отчёта по лабораторной работе 15

Операционные системы

Алхатиб Осама

Содержание

1 Отчет по лабораторной работе №15 **5**
1.0.1 Средства, применяемые при разработке программного обеспечения в ОС типа UNIX/Linux 5
1.1 Российский Университет Дружбы Народов 5
1.1.1 Факультет Физико-Математических и Естественных Наук . 5
1.2 Цель работы 5
1.3 Вывод 11

List of Tables

List of Figures

1 Отчет по лабораторной работе №15

1.0.1 Средства, применяемые при разработке программного обеспечения в ОС типа UNIX/Linux

---

1.1 Российский Университет Дружбы Народов

1.1.1 Факультет Физико-Математических и Естественных Наук

Дисциплина: Операционные системы

Студент: Алхатиб Осама

Группа: НПИбд-02-20

2021г.

1.2 Цель работы

Приобретение практических навыков работы с именованными каналами.

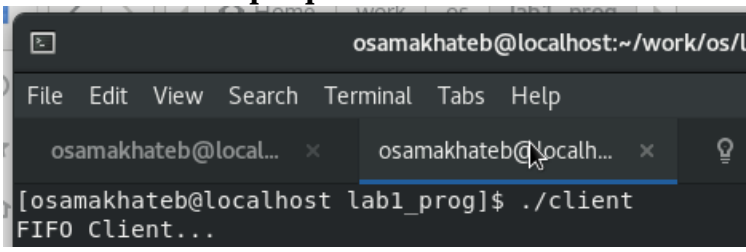
Ход работы 1,2,3. Работают 2 клиента, передают время раз в 5 и 4 секунды, сервер работает 20 секунд

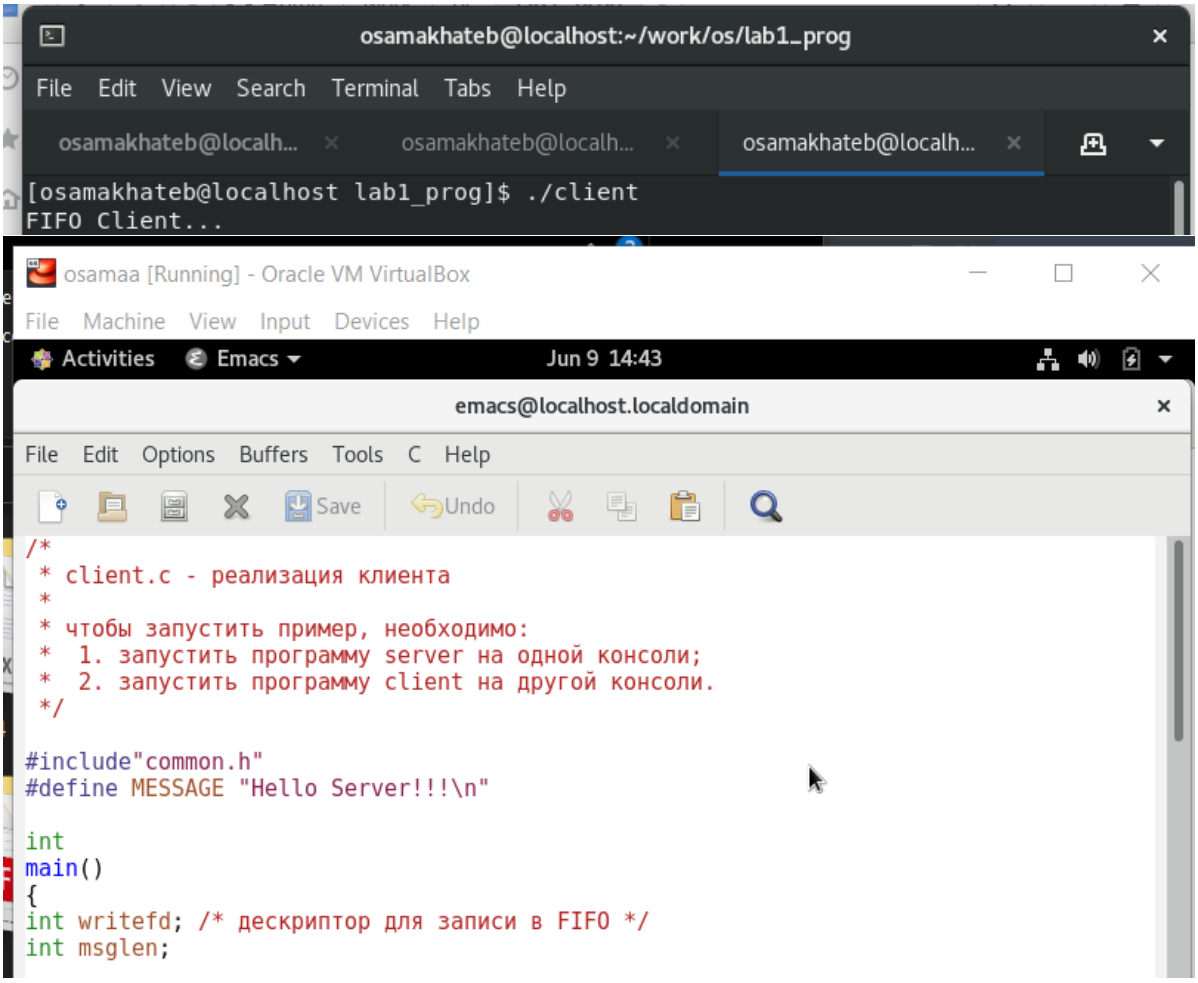
```
[osamakhateb@localhost ~]$ mkdir work
[osamakhateb@localhost ~]$ mkdir work/os
[osamakhateb@localhost ~]$ mkdir work/os/lab1_prog
[osamakhateb@localhost ~]$ cd work/os/lab1_prog
[osamakhateb@localhost lab1_prog]$ touch common.h
[osamakhateb@localhost lab1_prog]$ touch common.h server.c client.c makefile
[osamakhateb@localhost lab1_prog]$ emacs server.c
```

