

Лабораторная работа №14

Именованные каналы

Захаренко Анастасия Викторовна

Содержание

Цель работы	5
Задание	6
Теоретическое введение	7
Выполнение лабораторной работы	8
Выводы	17

Список иллюстраций

0.1	команды	8
0.2	файлы	9
0.3	common.h	10
0.4	server.c	11
0.5	server.c	12
0.6	server.c	13
0.7	client.c	14
0.8	client.c	14
0.9	Makefile	15
0.10	make all	15
0.11	./server	16
0.12	./client	16

Список таблиц

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

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: общеоуниковые (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

Выполнение лабораторной работы

1. Я создала необходимые для работы файлы(common.h, server.c, client.c, Makefile)

```
[avzakharenko@fedora lab14]$ touch common.h server.c client.c Makefile  
[avzakharenko@fedora lab14]$ touch Makefile
```

Рис. 0.1: команды









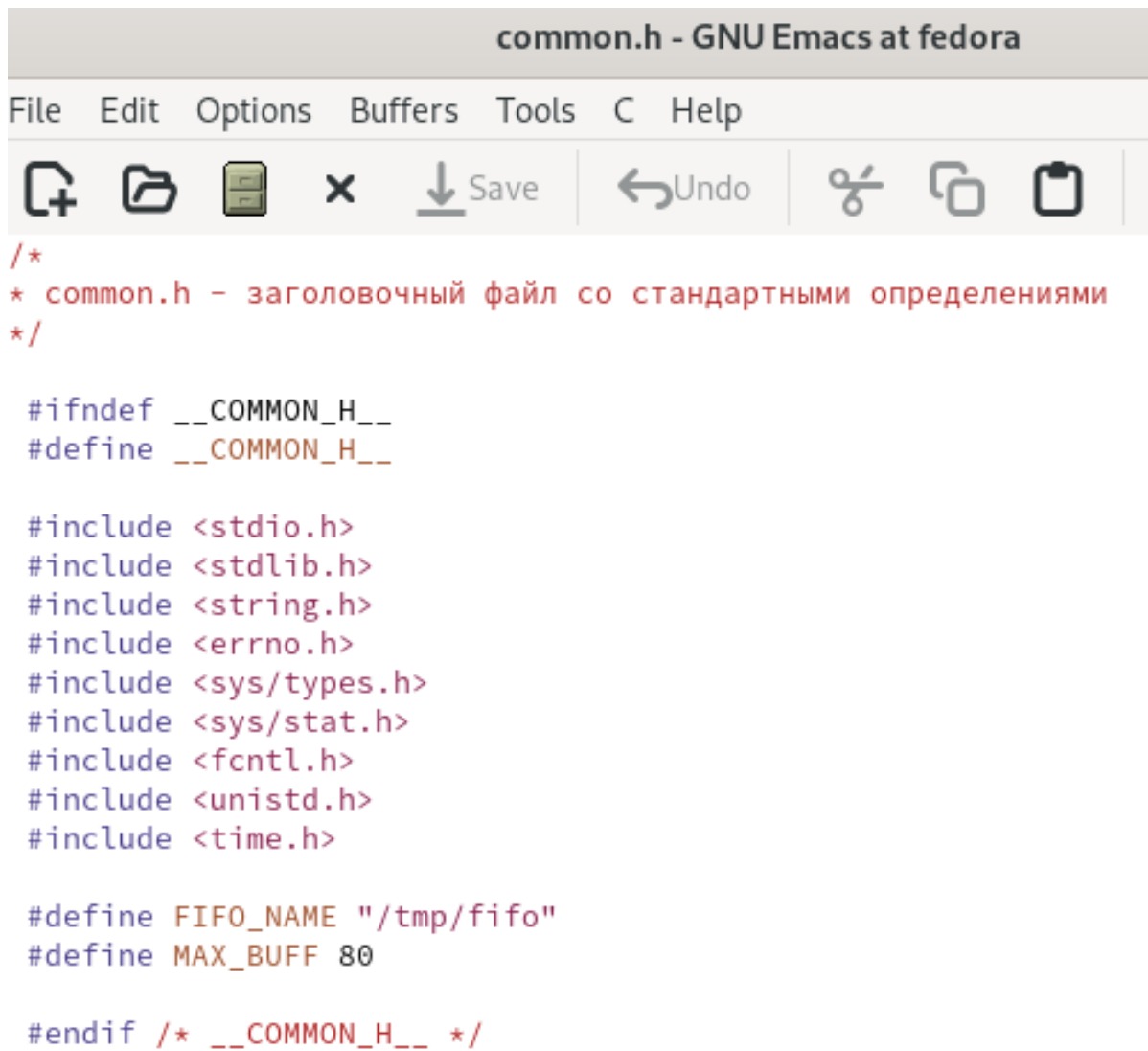
	client	25,5 кБ	21:19	☆
	client.c	1,2 кБ	21:13	☆
	common.h	426 байт	20:40	☆
	Makefile	156 байт	21:19	☆
	presentation	3 объекта	24 фев	☆
	report	5 объектов	24 фев	☆
	server	25,5 кБ	21:19	☆
	server.c	1,6 кБ	20:49	☆

