

# Front matter

---

lang: ru-RU title: "Отчет по лабораторной работе №15" subtitle: "по дисциплине: Операционные системы" author: "Трефилова Мария Андреевна"

## Formatting

---

toc-title: "Содержание" toc: true # Table of contents toc\_depth: 2 lof: false # List of figures lot: false # List of tables fontsize: 12pt linestretch: 1.5 papersize: a4paper documentclass: scrreprt polyglossia-lang: russian polyglossia-otherlangs: english mainfont: PT Serif romanfont: PT Serif sansfont: PT Sans monofont: PT Mono mainfontoptions: Ligatures=TeX romanfontoptions: Ligatures=TeX sansfontoptions: Ligatures=TeX,Scale=MatchLowercase monofontoptions: Scale=MatchLowercase indent: true pdf-engine: lualatex header-includes: - \linepenalty=10 # the penalty added to the badness of each line within a paragraph (no associated penalty node) Increasing the value makes tex try to have fewer lines in the paragraph. - \interlinepenalty=0 # value of the penalty (node) added after each line of a paragraph. - \hyphenpenalty=50 # the penalty for line breaking at an automatically inserted hyphen - \exhyphenpenalty=50 # the penalty for line breaking at an explicit hyphen - \binoppenalty=700 # the penalty for breaking a line at a binary operator - \relpenalty=500 # the penalty for breaking a line at a relation - \clubpenalty=150 # extra penalty for breaking after first line of a paragraph - \widowpenalty=150 # extra penalty for breaking before last line of a paragraph - \displaywidowpenalty=50 # extra penalty for breaking before last line before a display math - \brokenpenalty=100 # extra penalty for page breaking after a hyphenated line - \predisplaypenalty=10000 # penalty for breaking before a display - \postdisplaypenalty=0 # penalty for breaking after a display - \floatingpenalty = 20000 # penalty for splitting an insertion (can only be split footnote in standard LaTeX) - \raggedbottom # or \flushbottom - \usepackage{float} # keep figures where there are in the text

- \floatplacement{figure}{H} # keep figures where there are in the text

---

## Цель работы

---

Приобретение практических навыков работы с именованными каналами.

## Выполнение лабораторной работы

---

### Начальный этап

---

Изучила приведённые в тексте лабораторной программы server.c и client.c. Взяв данные примеры за образец, написала 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). Использовал функцию sleep() для приостановки работы клиента. 3. Сервер работает не бесконечно, а прекращает работу через некоторое время (например, 30 сек). Server.c

matrefilova [Работает] - Oracle VM VirtualBox

Файл Машина Вид Ввод Устройства Справка

en Вс, 18:38

Виртуальная машина сообщает, что гостевая ОС поддерживает интеграцию указателя мыши. Это

matrefilova@matrefilova:~/lab15

Файл Правка Вид Поиск Терминал Справка

```
#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": It is impossible (%s)\n",
            __FILE__, strerror(errno));
        exit(-1);
    }
    if ((readfd = open(FIFO_NAME, O_RDONLY)) < 0)
    {
        fprintf(stderr, "%s": It is impossible (%s)\n",
            __FILE__, strerror(errno));
        exit(-2);
    }
    clock_t now=time(NULL), start=time(NULL);
    while (now-start<30)
```

