

# **Отчет по лабораторной работе №14**

**Операционные системы**

Ищенко Ирина Олеговна

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>5</b>
<b>2</b>	<b>Задание</b>	<b>6</b>
<b>3</b>	<b>Теоретическое введение</b>	<b>7</b>
<b>4</b>	<b>Выполнение лабораторной работы</b>	<b>8</b>
<b>5</b>	<b>Вывод</b>	<b>12</b>
<b>6</b>	<b>Ответы на контрольные вопросы</b>	<b>13</b>

## Список иллюстраций

4.1	Создание файлов <code>common.h</code> , <code>service.c</code> , <code>client.c</code> . . . . .	8
4.2	Содержимое заголовочного файла. . . . .	8
4.3	Содержимое файла реализации сервера . . . . .	9
4.4	Содержимое файла реализации клиента . . . . .	10
4.5	Содержание <code>Makefile</code> . . . . .	10
4.6	Запуск программ на двух консолях. . . . .	11

## Список таблиц

# **1 Цель работы**

Приобретение практических навыков работы с именованными каналами

## 2 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?

### 3 Теоретическое введение

Одним из видов взаимодействия между процессами в операционных системах является обмен сообщениями. Под сообщением понимается последовательность байтов, передаваемая от одного процесса другому.

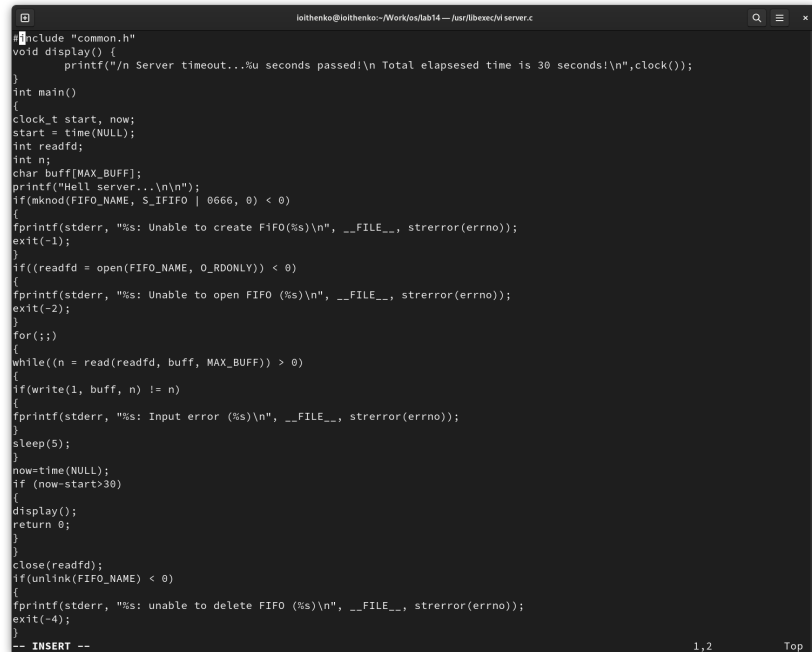
В операционных системах типа UNIX есть 3 вида межпроцессорных взаимодействий: общедюниксные (именованные каналы, сигналы), System V Interface Definition (SVID — разделяемая память, очередь сообщений, семафоры) и BSD (сокеты).

Для передачи данных между неродственными процессами можно использовать механизм именованных каналов (named pipes). Данные передаются по принципу FIFO (First In First Out) (первым записан — первым прочитан), поэтому они называются также FIFO pipes или просто FIFO. 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.





Заполняю файл server.c, реализующий сервер. Использую функцию clock для определения времени работы сервера. Сервер работает не бесконечно, а прекращает работу через некоторое время, в моем случае, через 30 секунд (рис. 4.3):



```
#include "common.h"
void display() {
    printf("/n Server timeout...%u seconds passed/n Total elapsed time is 30 seconds/n", clock());
}
int main()
{
    clock_t start, now;
    start = time(NULL);
    int readfd;
    int n;
    char buff[MAX_BUFF];
    printf("Hell server.../n/n");
    if(mknod(FIFO_NAME, S_IFIFO | 0666, 0) < 0)
    {
        fprintf(stderr, "%s: Unable to create Fifo(%s)/n", __FILE__, strerror(errno));
        exit(-1);
    }
    if((readfd = open(FIFO_NAME, O_RDONLY)) < 0)
    {
        fprintf(stderr, "%s: Unable to open FIFO (%s)/n", __FILE__, strerror(errno));
        exit(-2);
    }
    for(;;)
    {
        while((n = read(readfd, buff, MAX_BUFF)) > 0)
        {
            if(write(1, buff, n) != n)
            {
                fprintf(stderr, "%s: Input error (%s)/n", __FILE__, strerror(errno));
            }
            sleep(5);
        }
        now = time(NULL);
        if (now - start > 30)
        {
            display();
            return 0;
        }
    }
    close(readfd);
    if(unlink(FIFO_NAME) < 0)
    {
        fprintf(stderr, "%s: unable to delete FIFO (%s)/n", __FILE__, strerror(errno));
        exit(-4);
    }
}
-- INSERT --
```

