

Отчёт по лабораторной работе 14

Елизавета Александровна Гайдамака

Содержание

Цель работы	3
Задание	4
Теоретическое введение	5
Выполнение лабораторной работы	8
Выводы	14

Цель работы

Приобретение практических навыков работы с именованными каналами.

Задание

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

Теоретическое введение

1. В чем ключевое отличие именованных каналов от неименованных?

Традиционный канал — «безымянный», потому что существует анонимно и только во время выполнения процесса. Именованный канал — существует в системе и после завершения процесса. Он должен быть «отсоединён» или удалён, когда уже не используется.

2. Возможно ли создание неименованного канала из командной строки?

Неименованный канал создается вызовом `pipe`, который заносит в массив `int` [2] два дескриптора открытых файлов. `fd[0]` – открыт на чтение, `fd[1]` – на запись (вспомните `STDIN == 0`, `STDOUT == 1`). Канал уничтожается, когда будут закрыты все файловые дескрипторы ссылающиеся на него.

3. Возможно ли создание именованного канала из командной строки?

Вы можете создавать именованные каналы из командной строки и внутри программы. С давних времен программой создания их в командной строке была команда `mknod`.

4. Опишите функцию языка C, создающую неименованный канал.

Для создания файла FIFO можно использовать более общую функцию `mknod(2)`, предназначенную для создания специальных файлов различных типов (FIFO, сокеты, файлы устройств и обычные файлы для хранения данных).

```
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <unistd.h>
```

```
int mknod(const char *pathname, mode_t mode, dev_t dev);
```

5. Опишите функцию языка C, создающую именованный канал.

Файлы именованных каналов создаются функцией mkfifo(3).

```
#include <sys/types.h>
#include <sys/stat.h>
```

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

6. Что будет в случае прочтения из fifo меньшего числа байтов, чем находится в канале? Большого числа байтов?

При чтении меньшего - возвращается обрезанное число байтов. Большого - возвращаются все, которые есть.

7. Аналогично, что будет в случае записи в fifo меньшего числа байтов, чем позволяет буфер? Большого числа байтов?

Меньшего - все нормально, и данные не перемешиваются. Большого - происходит блокировка.

8. Могут ли два и более процессов читать или записывать в канал?

Да, могут.

9. Опишите функцию write (тип возвращаемого значения, аргументы и логику работы). Что означает 1 (единица) в вызове этой функции в программе server.c (строка 42)?

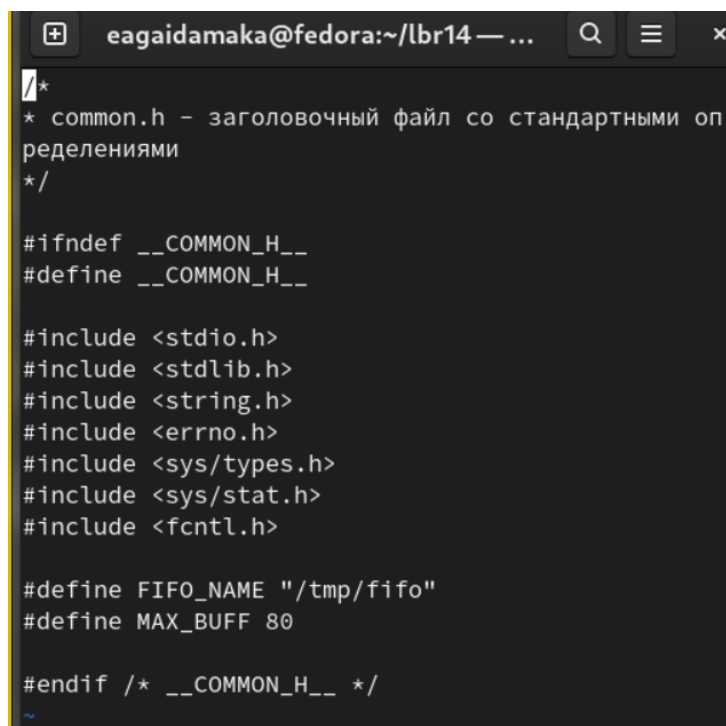
`write` записывает до `count` байтов из буфера `buf` в файл, на который ссылается файловый дескриптор `fd`. POSIX указывает на то, что вызов `write()`, произошедший после вызова `read()` возвращает уже новое значение. Заметьте, что не все файловые системы соответствуют стандарту POSIX. При единице возвращает действительное число байтов.

10. Опишите функцию `strerror`.

Функция `strerror()` возвращает указатель на строку, которая описывает код ошибки, переданный в аргументе `errnum`, возможно используя часть `LC_MESSAGES` текущей локали, чтобы выбрать соответствующий язык. (Например, если `errnum` равно `EINVAL`, возвращаемое описание будет “Недопустимый аргумент”.) Эта строка не должна быть изменена приложением, но могут быть изменены последующий вызов `strerror()` или `strerror_l()`. Нет другой библиотеки функция, включая `reerror(3)`, изменит эту строку.