1,1 Наверху

matrefilova@matrefilova:~/lab15

matrefilova [Работает] - Oracle VM VirtualBox

Файл Машина Вид Ввод Устройства Справка

en Вс, 18:39

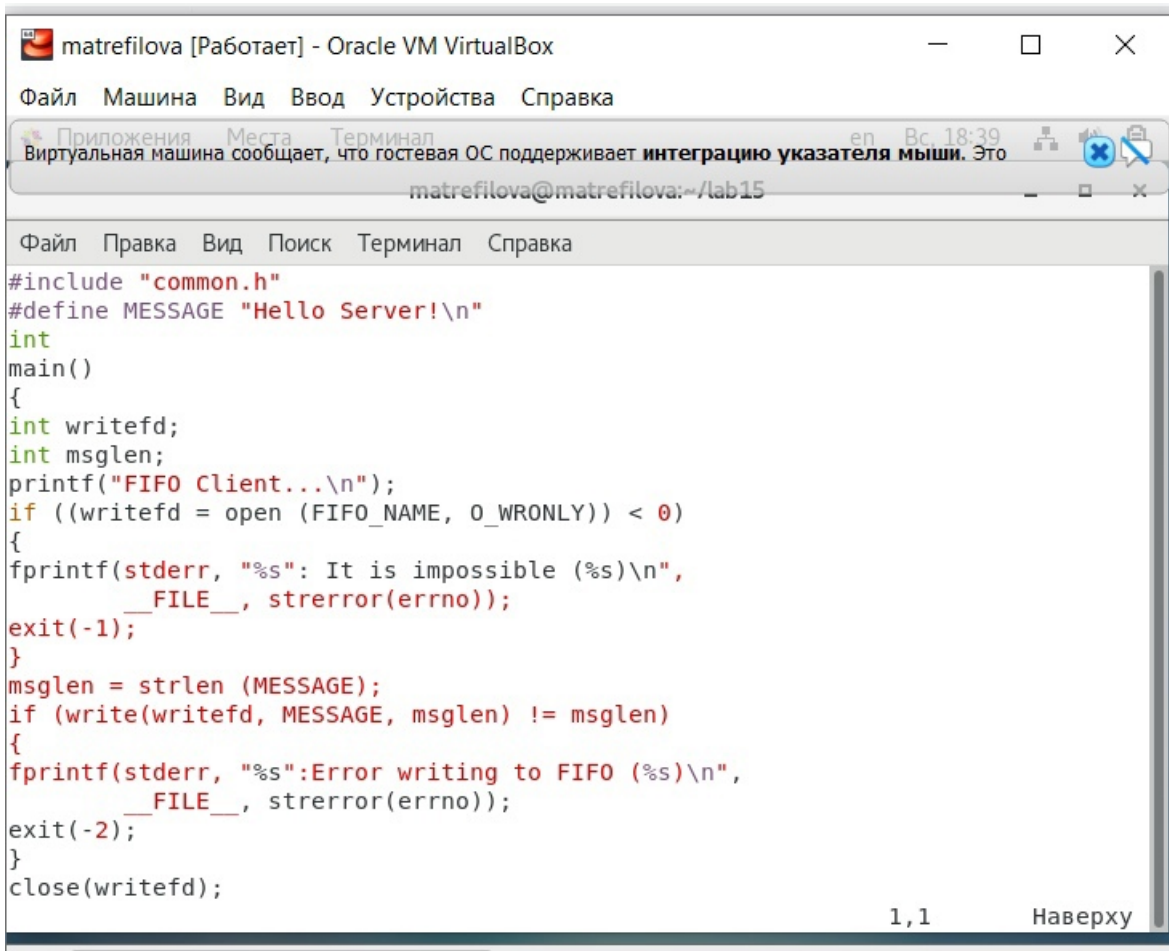
Виртуальная машина сообщает, что гостевая ОС поддерживает интеграцию указателя мыши. Это

matrefilova@matrefilova:~/lab15

Файл Правка Вид Поиск Терминал Справка

```
while (now-start<30)
{
    while((n=read(readfd, buff, MAX_BUFF)) > 0)
    {
        if(write(1, buff, n) != n)
        {
            fprintf(stderr, "%s":Output error (%s)\n",
                __FILE__, strerror(errno));
            exit(-3);
        }
    }
    now=time(NULL);
}
printf("\n.....\nserver timeout\n%u seconds passed!\n.....\n",now-start);
close(readfd);
if(unlink(FIFO_NAME) < 0)
{
    fprintf(stderr, "%s": It is impossible (%s)\n",
        __FILE__, strerror(errno));
    exit(-4);
}
exit(0);
```

22,1 84%



matrefilova [Работает] - Oracle VM VirtualBox

Файл Машина Вид Ввод Устройства Справка

Приложения Места Терминал en Вс, 18:39

Виртуальная машина сообщает, что гостевая ОС поддерживает интеграцию указателя мыши. Это