Рис. 4.3: Содержимое файла реализации сервера

Заполняю файлы client.c (рис. 4.4):

```
ioithenko@ioithenko:~/Work/os/lab14 — /usr/libexec/vi client.c
#define MESSAGE "Hello Server!!! \n"

int main ()
{
    int writefd;
    int msglen;
    printf("FIFO Client...\n");
    if ((writefd = open(FIFO_NAME, O_WRONLY)) < 0)
    {
        fprintf(stderr, "%s: Невозможно открыть FIFO (%s)\n",
            __FILE__, strerror(errno));
        exit(-1);
    }
    msglen = strlen(MESSAGE);
    if (write(writefd, MESSAGE, msglen) != msglen)
    {
        fprintf(stderr, "%s: Ошибка записи в FIFO (%s)\n",
            __FILE__, strerror(errno));
        exit(-2);
    }
    close(writefd);
    exit(0);
}

-- INSERT --
```

Рис. 4.4: Содержимое файла реализации клиента

Создаю и заполняю Makefile (рис. 4.5):

```
ioithenko@ioithenko:~/Work/os/lab14 — /usr/libexec/vi Makefile
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
```

Рис. 4.5: Содержание Makefile.

Затем выполняю программу. На одной консоли запускаю программу server, а на другой консоли запускаю программу client (рис. 4.6).

```
ioithenko@ioithenko:~/Work/os/lab14$ cat client.c
//~~~~~
// write
client.c:21:11: предупреждение: неявная декларация функции «close»; имелось в виду «pclose»? [-Wimplicit-function-declaration]
    21 | close(writefd);
        | ~~~~~
        | pclose
[ioithenko@ioithenko lab14]$ ./server
Hell server...

Hello Server!!!
Hello Server!!!
Hello Server!!!
Hello Server!!!
Hello Server!!!
/n Server timeout...7895968 seconds passed!
Total elapsed time is 30 seconds!
[ioithenko@ioithenko lab14]$

[ioithenko@ioithenko lab14]$ ./client
FIFO Client...
[ioithenko@ioithenko lab14]$ ./client
FIFO Client...
[ioithenko@ioithenko lab14]$ ./client
FIFO Client...
[ioithenko@ioithenko lab14]$ ./client
FIFO Client...
[ioithenko@ioithenko lab14]$ ./client
FIFO Client...
[ioithenko@ioithenko lab14]$
```

Рис. 4.6: Запуск программ на двух консолях.

## **5 Вывод**

В ходе выполнения лабораторной работы я приобрела практические навыки в работы с именованными каналами.

## 6 Ответы на контрольные вопросы

1. В чем ключевое отличие именованных каналов от неименованных?

Именованные каналы отличаются от неименованных наличием идентификатора канала, который представлен как специальный файл (соответственно имя именованного канала — это имя файла).

2. Возможно ли создание неименованного канала из командной строки?

Создание неименованного канала из командной строки возможно командой `pipe`.

3. Возможно ли создание именованного канала из командной строки?

Создание именованного канала из командной строки возможно с помощью `mkfifo`.

4. Опишите функцию языка C, создающую неименованный канал.

Функция языка C, создающая неименованный канал: `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. Опишите функцию языка C, создающую именованный канал.

Функция языка C, создающая именованный канал: `int mkfifo (const char *pathname, mode_t mode)`; 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`): `mkfifo(FIFO_NAME, 0600)`.

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

При чтении меньшего числа байтов, возвращается требуемое число байтов, остаток сохраняется для следующих чтений.

При чтении большего числа байтов, возвращается доступное число байтов

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

Запись числа байтов, меньшего ёмкости канала или FIFO, гарантированно атомарно. Это означает, что в случае, когда несколько процессов одновременно записывают в канал, порции данных от этих процессов не перемешиваются.

При записи большего числа байтов, чем это позволяет канал или FIFO, вызов `write` блокируется до освобождения требуемого места.

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

Два и более процессов могут читать и записывать в канал.

9. Опишите функцию `write` (тип возвращаемого значения, аргументы и логику работы).

Функция `write` записывает `length` байтов из буфера `buffer` в файл, определенный дескриптором файла `fd`. Эта операция чисто 'двоичная' и без буферизации. При единице возвращает действительное число байтов. Функция `write` возвращает число действительно записанных в файл байтов или -1 при ошибке, устанавливая при этом `errno`.

10. Опишите функцию `strerror`.

Строковая функция `strerror` - функция языков C/C++, транслирующая код ошибки, который обычно хранится в глобальной переменной `errno`, в сообщение об ошибке, понятном человеку.