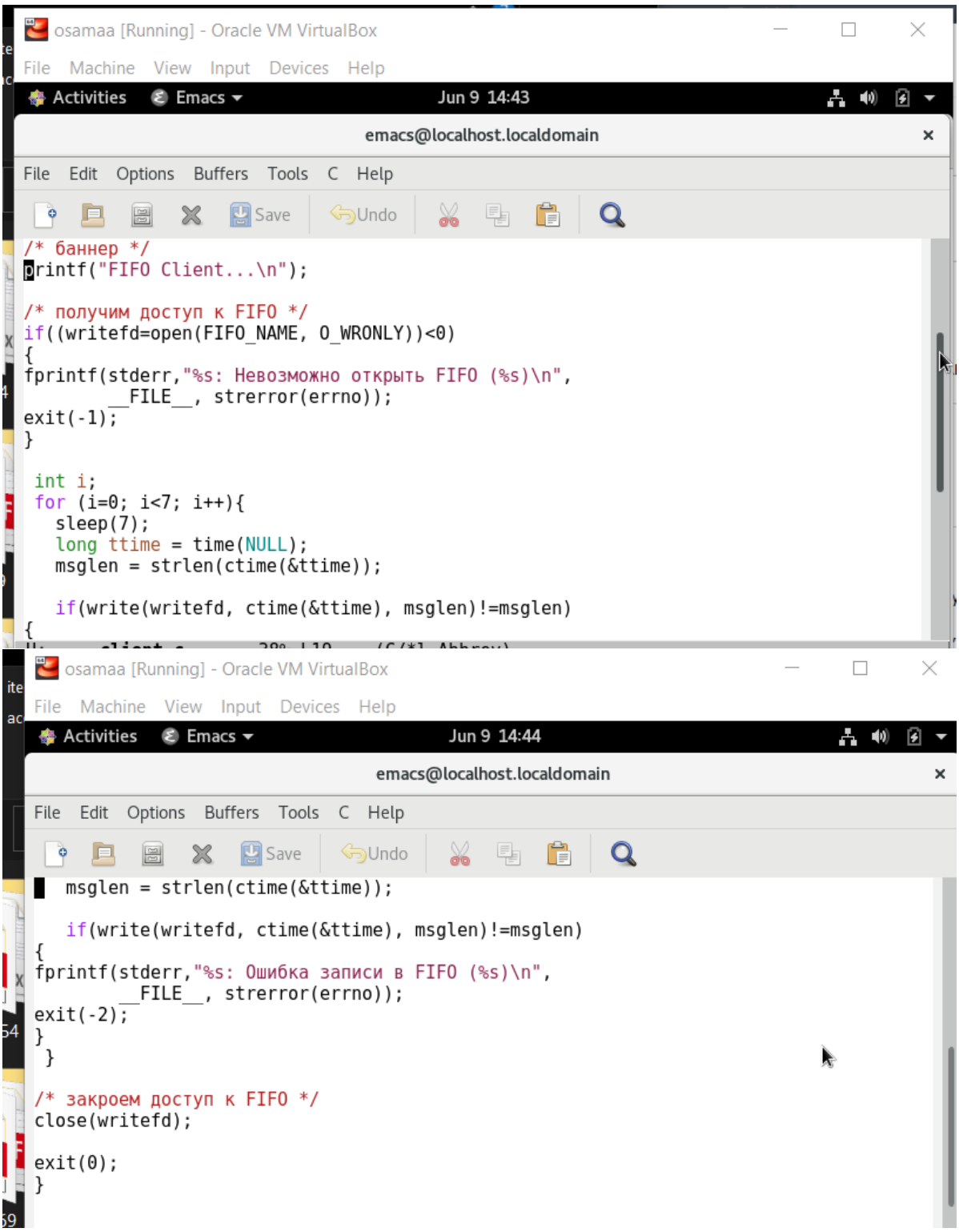
```
[osamakhateb@localhost lab1_prog]$ make
gcc client.c -o client
client.c: In function ‘main’:
client.c:31:4: warning: implicit declaration of function ‘sleep’; did you mean ‘
strsep’? [-Wimplicit-function-declaration]
    sleep(7);
    ^~~~~
    strsep
client.c:35:7: warning: implicit declaration of function ‘write’; did you mean ‘
fwrite’? [-Wimplicit-function-declaration]
    if(write(writefd, ctime(&ttime), msglen)!=msglen)
       ^~~~~
       fwrite
client.c:44:1: warning: implicit declaration of function ‘close’; did you mean ‘
pclose’? [-Wimplicit-function-declaration]
 close(writefd);
 ^~~~~
 pclose
```

