

Отчет по лабораторной работе №15

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

Российский Университет Дружбы Народов

Факультет Физико-Математических и Естественных Наук

Дисциплина: *Операционные системы*

Студент: YASSINE OULED SALEM

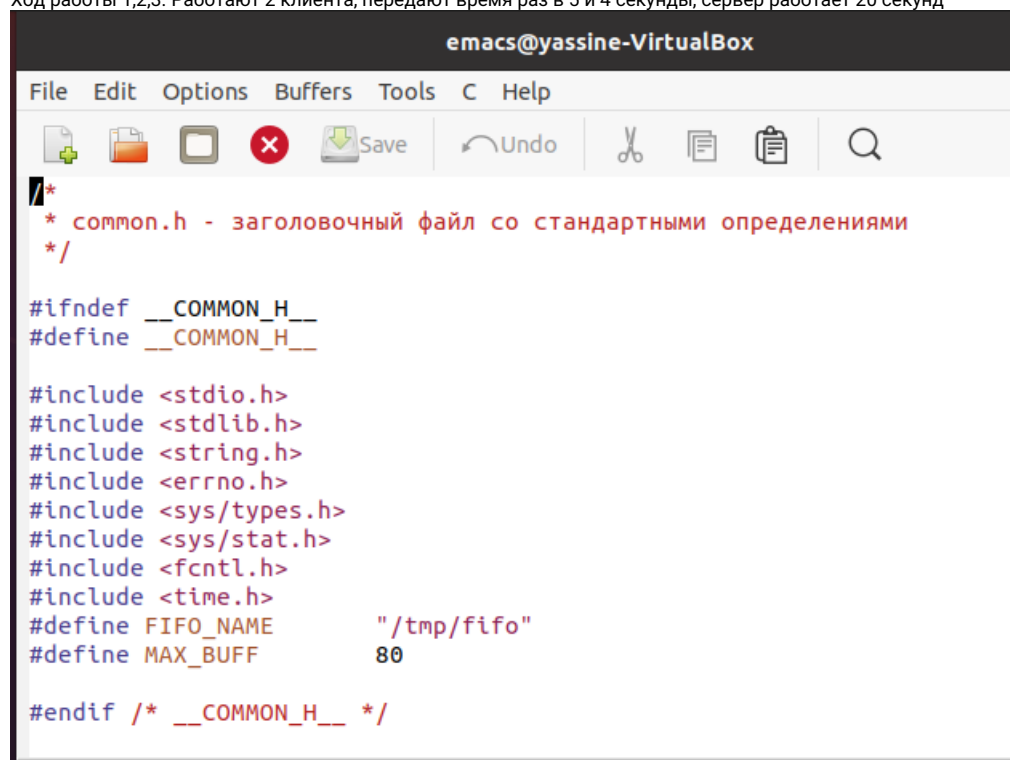
Группа: НПИбд-02-20

2021г.

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



```
emacs@yassine-VirtualBox
File Edit Options Buffers Tools C Help
[Icons: New, Open, Save, Undo, Redo, Find, etc.]
/*
 * 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__ */
```

```

#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));
        U:--- client.c      16% L16      (C/*l Abbrev)

        /* 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)
            {
                fprintf(stderr,"%s: Ошибка записи в FIFO (%s)\n",
                    __FILE__, strerror(errno));
                exit(-2);
            }
        }
        U:--- client.c      43% L25      (C/*l Abbrev)
    }

```

```

#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);
    }
    int start = time(NULL);
    U:--- server.c      15% L20      (C/*l Abbrev)

```

```
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);
}
}
close(readfd);/* закроем FIFO */

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

U:--- server.c Bot L53 (C/*l Abbrev)

```
File Edit Options Buffers Tools Makefile Help
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
```

```
yassine@yassine-VirtualBox:~$ ./server
FIFO Server...
Sat Jun 12 14:38:07 2021
Sat Jun 12 14:38:14 2021
Sat Jun 12 14:38:15 2021
Sat Jun 12 14:38:21 2021
Sat Jun 12 14:38:22 2021
Sat Jun 12 14:38:28 2021
Sat Jun 12 14:38:29 2021
Sat Jun 12 14:38:35 2021
Sat Jun 12 14:38:36 2021
Sat Jun 12 14:38:42 2021
Sat Jun 12 14:38:43 2021
```

Вывод

В результате работы, я 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).
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 имяфайла`, однако команды `mknod` нет в списке команд *X/Open*, поэтому она включена не во все UNIX-подобные системы. Предпочтительнее применять в командной строке - `$ mkfifo имяфайла`.
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