

Отчёт по лабораторной работе

Лабораторная работа № 15.

Lukashov Nikita

Содержание

1	Цель работы	5
2	Выполнение лабораторной работы	6
3	Контрольные вопросы	11
4	Вывод	13

List of Tables

List of Figures

1 Цель работы

Приобретение практических навыков работы с именованными каналами.

2 Выполнение лабораторной работы

1. Изучите приведённые в тексте программы `server.c` и `client.c`. Взяв данные примеры за образец, напишите аналогичные программы, внося следующие изменения:
2. Работает не 1 клиент, а несколько (например, два).
3. Клиенты передают текущее время с некоторой периодичностью (например, раз в пять секунд). Используйте функцию `sleep()` для приостановки работы клиента.
4. 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?

```

#include "common.h"
int
main()
{
    int readfd; /* дескриптор для чтения из FIFO */
    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)) < 0)
    {
        fprintf(stderr, "%s: Невозможно открыть FIFO (%s)\n",
            __FILE__, strerror(errno));
        exit(-2);
    }

    /* читаем данные из FIFO и выводим на экран */
    clock_t now = time(NULL), start = time(NULL);
    while((now-start)<30){
        while((n = read(readfd, buff, MAX_BUFF)) > 1)
        {
            if(write(2, buff, n) != n)
            {
                fprintf(stderr, "%s: Ошибка вывода (%s)\n",
                    __FILE__, strerror(errno));
                exit(-3);
            }
        }
        now=time(NULL);
    }
    printf("Работа сервера окончена, прошло %li секунд", (now-start));
    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);
}

```

```

#include "common.h"
#define MESSAGE "Hello Server!!! from 1\n"
int
main()
{
    int writefd; /* дескриптор для записи в FIFO */
    int msglen;
    long long int T;
    for(int i=0; i<5;i++){
        sleep(5);
        T=time(NULL);
        /* баннер */
        printf("FIFO Client...\n");

        /* 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);
        }
        /* передадим сообщение серверу */
        msglen = strlen(MESSAGE);
        if(write(writefd, MESSAGE, msglen) != msglen)
        {
            fprintf(stderr, "%s: Ошибка записи в FIFO (%s)\n",
                __FILE__, strerror(errno));
            exit(-2);
        }

        /* закроем доступ к FIFO */
        close(writefd);
    }
    exit(0);
}

```



```

#include "common.h"
#define MESSAGE "Hello Server!!! from 2\n"
int
main()
{
    int writefd; /* дескриптор для записи в FIFO */
    int msglen;
    long long int T;
    for(int i=0; i<3; i++){
        sleep(5);
        T=time(NULL);
        /* баннер */
        printf("FIFO Client...\n");

        /* 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);
        }
        /* передадим сообщение серверу */
        msglen = strlen(MESSAGE);
        if(write(writefd, MESSAGE, msglen) != msglen)
        {
            fprintf(stderr, "%s: Ошибка записи в FIFO (%s)\n",
                    __FILE__, strerror(errno));
            exit(-2);
        }

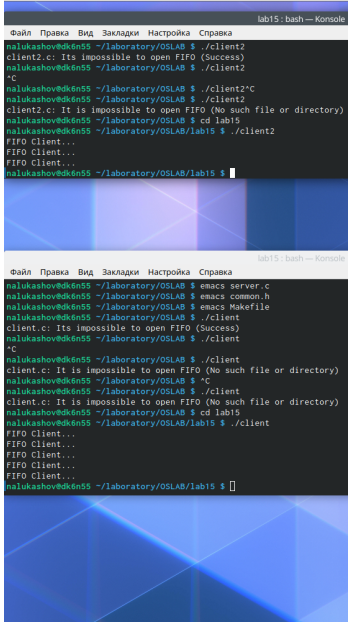
        /* закроем доступ к FIFO */
        close(writefd);
    }
    exit(0);
}

