

отчёта по лабораторной работе 15

Операционные системы

Саинт Амур Измаэль

Содержание

0.1	Цель работы:	4
-----	------------------------	---

List of Tables

List of Figures

0.1 Цель работы:

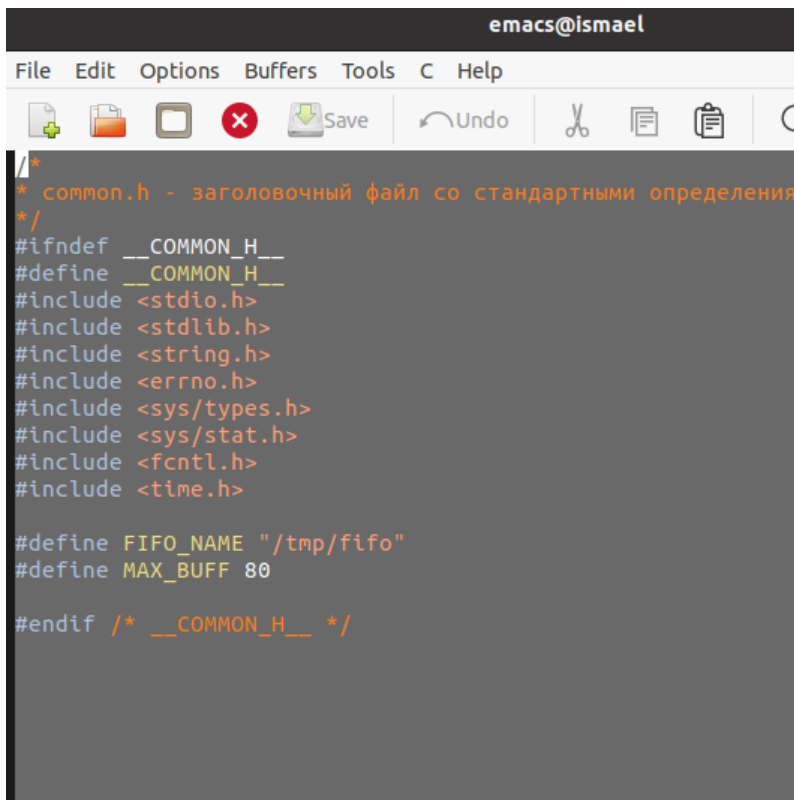
Приобретение практических навыков работы с именованными каналами. ## Ход работы: 1. Изучил приведённые в тексте программы `server.c` и `client.c` и взяла данные примеры за образец. (рис. ??)

2. Написал аналогичные программы, внося следующие изменения:

- работает не 1 клиент, а несколько (например, два).
- клиенты передают текущее время с некоторой периодичностью (например, раз в пять секунд). Использовала функцию `sleep()` для приостановки работы клиента.
- сервер работает не бесконечно, а прекращает работу через некоторое время (например, 30 сек). Использовала функцию `clock()` для определения времени

работы сервера. common.h: (рис. ??)

fig:002 width=70%}

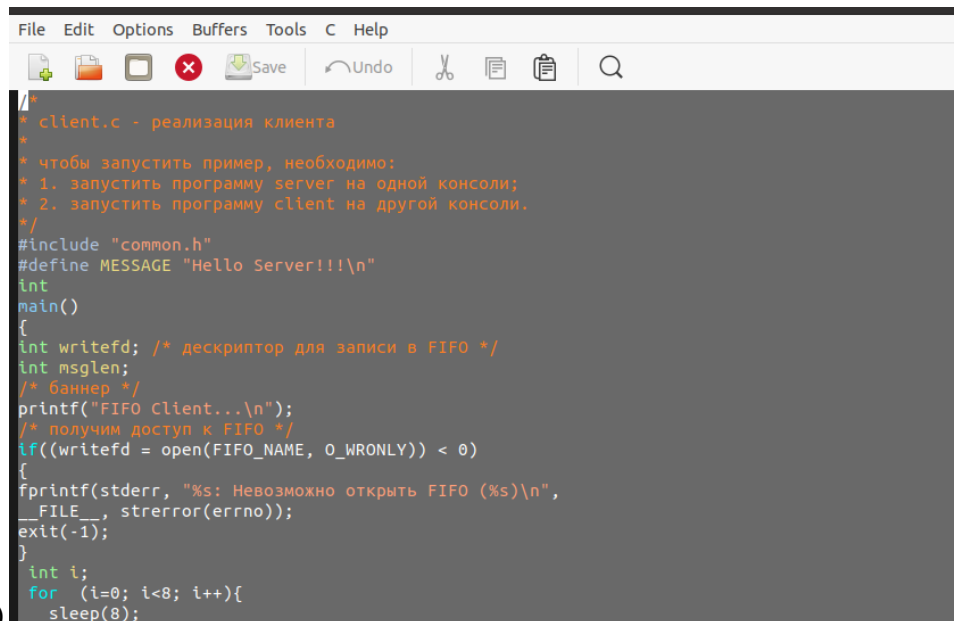


```
/*
 * common.h - заголовочный файл со стандартными определениями
 */
#ifndef __COMMON_H__
#define __COMMON_H__
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <errno.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <time.h>

#define FIFO_NAME "/tmp/fifo"
#define MAX_BUFF 80

#endif /* __COMMON_H__ */
```