Рис. 0.2: файлы

- Далее я скопировала коды из лабораторной и дополнила их в соответствии с заданием.
- common.h: добавила стандартные заголовочные файлы unistd.h, time.h.



```
/*
 * common.h - заголовочный файл со стандартными определениями
 */

#ifndef __COMMON_H__
#define __COMMON_H__

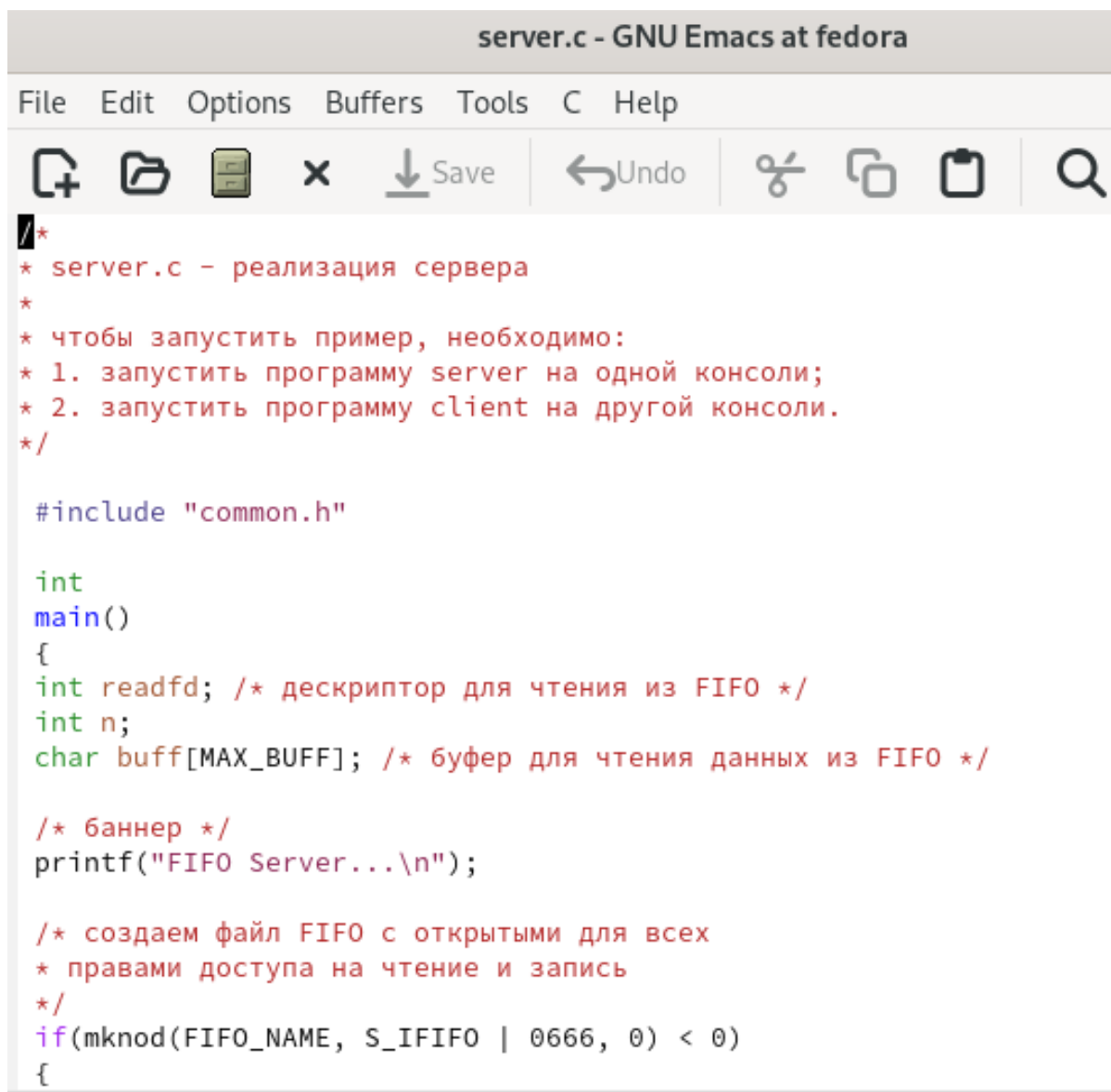
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <errno.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <unistd.h>
#include <time.h>

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

#endif /* __COMMON_H__ */
```