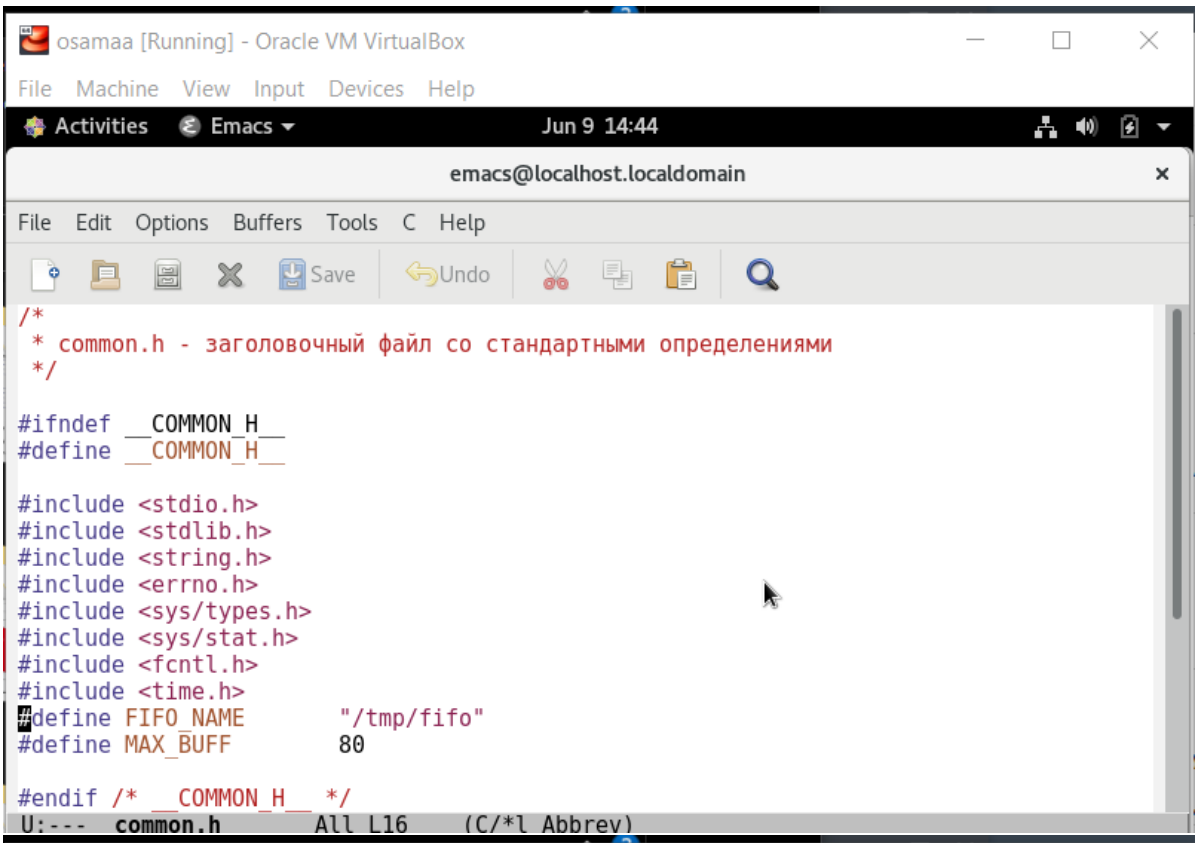
```
[osamakhateb@localhost lab1_prog]$ ./server
FIFO Server...
Wed Jun  9 14:38:23 2021
Wed Jun  9 14:38:25 2021
Wed Jun  9 14:38:30 2021
Wed Jun  9 14:38:32 2021
Wed Jun  9 14:38:37 2021
Wed Jun  9 14:38:39 2021
Wed Jun  9 14:38:44 2021
Wed Jun  9 14:38:46 2021
Wed Jun  9 14:38:51 2021
Wed Jun  9 14:38:53 2021
Wed Jun  9 14:38:58 2021
Wed Jun  9 14:39:00 2021
Wed Jun  9 14:39:05 2021
Wed Jun  9 14:39:07 2021
[osamakhateb@localhost lab1_prog]$
```

