

Отчёт по лабораторной работе 14

Именованные каналы

Сидорова Наталья Андреевна

Содержание

1	Цель работы	5
2	Задание	6
3	Теоретическое введение	7
4	Выполнение лабораторной работы	8
5	Выводы	13
	Список литературы	14

Список иллюстраций

4.1	1 часть кода	8
4.2	2 часть кода	9
4.3	Код	10
4.4	Работа файлов	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:

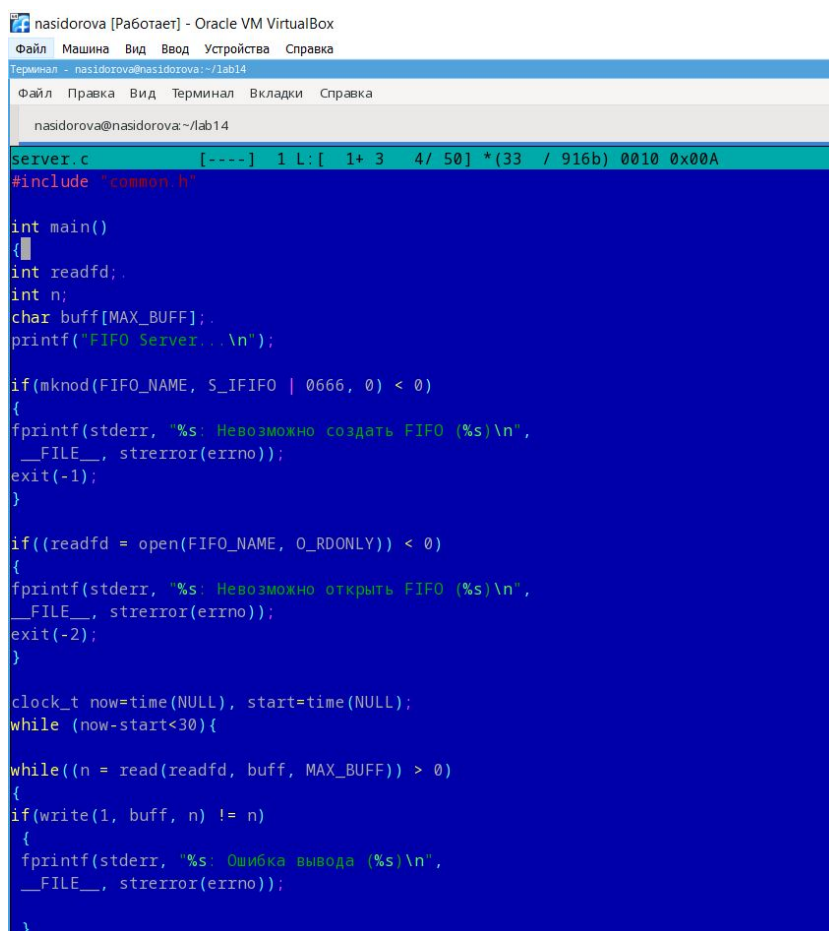
1. Работает не 1 клиент, а несколько (например, два).
2. Клиенты передают текущее время с некоторой периодичностью (например, раз в пять секунд). Используйте функцию `sleep()` для приостановки работы клиента.
3. Сервер работает не бесконечно, а прекращает работу через некоторое время (например, 30 сек). 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: общепонимание (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. Файлы именованных каналов создаются функцией `mkfifo(3)`.