Рис. 0.3: common.h

4. server.c: добавила цикл для контроля за временем работы сервера.



```
server.c - GNU Emacs at fedora
File Edit Options Buffers Tools C Help

/*
 * server.c - реализация сервера
 *
 * чтобы запустить пример, необходимо:
 * 1. запустить программу server на одной консоли;
 * 2. запустить программу client на другой консоли.
 */

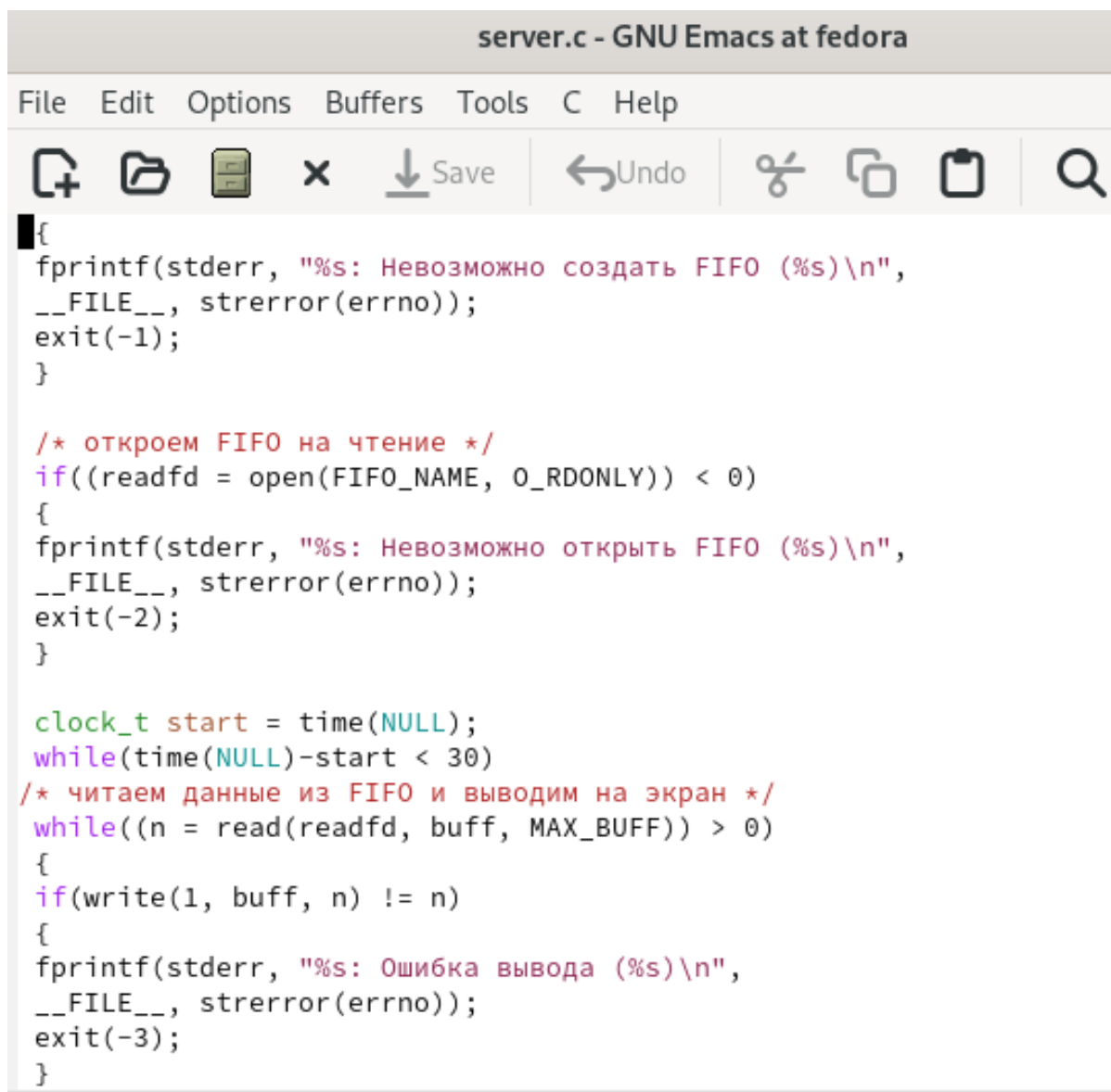
#include "common.h"

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)
    {
```

