

Российский университет дружбы народов

факультет физико-математических и естественных наук

Отчет по лабораторной работе № 15

дисциплина : Операционные системы

студент Блохин александр НКН

Москва

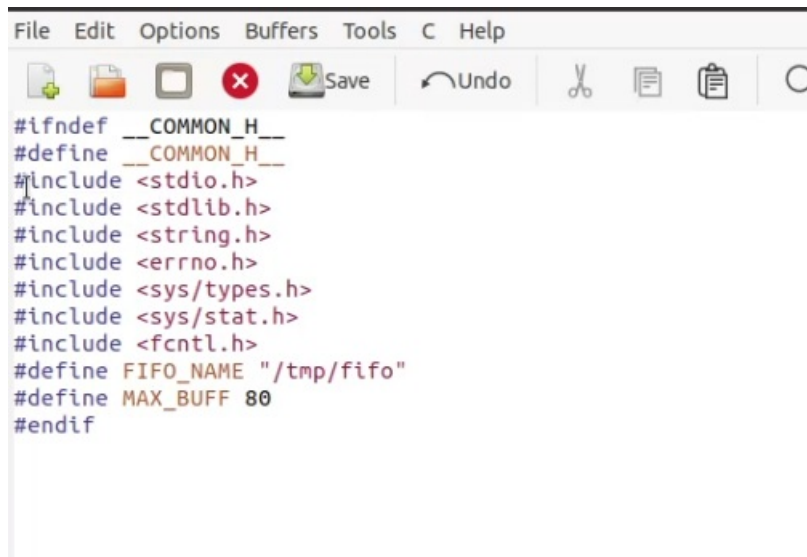
2021

Цель работы

Приобретение практических навыков работы с именованными каналами.