Выполнение лабораторной работы

Текст файла `common.h`:

A screenshot of a terminal window with a dark background. The window title is "eagaidamaka@fedora:~/lbr14 — ...". The terminal displays the content of a C header file named common.h. The code includes a multi-line comment, preprocessor directives for __COMMON_H__, several #include statements for standard C libraries, and two #define macros for FIFO_NAME and MAX_BUFF. The code is as follows:

```
/*
 * common.h - заголовочный файл со стандартными оп
 * ределениями
 */

#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__ */
```

Рис. 1: Рис.1

Текст файла `server.c`:


```

#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);
    }

    clock_t now=time(NULL), start=time(NULL);
    while(now-start<30){
        while((n = read(readfd, buff, MAX_BUFF)) > 0){
            if (write(1, buff, n) !=n){

```

Рис. 2: Рис.2

```

        fprintf(stderr, "%s: Ошибка вывода (%s)\n",
                __FILE__, strerror(errno));
    }
}
now=time(NULL);
printf("server timeout, %li - second passed\n", (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);
}

```

Рис. 3: Рис.3

Текст файла client.c:

```
#include "common.h"

#define MESSAGE "Hello Server!!!\n"

int
main(){
    int msg, len, i; /* дескриптор для записи в FIFO */
    long int t;
    for(i=0; i<20; i++){
        sleep(3);
        t = time(NULL);
        printf("FIFO Client...\n");

        if((msg = open(FIFO_NAME,O_WRONLY))<0){
            fprintf(stderr, "%s: Невозможно открыть FIFO (%s)\n",
                __FILE__, strerror(errno));
            exit(-1);
        }
        len = strlen(MESSAGE);

        if (write(msg, MESSAGE, len) != len){
            fprintf(stderr, "s:Ошибка записи в FIFO (%s)\n",
                __FILE__, strerror(errno));
            exit(-2);
        }
        close(msg);
    }
    exit(0);
}
```

Рис. 4: Рис.4

Запускаем make.

```

[eagaidamaka@fedora lbr14]$ make
gcc server.c -o server
server.c: В функции «main»:
server.c:39:13: предупреждение: неявная декларация функции «time» [-Wimplicit-function-declaration]
   39 |   clock_t now=time(NULL), start=time(NULL);
      |               ^~~~
server.c:41:16: предупреждение: неявная декларация функции «read»; имелось
в виду «fread»? [-Wimplicit-function-declaration]
   41 |   while((n = read(readfd, buff, MAX_BUFF)) > 0){
      |               ^~~~
      |               fread
server.c:42:13: предупреждение: неявная декларация функции «write»; имелось
в виду «fwrite»? [-Wimplicit-function-declaration]
   42 |   if (write(1, buff, n) !=n){
      |       ^~~~~
      |       fwrite
server.c:51:1: предупреждение: неявная декларация функции «close»; имелось
в виду «pclose»? [-Wimplicit-function-declaration]
   51 |   close(readfd); /* закроем FIFO */
      |   ^~~~~
      |   pclose
server.c:54:4: предупреждение: неявная декларация функции «unlink» [-Wimplicit-function-declaration]
   54 |   if(unlink(FIFO_NAME) < 0)
      |       ^~~~~~
gcc client.c -o client
client.c: В функции «main»:
client.c:18:9: предупреждение: неявная декларация функции «sleep» [-Wimplicit-function-declaration]
   18 |     sleep(3);
      |     ^~~~~
client.c:19:13: предупреждение: неявная декларация функции «time» [-Wimplicit-function-declaration]
   19 |     t = time(NULL);

```

Рис. 5: Рис.5

Запускаем ./server и ./client, ждем 30 секунд.

```
[eagaidamaka@fedora lbr14]$ ./server
FIFO Server...
Hello Server!!!
Hello Server!!!
Hello Server!!!
Hello Server!!!
Hello Server!!!
Hello Server!!!
Hello Server!!!
Hello Server!!!
Hello Server!!!
Hello Server!!!
server timeout, 30 - second passed
[eagaidamaka@fedora lbr14]$
```

```
[eagaidamaka@fedora lbr14]$ ./client
FIFO Client...
FIFO Client...
FIFO Client...
FIFO Client...
FIFO Client...
FIFO Client...
FIFO Client...
FIFO Client...
FIFO Client...
FIFO Client...
FIFO Client...
client.c: Невозможно открыть FIFO (No such file o
r directory)
[eagaidamaka@fedora lbr14]$
```

Рис. 6: Рис.6

Выводы

Благодаря данной работе я приобрела практические навыки работы с именovanными каналами.