4 Выполнение лабораторной работы

Изменила код в файле server.c в соответствии с заданием (рис. 4.1).



```
server.c [----] 1 L: [ 1+ 3 4/ 50] *(33 / 916b) 0010 0x00A
#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: Невозможно создать FIFO (%s)\n",
            __FILE__, strerror(errno));
        exit(-1);
    }

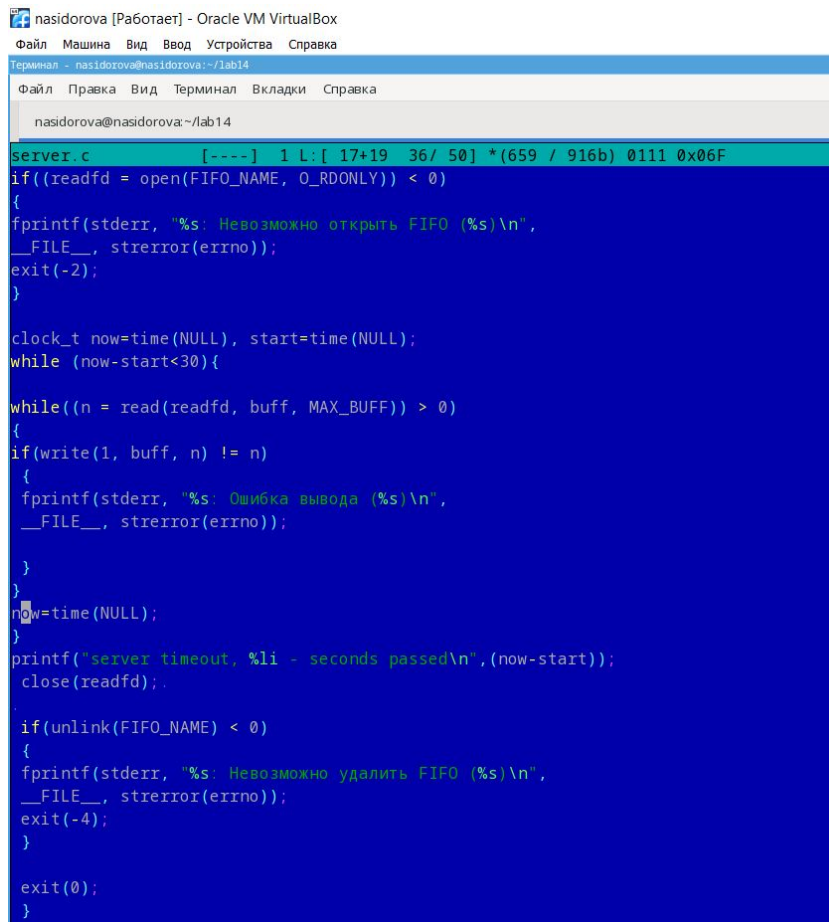
    if((readfd = open(FIFO_NAME, O_RDONLY)) < 0)
    {
        fprintf(stderr, "%s: Невозможно открыть FIFO (%s)\n",
            __FILE__, strerror(errno));
        exit(-2);
    }

    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)
            {
                fprintf(stderr, "%s: Ошибка вывода (%s)\n",
                    __FILE__, strerror(errno));
            }
        }
    }
}
```

Рис. 4.1: 1 часть кода

Продолжение кода (рис. 4.2).



```
server.c [----] 1 L: [ 17+19 36/ 50] *(659 / 916b) 0111 0x06F
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){

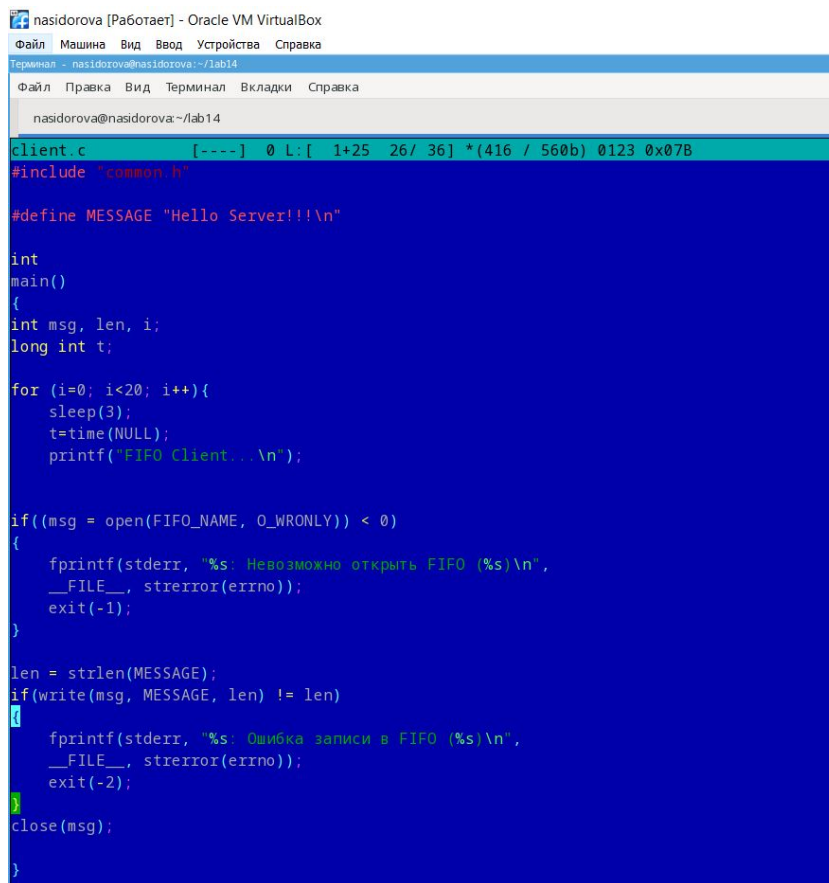
while((n = read(readfd, buff, MAX_BUFF)) > 0)
{
    if(write(1, buff, n) != n)
    {
        fprintf(stderr, "%s: Ошибка вывода (%s)\n",
        __FILE__, strerror(errno));
    }
}
now=time(NULL);
}
printf("server timeout, %li - seconds passed\n", (now-start));
close(readfd);

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

exit(0);
}
```