- server.c: (рис. ??) (рис. ??)



```
/*
 * client.c - реализация клиента
 *
 * чтобы запустить пример, необходимо:
 * 1. запустить программу server на одной консоли;
 * 2. запустить программу client на другой консоли.
 */
#include "common.h"
#define MESSAGE "Hello Server!!!\n"
int
main()
{
    int writefd; /* дескриптор для записи в FIFO */
    int msglen;
    /* 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);
    }
    int i;
    for (i=0; i<8; i++){
        sleep(8);
    }
```

```

File Edit Options Buffers Tools C Help
[Icons: New, Open, Save, Undo, Cut, Copy, Paste, Find]

fprintf(stderr, "%s: Невозможно открыть FIFO (%s)\n",
__FILE__, strerror(errno));
exit(-1);
}
int i;
for (i=0; i<8; i++){
    sleep(8);
    long ttime = time(NULL);
    msglen = strlen(ctime(&ttime));

    if(write(writefd, ctime(&ttime), msglen) != msglen)
    {
        fprintf(stderr, "%s: Ошибка записи в FIFO (%s)\n",
        __FILE__, strerror(errno));
        exit(-2);
    }
}
/* закроем доступ к FIFO */
close(writefd);
exit(0);
}

```

fig:003 width=70%}

fig:004 width=70%}

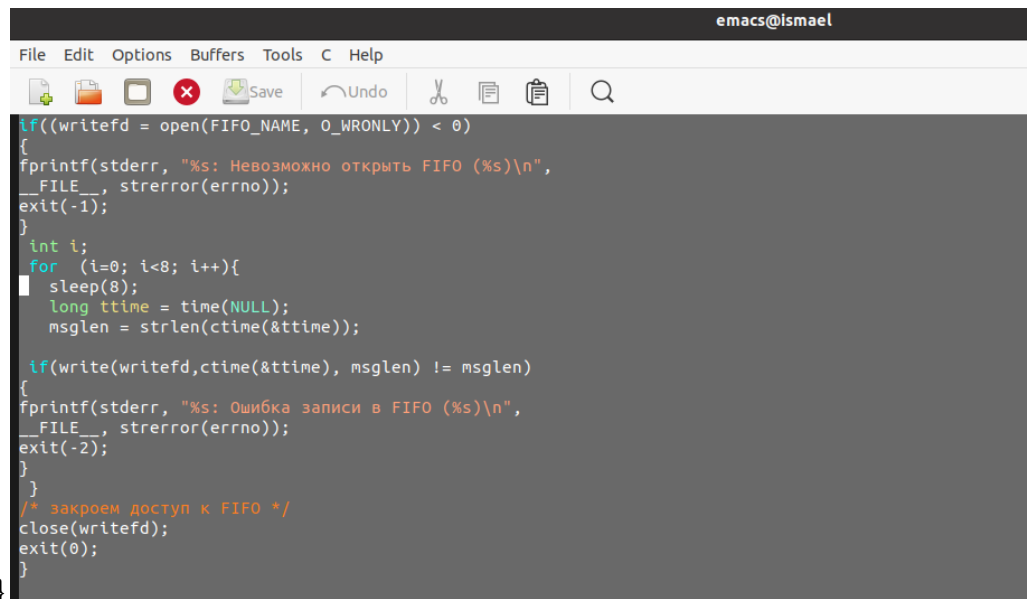
```

emacs@ism
File Edit Options Buffers Tools C Help
[Icons: New, Open, Save, Undo, Cut, Copy, Paste, Find]

/*
 * client.c - реализация клиента
 *
 * чтобы запустить пример, необходимо:
 * 1. запустить программу server на одной консоли;
 * 2. запустить программу client на другой консоли.
 */
#include "common.h"
#define MESSAGE "Hello Server!!!\n"
int
main()
{
    int writefd; /* дескриптор для записи в FIFO */
    int msglen;
    /* 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);
    }
    int i;
    for (i=0; i<8; i++){
        sleep(8);

```

- client.c:(рис. ??) (рис. ??)