Рис. 0.4: server.c

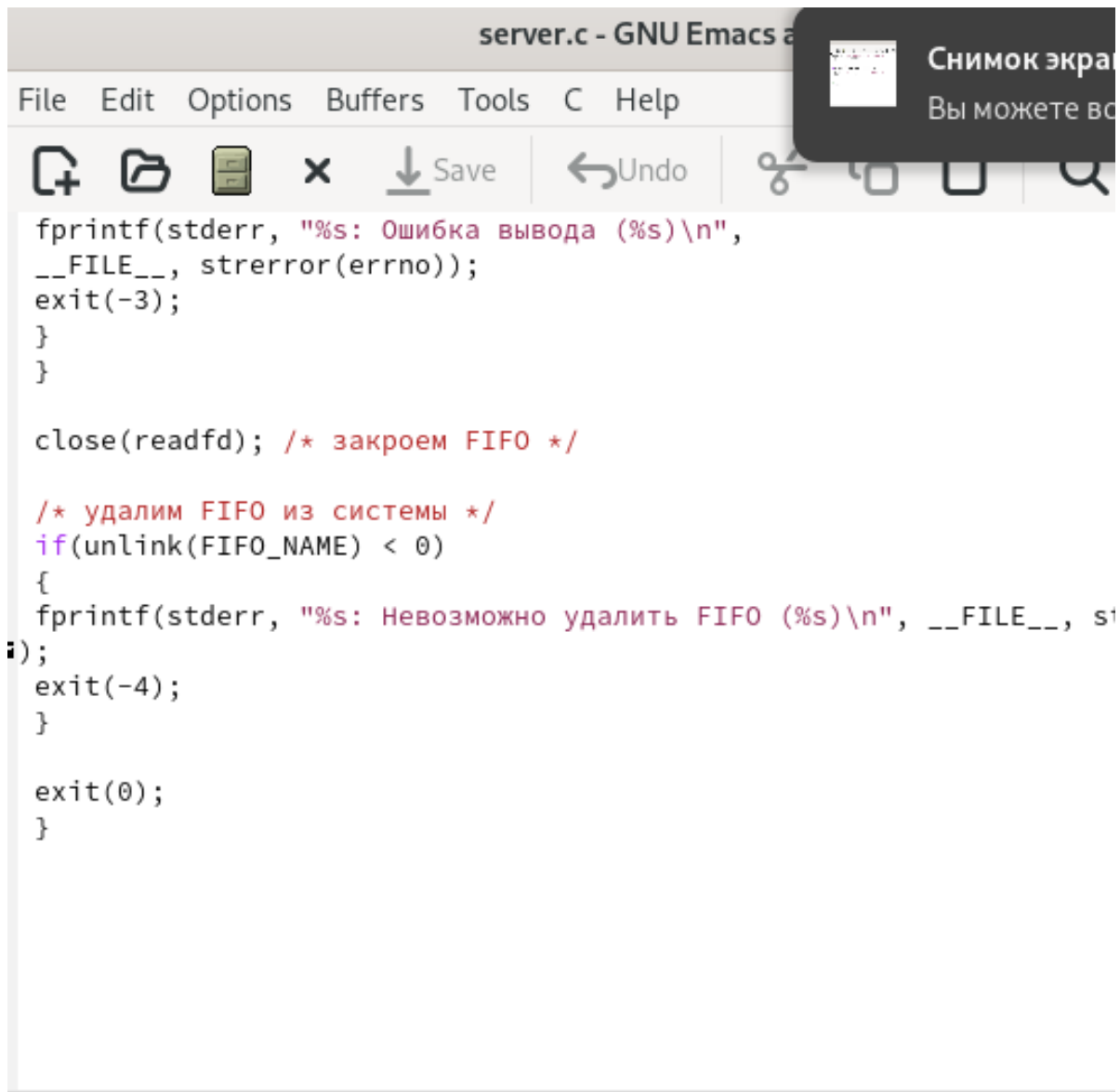


```
{
fprintf(stderr, "%s: Невозможно создать FIFO (%s)\n",
__FILE__, strerror(errno));
exit(-1);
}

/* 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 start = time(NULL);
while(time(NULL)-start < 30)
/* читаем данные из FIFO и выводим на экран */
while((n = read(readfd, buff, MAX_BUFF)) > 0)
{
if(write(1, buff, n) != n)
{
fprintf(stderr, "%s: Ошибка вывода (%s)\n",
__FILE__, strerror(errno));
exit(-3);
}
}
```

Рис. 0.5: server.c



```
server.c - GNU Emacs a
File Edit Options Buffers Tools C Help
[Icons] Save Undo [Icons]
fprintf(stderr, "%s: Ошибка вывода (%s)\n",
__FILE__, strerror(errno));
exit(-3);
}
}

close(readfd); /* закроем FIFO */

/* удалим FIFO из системы */
if(unlink(FIFO_NAME) < 0)
{
fprintf(stderr, "%s: Невозможно удалить FIFO (%s)\n", __FILE__, st
);
exit(-4);
}

exit(0);
}
```

Рис. 0.6: server.c

5. client.c: добавила цикл, отвечающий за кол-во сообщений о текущем времени.

```

#include "common.h"

int main()
{
    int writefd; /* дескриптор для записи в FIFO */
    int msglen;

    /* баннер */
    printf("FIFO Client...\n");

    for(int i=0; i<4; i++)
    {
        /* 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);
        }

        long int ttime = time(NULL);
        char* text = ctime(&ttime);
        /* передадим сообщение серверу */
        msglen = strlen(text);
        if(write(writefd, text, msglen) != msglen)

```

Рис. 0.7: client.c

```

        if(write(writefd, text, msglen) != msglen)
        {
            fprintf(stderr, "%s: Ошибка записи в FIFO (%s)\n", __FILE__, strerror(errno));
            exit(-2);
        }
        sleep (5);
    }

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