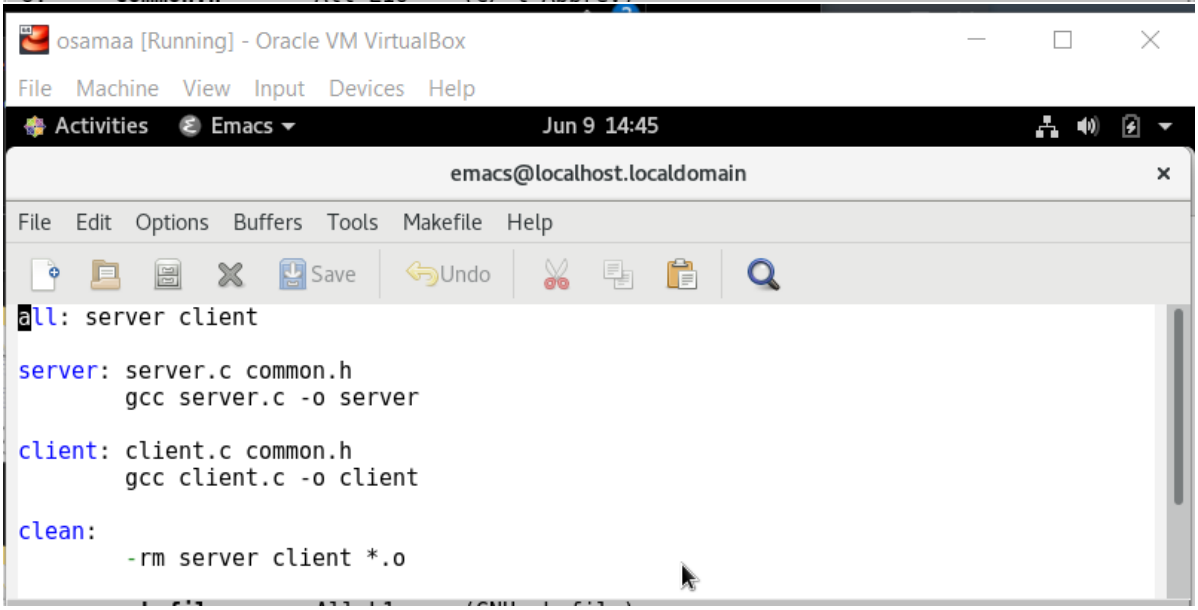
Сервер отключается и файл не существует, закрывая клиенты