The screenshot shows the Emacs editor interface with the title bar 'emacs@ismael'. The menu bar includes 'File', 'Edit', 'Options', 'Buffers', 'Tools', 'C', and 'Help'. The toolbar contains icons for file operations and editing. The code in the buffer is as follows:

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

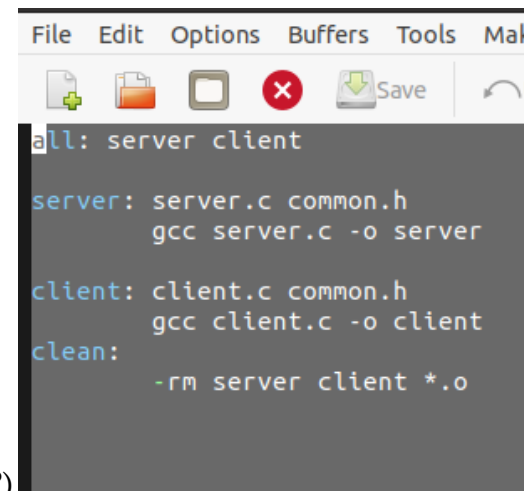
int i;
for (i=0; i<8; i++){
    sleep(8);
    long ttime = time(NULL);
    msglen = strlen(ctime(&ttime));

    if(write(writefd, ctime(&ttime), msglen) != msglen)
    {
        fprintf(stderr, "%s: Ошибка записи в FIFO (%s)\n",
            __FILE__, strerror(errno));
        exit(-2);
    }
}

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

fig:005 width=70%}

fig:006 width=70%}



The screenshot shows a terminal window with the following content:

```
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
```

- я создал Makefile и проверить (рис. ??)(рис. ??) (рис. ??)

```
marc@ismael:~/work/os/prog_lab1$ touch Makefile
marc@ismael:~/work/os/prog_lab1$ emacs Makefile
```

fig:007 width=70%}

```

marc@ismael:~/work/os/prog_lab1$ make
gcc client.c -o client
client.c: In function 'main':
client.c:26:4: warning: implicit declaration of function 'sleep' [-Wimplicit-
function-declaration]
   26 |     sleep(8);
      |     ~~~~~
client.c:30:5: warning: implicit declaration of function 'write'; did you m
fwrite'? [-Wimplicit-function-declaration]
   30 |     if(write(writefd, ctime(&tttime), msglen) != msglen)
      |        ~~~~~
      |        fwrite
client.c:38:1: warning: implicit declaration of function 'close'; did you m
pclose'? [-Wimplicit-function-declaration]
   38 |     close(writefd);
      |     ~~~~~
      |     pclose

```

fig:008 width=70%}

fig:009 width=70%}

```

marc@ismael:~/work/os/prog_lab1$ ./serv
FIFO Server...
Wed Jun  9 14:14:53 2021
Wed Jun  9 14:14:57 2021
Wed Jun  9 14:15:01 2021
Wed Jun  9 14:15:05 2021
Wed Jun  9 14:15:09 2021
Wed Jun  9 14:15:13 2021
Wed Jun  9 14:15:17 2021
Wed Jun  9 14:15:21 2021
Wed Jun  9 14:15:25 2021
Wed Jun  9 14:15:29 2021
Wed Jun  9 14:15:33 2021
Wed Jun  9 14:15:37 2021
Wed Jun  9 14:15:41 2021
Wed Jun  9 14:15:45 2021
Wed Jun  9 14:15:49 2021
Wed Jun  9 14:15:53 2021

```

- результаты (рис.-fig. ??) рис. ??) (рис. ??)

```

marc@ismael:~/work/os/prog_lab1$ ./client
FIFO Client...
marc@ismael:~/work/os/prog_lab1$ █

```

fig:0010 width=70%}


```
marc@ismael: ~/work... x marc@ismael: ~/work... x marc@ismael: ~/  
marc@ismael:~/work/os/prog_lab1$ ./client  
FIFO Client...  
marc@ismael:~/work/os/prog_lab1$
```

fig:0011 width=70%}

fig:0012 width=70%} В случае, если сервер завершит работу, не закрыв канал, файл FIFO не удалится, поэтому его в следующий раз создать будет нельзя и вылезет ошибка, следовательно, работать ничего не будет.

Вывод: приобрел практические навыки работы с именованными каналами.

Ответы на 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, гарантировано 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.