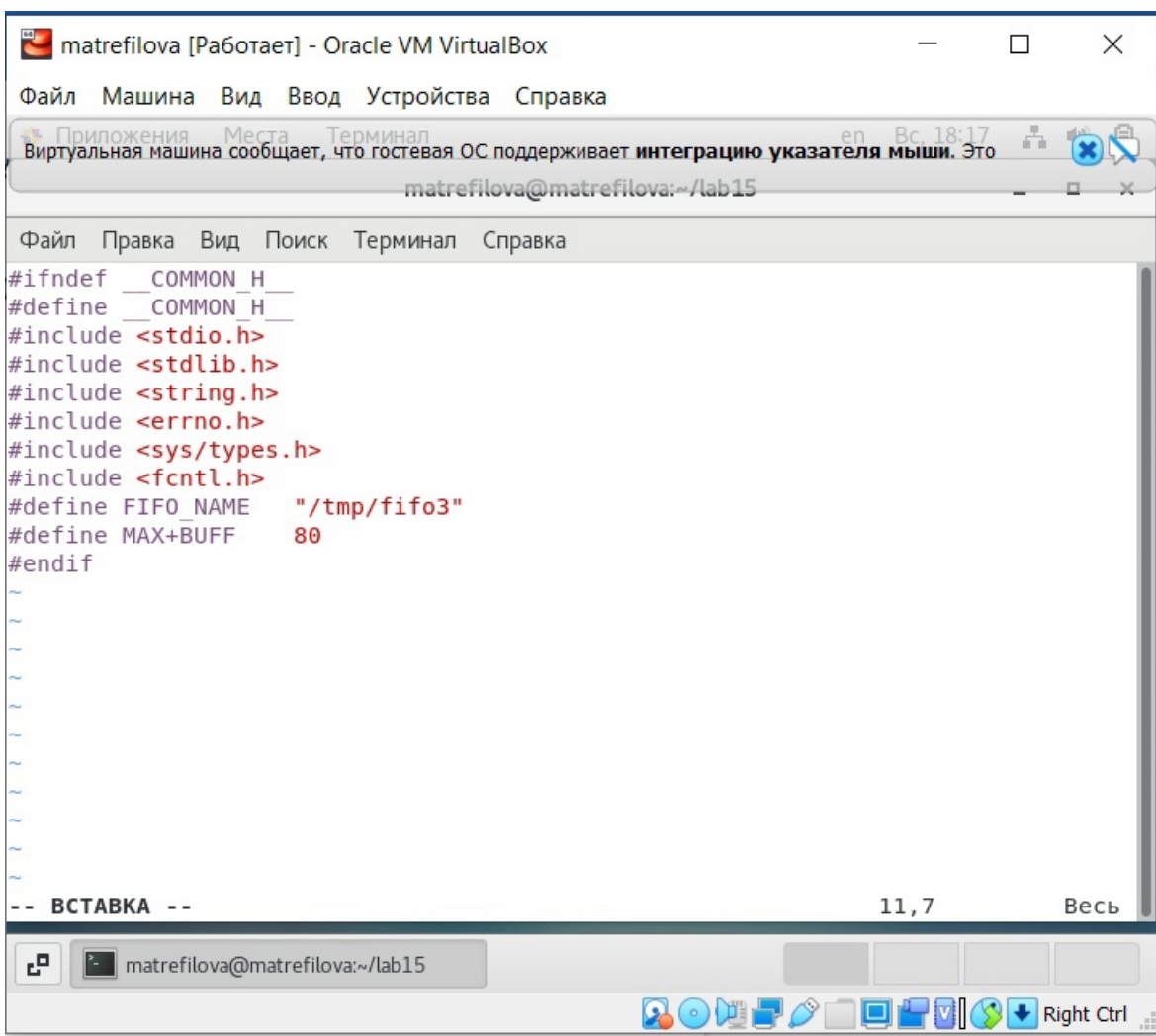
matrefilova@matrefilova:~/lab15

Файл Правка Вид Поиск Терминал Справка

```
#include "common.h"
#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": It is impossible (%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);
    }
    close(writefd);
}
```

1,1 Наверху

Common.h



matrefilova [Работает] - Oracle VM VirtualBox

Файл Машина Вид Ввод Устройства Справка

Приложения Места Терминал en Вс, 18:17

Виртуальная машина сообщает, что гостевая ОС поддерживает интеграцию указателя мыши. Это

matrefilova@matrefilova:~/lab15

Файл Правка Вид Поиск Терминал Справка

```
#ifndef __COMMON_H__
#define __COMMON_H__
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <errno.h>
#include <sys/types.h>
#include <fcntl.h>
#define FIFO_NAME "/tmp/fifo3"
#define MAX+BUFF 80
#endif
~
~
~
~
~
~
~
~
-- ВСТАВКА --
```

11,7 Весь

matrefilova@matrefilova:~/lab15

Right Ctrl

Проверила работу программы. При завершении программы каналы автоматически закрываются

## Выводы

---

Таким образом, я приобрела практические навыки работы с именованными каналами.

## Контрольные вопросы

---

1. Именованные каналы отличаются от неименованных наличием идентификатора канала, который представлен как специальный файл (соответственно имя именованного канала — это имя файла).
2. Создание неименованного канала из командной строки возможно командой `pipe`.
3. Создание именованного канала из командной строки возможно с помощью `mkfifo`.
4. 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. 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. 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=EP1PE`) (если процесс не установил обработки сигнала `SIGPIPE`, производится обработка по умолчанию — процесс завершается).
8. Два и более процессов могут читать и записывать в канал.
9. 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. Строковая функция `strerror` - функция языков C/C++, транслирующая код ошибки, который обычно хранится в глобальной переменной `errno`, в сообщение об ошибке, понятном человеку