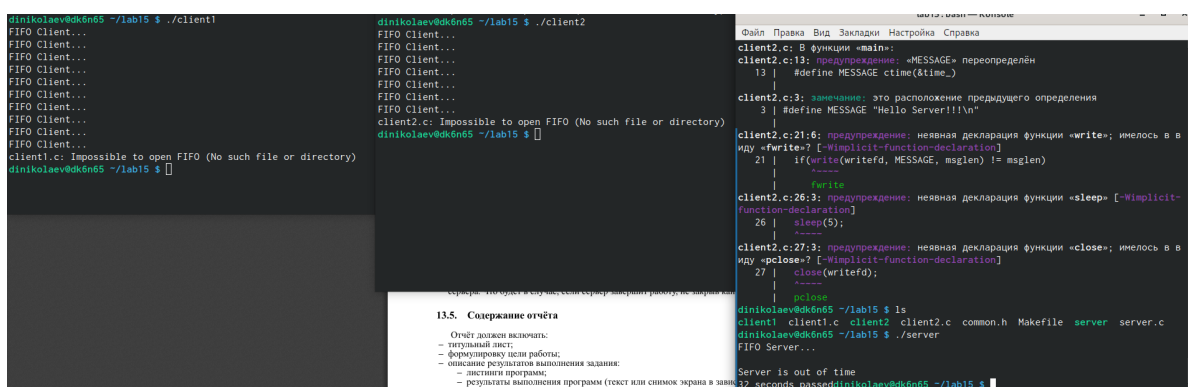
client2: client2.c common.h
<----->gcc client2.c -o client2

clean: -rm server client1 client2 *.o
```

- Makefile

3) Таким образом, я написал программы сервера и двух клиентов, где клиенты передают сообщение с разной периодичностью, приостанавливая свою работу, а сервер заканчивает свою работу, выводя при этом время своей работы.

4) С помощью make я скомпилировал программы и запустил их исполняемые файлы на разных консолях (под сервер и два клиента).



The image shows three terminal windows. The left window shows client1 sending messages. The middle window shows client2 sending messages and then an error. The right window shows the server output and compilation errors for client2.c.

```

dinikolaev@dk6n65 ~/lab15 $ ./client1
FIFO Client...
FIFO Client...
FIFO Client...
FIFO Client...
FIFO Client...
FIFO Client...
FIFO Client...
FIFO Client...
FIFO Client...
FIFO Client...
client1.c: Impossible to open FIFO (No such file or directory)
dinikolaev@dk6n65 ~/lab15 $

dinikolaev@dk6n65 ~/lab15 $ ./client2
FIFO Client...
FIFO Client...
FIFO Client...
FIFO Client...
FIFO Client...
FIFO Client...
FIFO Client...
FIFO Client...
FIFO Client...
FIFO Client...
client2.c: Impossible to open FIFO (No such file or directory)
dinikolaev@dk6n65 ~/lab15 $

dinikolaev@dk6n65 ~/lab15 $ ls
client1 client1.c client2 client2.c common.h Makefile server server.c
dinikolaev@dk6n65 ~/lab15 $ ./server
FIFO Server...
Server is out of time
32 seconds passed
dinikolaev@dk6n65 ~/lab15 $

```

13.5. Содержание отчёта

- Отчёт должен включать:
 - титульный лист;
 - формулировку цели работы;
 - описание результатов выполнения задания;
 - листинги программы;
 - результаты выполнения программ (текст или снимок экрана в зависимости от задания);

- Результат работы сервера и клиентов на нём

2.1 Контрольные вопросы

1. Именованные файлы имеют идентификатор канала, представленный в специальном файле (неименованные соответственно не имеют идентификатора).
2. Можно с помощью системного вызова `pipe` (массив из двух целых чисел - выходной параметр).
3. Можно, в основном используя `mkfifo`.
4. `int read(int pipe_fd, void area, int cnt); int write(int pipe_fd, void area, int cnt);`
Где первый аргумент - дескриптор канала, второй - указатель на область памяти, а третий - количество байт памяти.
5. `int mkfifo(const char *pathname, mode_t mode); mkfifo(FIFO_NAME, 0600);` Где первый параметр - имя файла(идентификатор канала), второй - маска прав доступа.
6. При чтении меньшего числа байт, чем находится в канале, возвращается требуемое число байт, остаток сохраняется для следующих чтений, при чтении большего - возвращается доступное число байт.
7. При записи меньшего числа байт, чем это позволяет канал или FIFO, гарантируется атомарность операции, другими словами, если несколько процессов одновременно записывают в канал, порции данных от процессов не перемешиваются, если же производится запись большего числа байт - вызов `write(2)` блокируется до освобождения места, а атомарность операции не гарантируется.
8. В общем случае это возможно (каждый из взаимодействующих каналов пишет и читает информацию в канал), но традиционной схемой работы с каналом является однонаправленная организация, где канал связывают несколько взаимодействующих процессов, каждый из которых либо читает, либо пишет в канал.
9. Функция `write` записывает `length` байт из буфера `buffer` в файл, определённый дескриптором файла `fd`. С помощью `write` можно посылать сообщение

- клиенту или серверу. Операция является двоичной и без буферизации(1).
10. `Strerror` - функция языков C/C++, которая транслирует код ошибки, хранящийся в глобальной переменной `errno`, в сообщение об ошибке. Возвращённый указатель ссылается на статическую строку с ошибкой, не изменённая программой. Дальнейшие вызовы функции `strerror` перезапишут содержание ошибочной строки. Сообщения об ошибках зависят от платформы или компилятора.

3 Выводы

В результате работы, я приобрёл практические навыки работы с именнованными каналами.