```

Рис. 0.8: client.c

6. Makefile: поставила таб вместо пробелов в 4 и тд. строке

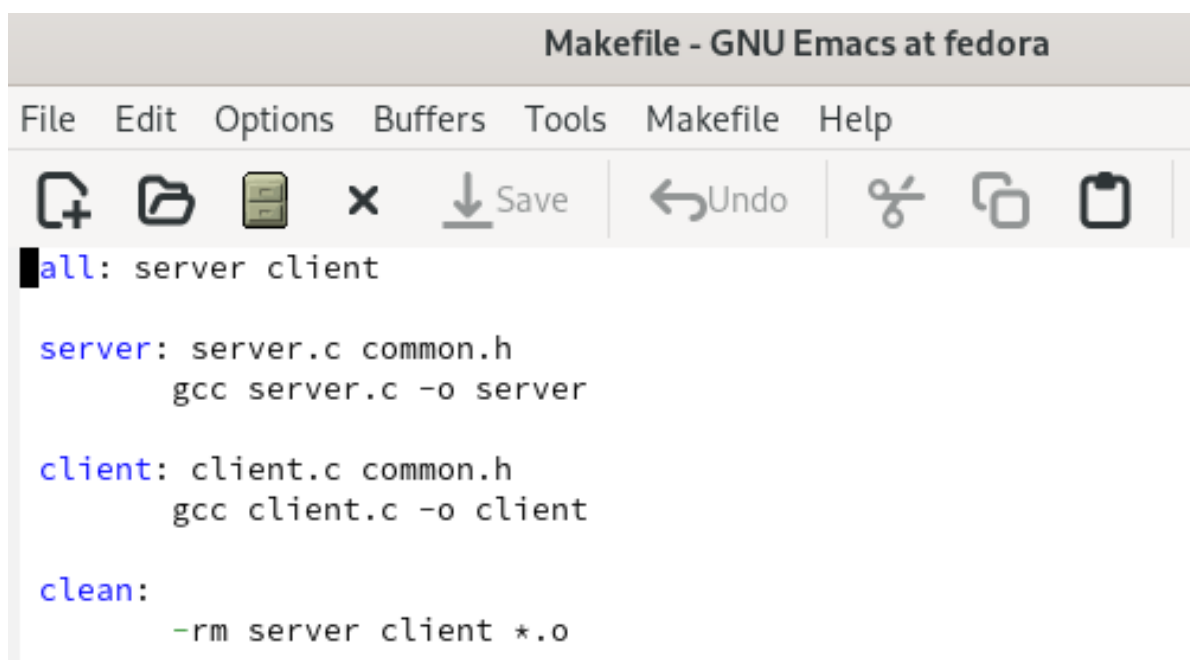


Рис. 0.9: Makefile

7. Далее скомпилировала все файлы

```
[avzakharenko@fedora lab14]$ make all
Makefile:4: *** пропущен разделитель. Останов.
[avzakharenko@fedora lab14]$ make all
Makefile:4: *** пропущен разделитель (возможно нужен TAB вместо восьми пробелов?). Останов.
[avzakharenko@fedora lab14]$ make all
gcc server.c -o server
gcc client.c -o client
```

Рис. 0.10: make all

8. В конце я проверила работу сервера, используя команды: `./server`, `./client`

```
[avzakharenko@fedora lab14]$ ./server
FIFO Server...
Mon Apr 10 21:28:01 2023
Mon Apr 10 21:28:06 2023
Mon Apr 10 21:28:11 2023
Mon Apr 10 21:28:16 2023
[avzakharenko@fedora lab14]$
```

Рис. 0.11: ./server

```
[avzakharenko@fedora lab14]$ ./client
FIFO Client...
[avzakharenko@fedora lab14]$
```

Рис. 0.12: ./client

Выводы

Я приобрела практические навыки работы с именованными каналами.