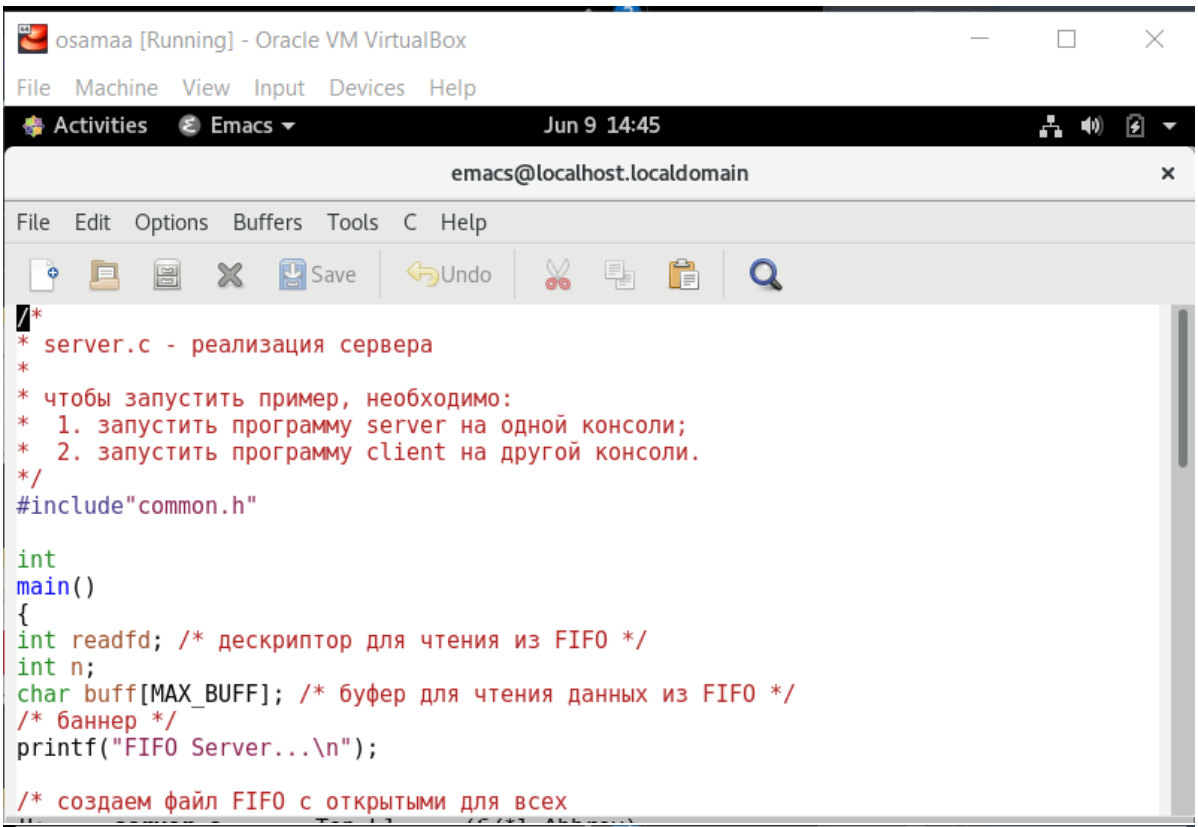


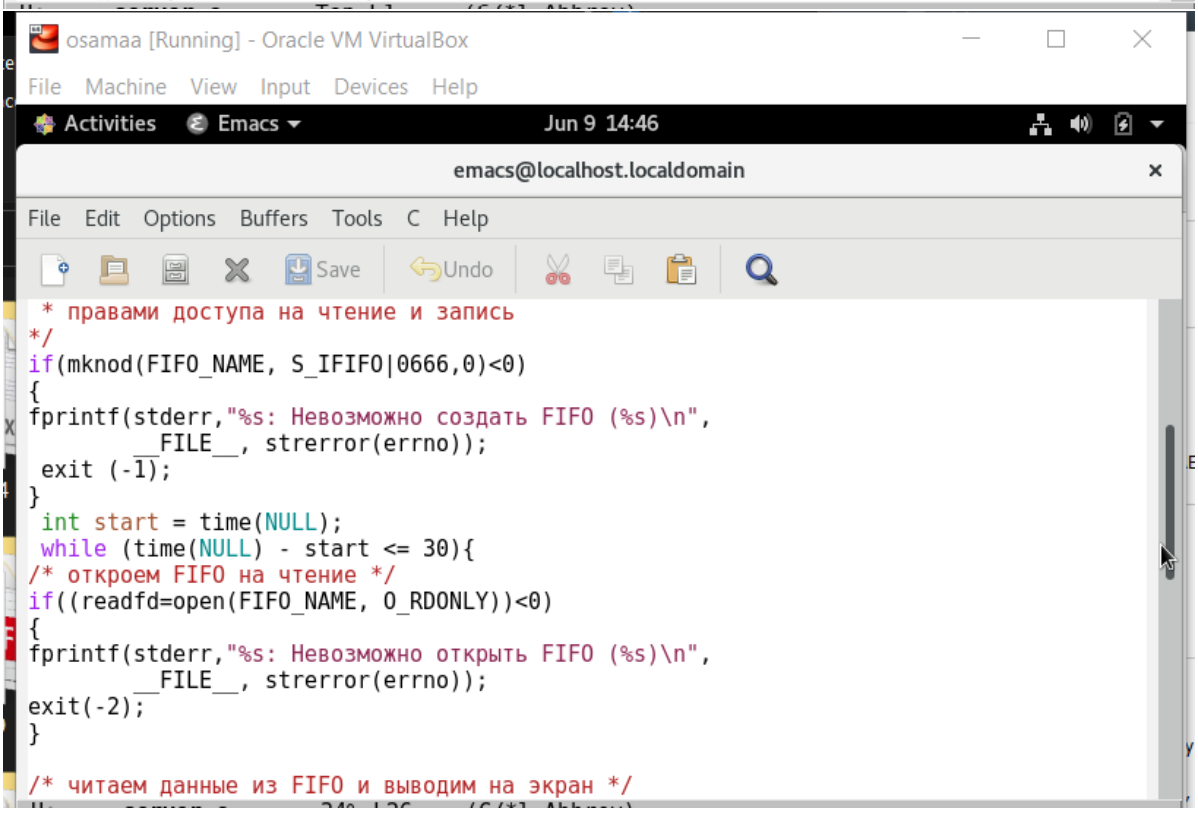

osamakhateb@localhost:~/work/os/lab1_prog
File Edit View Search Terminal Tabs Help
osamakhateb@localh...
osamakhateb@localh...
osamakhateb@localh...
[osamakhateb@localhost lab1_prog]$ ./client
FIFO Client...
osamaa [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities
Emacs
Jun 9 14:43
emacs@localhost.localdomain
File Edit Options Buffers Tools C Help
Save
Undo
/*
* 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;


osamaa [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Emacs Jun 9 14:43
emacs@localhost.localdomain
File Edit Options Buffers Tools C Help
Save Undo
/* 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<7; i++){
sleep(7);
long ttime = time(NULL);
msglen = strlen(ctime(&ttime));
if(write(writefd, ctime(&ttime), msglen)!=msglen)
{
osamaa [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Emacs Jun 9 14:44
emacs@localhost.localdomain
File Edit Options Buffers Tools C Help
Save Undo
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);
}


osamaa [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Emacs Jun 9 14:44
emacs@localhost.localdomain
File Edit Options Buffers Tools C Help
Save Undo
/*
* 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__ */
U:--- common.h All L16 (C/*l Abbrev)


osamaa [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Emacs Jun 9 14:45
emacs@localhost.localdomain
File Edit Options Buffers Tools Makefile Help
Save Undo
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


osamaa [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Emacs Jun 9 14:45
emacs@localhost.localdomain
File Edit Options Buffers Tools C Help