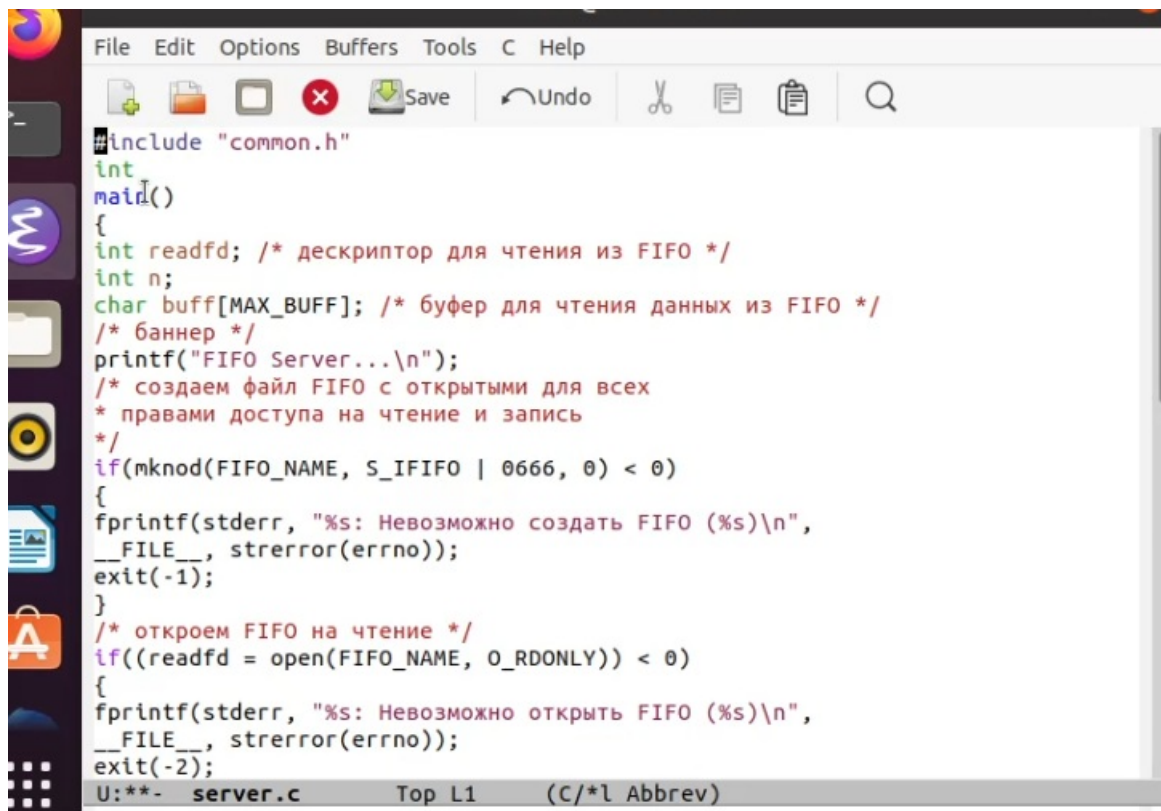
Ход работы

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

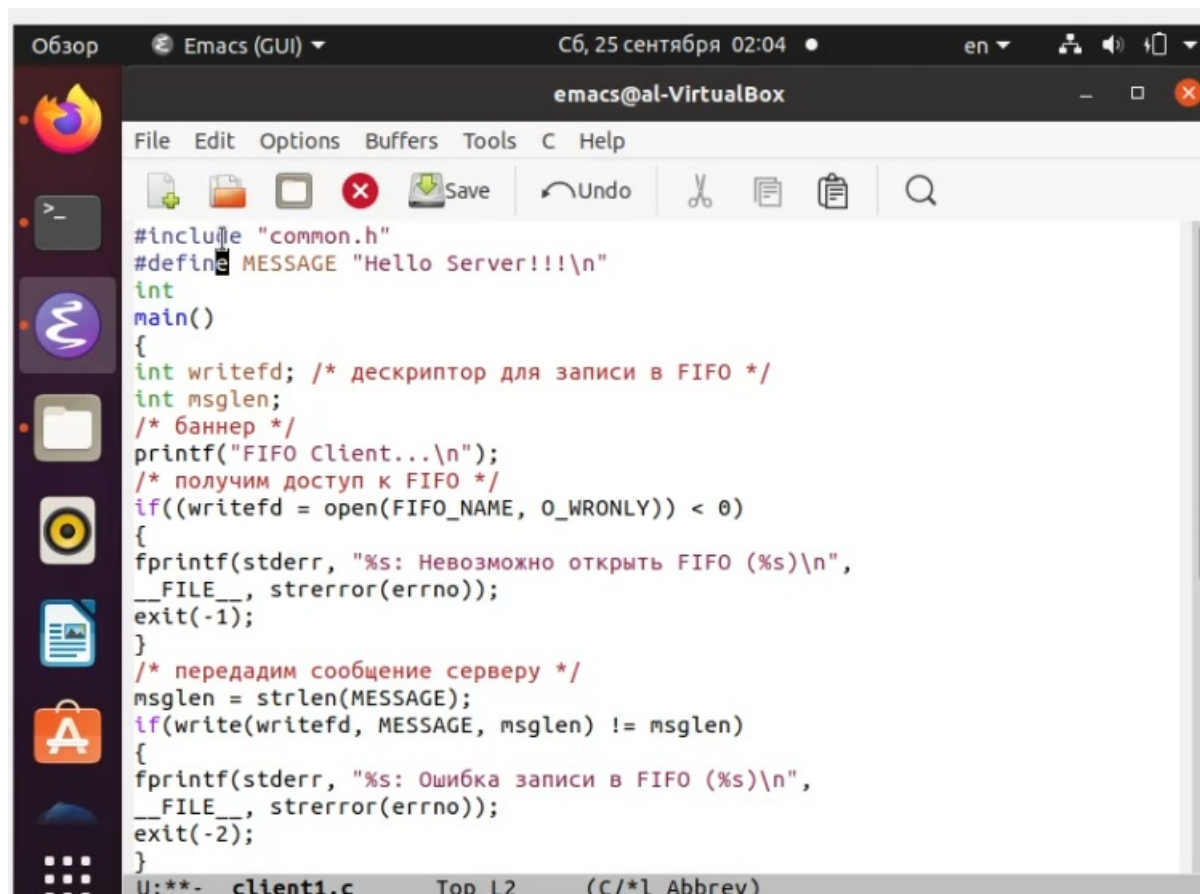


```
File Edit Options Buffers Tools C Help
[Icons: New, Open, Run, Close, Save, Undo, Cut, Copy, Paste, Find]

#ifdef __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>
#define FIFO_NAME "/tmp/fifo"
#define MAX_BUFF 80
#endif
```



```
#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)
    {
        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);
    }
}
U:***- server.c Top L1 (C/*l Abbrev)
```



```
Обзор Emacs (GUI) Сб, 25 сентября 02:04 en emacs@al-VirtualBox
File Edit Options Buffers Tools C Help
#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);
    }
    /* передадим сообщение серверу */
    msglen = strlen(MESSAGE);
    if(write(writefd, MESSAGE, msglen) != msglen)
    {
        fprintf(stderr, "%s: Ошибка записи в FIFO (%s)\n",
            __FILE__, strerror(errno));
        exit(-2);
    }
}
U:***- client1.c Top L2 (C/*l Abbrev)
```

```
#include "common.h"
#define MESSAGE "Hello Server!!!\n"
int
main()
{
    int writefd;
    int msglen;
    char message[10];
    int cout;
    long long int T;

    printf("FIFO Client...\n");
    if((writefd = open(FIFO_NAME, O_WRONLY)) < 0)
    {
        fprintf(stderr, "%s: Невозможно открыть FIFO (%s)\n",
            __FILE__, strerror(errno));
        exit(-1);
    }
    msglen = strlen(MESSAGE);
    if(write(writefd, MESSAGE, msglen) != msglen)
    {
        fprintf(stderr, "%s: Ошибка записи в FIFO (%s)\n",
            __FILE__, strerror(errno));
        exit(-2);
    }
}
```

```
all: server client1 client2

server: server.c common.h
       gcc server.c -o server

client1: client1.c common1.h
        gcc client1.c -o client1

client2: client2.c common2.h
        gcc client2.c -o client2

clean:
       -rm server client *.o
```

```
al@al-VirtualBox:~/lab15/llab$ sh server
./server
FIFO Server...
Sut Sep 25 02:45:38 2021
Sut Sep 25 02:45:39 2021
Sut Sep 25 02:45:41 2021