```

```
#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>
```

```
#define FIFO_NAME "/tmp/fifo"
#define MAX_BUFF 80
#endif /* __COMMON_H__ */
```



```
lab15: bash — Konsole
halukashov@den55 ~/laboratory/OSLAB $ ./client2
client2.c: It's impossible to open FIFO (Success)
halukashov@den55 ~/laboratory/OSLAB $ ./client2
^C
halukashov@den55 ~/laboratory/OSLAB $ ./client2^C
halukashov@den55 ~/laboratory/OSLAB $ ./client2
client2.c: It is impossible to open FIFO (No such file or directory)
halukashov@den55 ~/laboratory/OSLAB $ cd lab15
halukashov@den55 ~/laboratory/OSLAB/lab15 $ ./client2
FIFO Client...
FIFO Client...
FIFO Client...
halukashov@den55 ~/laboratory/OSLAB/lab15 $

lab15: bash — Konsole
halukashov@den55 ~/laboratory/OSLAB $ emacs server.c
halukashov@den55 ~/laboratory/OSLAB $ emacs common.h
halukashov@den55 ~/laboratory/OSLAB $ emacs Makefile
halukashov@den55 ~/laboratory/OSLAB $ ./client
client.c: It's impossible to open FIFO (Success)
halukashov@den55 ~/laboratory/OSLAB $ ./client
^C
halukashov@den55 ~/laboratory/OSLAB $ ./client
client.c: It is impossible to open FIFO (No such file or directory)
halukashov@den55 ~/laboratory/OSLAB $ ^C
halukashov@den55 ~/laboratory/OSLAB $ ./client
client.c: It is impossible to open FIFO (No such file or directory)
halukashov@den55 ~/laboratory/OSLAB $ cd lab15
halukashov@den55 ~/laboratory/OSLAB/lab15 $ ./client
FIFO Client...
FIFO Client...
FIFO Client...
FIFO Client...
halukashov@den55 ~/laboratory/OSLAB/lab15 $
```

3 Контрольные вопросы

1. Именованные каналы отличаются от неименованных наличием идентификатора

канала, который представлен как специальный файл (соответственно имя именованного канала — это имя файла). Поскольку файл находится на локальной файловой системе, данное IPC используется внутри одной системы.

2. Создание неименованного канала из командной строки невозможно.

3. Создание именованного канала из командной строки возможно.

4. `int read(int pipe_fd, void *area, int cnt);`

`int write(int pipe_fd, void *area, int cnt);`

Первый аргумент этих вызовов - дескриптор канала, второй - указатель на область памяти, с которой происходит обмен, третий - количество байт. Оба вызова возвращают число переданных байт (или -1 - при ошибке).

5. `int mkfifo (const char *pathname, mode_t mode) ;`

`mkfifo(FIFO_NAME, 0600) ;`

Первый параметр — имя файла, идентифицирующего канал, второй параметр маска прав доступа к файлу. Вызов функции `mkfifo()` создаёт файл канала (с именем, заданным макросом `FIFO_NAME`).

6. При чтении меньшего числа байтов, чем находится в канале, возвращается требуемое число байтов, остаток сохраняется для последующих чтений. При чтении большего числа байтов, чем находится в канале или FIFO возвращается доступное число байтов.

7. При записи большего числа байтов, чем это позволяет канал или FIFO, вызов `write(2)` блокируется до освобождения требуемого места. При этом атомарность операции не гарантируется. Если процесс пытается записать данные в канал, не открытый ни одним процессом на чтение, процессу генерируется сигнал. Запись числа байтов, меньшего емкости канала или FIFO, гарантированно атомарно. Это означает, что в случае, когда несколько процессов одновременно записывают в канал, порции данных от этих процессов не перемешиваются.
8. В общем случае возможна много направленная работа процессов с каналом, т.е. возможна ситуация, когда с одним и тем же каналом взаимодействуют два и более процесса, и каждый из взаимодействующих каналов пишет и читает информацию в канал. Но традиционной схемой организации работы с каналом является однонаправленная организация, когда канал связывает два, в большинстве случаев, или несколько взаимодействующих процесса, каждый из которых может либо читать, либо писать в канал.
9. `Write` - Функция записывает `length` байтов из буфера `buffer` в файл, определенный дескриптором файла `fd`. Эта операция чисто 'двоичная' и без буферизации. Реализуется как непосредственный вызов `DOS`. С помощью функции `write` мы посылаем сообщение клиенту или серверу.
10. Строковая функция `strerror` - функция языков C/C++, транслирующая код ошибки, который обычно хранится в глобальной переменной `errno`, в сообщение об ошибке, понятном человеку. Ошибки эти возникают при вызове функций стандартных Си-библиотек. Возвращенный указатель ссылается на статическую строку с ошибкой, которая не должна быть изменена программой. Дальнейшие вызовы функции `strerror` перезапишут содержание этой строки. Интерпретированные сообщения об ошибках могут различаться, это зависит от платформы и компилятора.

4 Вывод

Приобрел практические навыки работы с именованными каналами.