Save Undo
/*
* 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


osamaa [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Emacs Jun 9 14:46
emacs@localhost.localdomain
File Edit Options Buffers Tools C Help
Save Undo
 * правами доступа на чтение и запись
*/
if(mknod(FIFO_NAME, S_IFIFO|0666,0)<0)
{
fprintf(stderr,"%s: Невозможно создать FIFO (%s)\n",
          __FILE__, strerror(errno));
 exit (-1);
}
 int start = time(NULL);
 while (time(NULL) - start <= 30){
/* 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);
}
/* читаем данные из FIFO и выводим на экран */

1.3 Вывод

В результате работы , я приобрел практические навыки работы с именованными каналами

1.3.0.1 *Контрольные вопросы*

1. Именованные каналы отличаются от неименованных наличием идентификатора канала, который представлен как специальный файл (соответственно имя именованного канала — это имя файла).
2. Для создания неименованного канала используется системный вызов pipe. Массив из двух целых чисел является выходным параметром этого системного вызова.
3. Вы можете создавать именованные каналы из командной строки и внутри программы. С давних времен программой создания их в командной строке была команда: mknod - $ mknod имя_файла , однако команды mknod нет в списке команд X/Open, поэтому она включена не во все UNIX-подобные системы. Предпочтительнее применять в командной строке - $ mkfifo имя_файла.
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); 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, 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, гарантированно 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 многонаправленная 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