Sut Sep 25 02:45:43 2021
Sut Sep 25 02:45:43 2021
Sut Sep 25 02:45:45 2021
Sut Sep 25 02:45:46 2021
Sut Sep 25 02:45:48 2021
Sut Sep 25 02:45:48 2021
Sut Sep 25 02:45:50 2021
Sut Sep 25 02:45:52 2021
Sut Sep 25 02:45:55 2021
Sut Sep 25 02:45:57 2021
Sut Sep 25 02:46:01 2021
Sut Sep 25 02:46:04 2021
Sut Sep 25 02:46:08 2021

server timeout
30 seconds passed
```

```
al@al-VirtualBox:~/lab15/llab$ sh client2.
./client2
FIFO Client2...
FIFO Client2...
FIFO Client2...
FIFO Client2...
FIFO Client2...
FIFO Client2...
FIFO Client2...
FIFO Client2...
FIFO Client2...
FIFO Client2...
```

```
al@al-VirtualBox:~/lab15/llab$ sh client1
./client1
FIFO Client1...
FIFO Client1...
FIFO Client1...
FIFO Client1...
FIFO Client1...
FIFO Client1...
FIFO Client1...
FIFO Client1...
FIFO Client1...
client.c: Невозможно открыть FI
FO (No such file or directory)
al@al-VirtualBox:~/lab15/llab$
```

Вывод

Приобрел 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). Поскольку файл находится на локальной файловой системе, данное IPC используется внутри одной системы
2. Для создания неименованного канала используется системный вызов `pipe`. Массив из двух целых чисел является выходным параметром этого системного вызова.
3. Вызов функции `mkfifo()` создаёт файл канала
4. `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 или FIFO, возвращается требуемое число байтов, остаток сохраняется для последующих чтений.
7. При чтении большего числа байтов, чем находится в канале или FIFO, возвращается доступное число байтов. Процесс, читающий из канала, должен соответствующим образом обработать ситуацию, когда прочитано меньше, чем заказано.
8. да
9. Функция `write()` переписывает `count` байт из буфера, на который указывает `buf` в файл, соответствующий дескриптору файла `handle`. Указателю положения в файле дается приращение на количество записанных байт.
10. Строковая функция `strerror` - функция языков C/C++, транслирующая код ошибки, который обычно хранится в глобальной переменной `errno`, в сообщение об ошибке, понятном человеку.