Рис. 4.2: 2 часть кода

Изменила код в файле client.c в соответствии с заданием (рис. 4.3).



```
nasidorova@nasidorova:~/lab14
client.c [----] 0 L: [ 1+25 26/ 36] *(416 / 560b) 0123 0x07B
#include "common.h"

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

int
main()
{
    int msg, len, i;
    long int t;

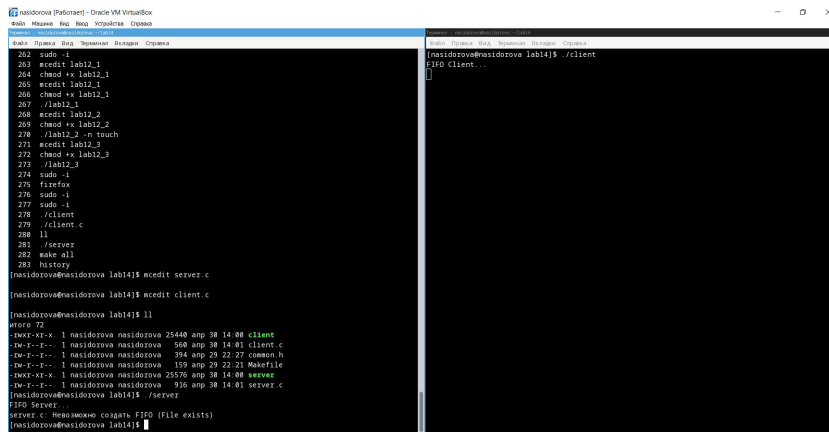
    for (i=0; i<20; i++){
        sleep(3);
        t=time(NULL);
        printf("FIFO Client...\n");

    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);
}
```

Рис. 4.3: Код

Открыла параллельно два окна терминала и запустила файлы server.c и client.c, они сработали верно (рис. 4.4).



```
nasidorova@nasidorova:~$ cat lab14.c
252 sudo -i
253 mcdit lab12_1
254 chmod +x lab12_1
255 mcdit lab12_1
256 chmod +x lab12_1
257 ./lab12_1
258 mcdit lab12_2
259 chmod +x lab12_2
260 ./lab12_2 -n touch
271 mcdit lab12_3
272 chmod +x lab12_3
273 ./lab12_3
274 sudo -i
275 firefox
276 sudo -i
277 sudo -i
278 ./client
279 ./client.c
280 ll
281 ./server
282 make all
283 history
[nasidorova@nasidorova lab14]$ mcdit server.c
[nasidorova@nasidorova lab14]$ mcdit client.c
[nasidorova@nasidorova lab14]$ ll
total 72
-rwxr-xr-x 1 nasidorova nasidorova 25600 apr 30 14:00 client
-rw-r--r-- 1 nasidorova nasidorova 560 apr 30 14:01 client.c
-rw-r--r-- 1 nasidorova nasidorova 394 apr 29 22:27 common.h
-rw-r--r-- 1 nasidorova nasidorova 159 apr 29 22:28 Makefile
-rwxr-xr-x 1 nasidorova nasidorova 25576 apr 30 14:00 server
-rw-r--r-- 1 nasidorova nasidorova 936 apr 30 14:01 server.c
[nasidorova@nasidorova lab14]$ ./server
server.c: неопределенный символ 'FIFO' (File exists)
[nasidorova@nasidorova lab14]$
```

Рис. 4.4: Работа файлов

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

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

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

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).

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);`

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

возвращается доступное число байтов 7. 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. 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 `SIGPIPE`, а вызов `write(2)` возвращает 0 с установкой ошибки (`errno=EPipe`) (если процесс не установил обработки сигнала `SIGPIPE`, производится обработка по умолчанию – процесс завершается).

Два и более процессов могут читать и записывать в канал.

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`.

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

5 Выводы

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

Список литературы