

Российский Университет Дружбы Народов

Факультет Физико-Математических и Естественных Наук

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №15

дисциплина: **Операционные системы**

Студент: ГАБРИЭЛЬ ТЬЕРРИ Группа: НКНбд 01-20

МОСКВА 2021 г.


Цель работы:

приобретение практических навыков работы с именованными каналами.

Ход работы: 1. Изучил приведённые в тексте программы server.c и client.c и взял данные примеры за образец. (Рисунки 1,2,3,4,5,6,7)

```
tgabriel@tagbriel:~  
File Edit View Search Terminal Help  
[tgabriel@tagbriel ~]$ touch common.h  
[tgabriel@tagbriel ~]$ touch server.c  
[tgabriel@tagbriel ~]$ touch client.c  
[tgabriel@tagbriel ~]$ touch Makefile  
[tgabriel@tagbriel ~]$ emacs
```

```
common.h - emacs@tagbriel.localdomain  
File Edit Options Buffers Tools C Help  
[Icons: File, Folder, Print, Close, Save, Undo, Cut, Copy, Paste, Find]  
#endif /* __COMMON_H__ */#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>  
#define FIFO_NAME "/tmp/fifo"  
#define MAX_BUFF 80  
#endif /* __COMMON_H__ */
```



```
#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);
}
/* 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);
}
}
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 L44      (C/l Abbrev)

```

client.c - emacs@tagbriel.localdomain

File Edit Options Buffers Tools C Help

```

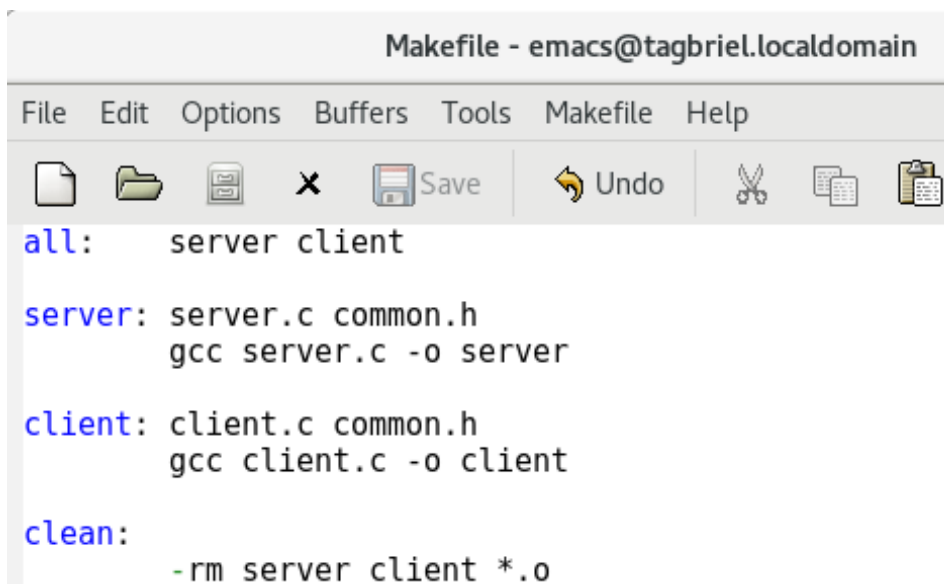
#include "common.h"
#define MESSAGE "Hello Server!!!\n"
int
main()
{
int writefd; /* дескриптор для записи в FIFO */
int msglen;
/* баннер */
printf("FIFO Client...\n");
/* получим доступ к FIFO */
if((writefd = open(FIFO_NAME, O_WRONLY)) < 0)
{
fprintf(stderr, "%s: Невозможно открыть FIFO (%s)\n",
FILE__, 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);
}
/* закроем доступ к FIFO */
close(writefd);
exit(0);
}

```



```

Makefile - emacs@tagbriel.localdomain
File Edit Options Buffers Tools Makefile Help
[Icons: File, Folder, Disk, X, Save, Undo, Scissors, Copy, Paste]
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

```

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). Использовал функцию `clock()` для определения времени работы сервера. (Рисунки,8,9,10,11,12,13,14,15,16,17,18,19)

```
common.h - emacs@tagbriel.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 <time.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 /* __COMMON_H__ */
```



```
#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);
    }
    clock_t now=time(NULL), start=time(NULL);
    while(now-start<30)
    {
        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);
            }
        }
        now=time(NULL);
    }
}
```










```

    printf("\n-----\nserver timeout\n%u seconds passed!\n-----\n", now -
start);
    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);
}

```

client.c - emacs@tagbriel.localdomain

File Edit Options Buffers Tools C Help





 Save
  Undo
 



```

#include "common.h"
#define MESSAGE "Hello Server!!!\n"
int
main()
{
    int writefd; /* дескриптор для записи в FIFO */
    int msglen;
    char message[10];
    int count;
    time_t t;
    time(&t);
    for (count=0; count<=5; ++count){
        sleep(5);
        message[9] = '\n';
        /* баннер */
        printf("%s", ctime(&t));
        /* 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);
}
}
/* закроем доступ к FIFO */
close(writefd);
exit(0);
}

```

Makefile - emacs@tagbriel.localdomain

File Edit Options Buffers Tools Makefile Help





 Save
 Undo




```

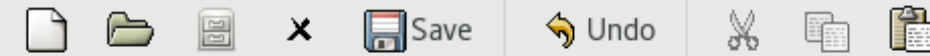
all:    server client

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

```

```
#include "common.h"
#define MESSAGE "Hello Server!!!\n"
int
main()
{
    int writefd; /* дескриптор для записи в FIFO */
    int msglen;
    char message[10];
    int count;
    time_t t;
    time(&t);
    for (count=0; count<=5; ++count){
        sleep(5);
        message[9] = '\n';
        /* баннер */
        printf("%s", ctime(&t));
        /* 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);
        }
    }
    /* закроем доступ к FIFO */
    close(writefd);
    exit(0);
}
```

```
[tgabriel@tagbriel ~]$ make
gcc server.c -o server
gcc client.c -o client
[tgabriel@tagbriel ~]$ touch client2.c
[tgabriel@tagbriel ~]$ emacs
[tgabriel@tagbriel ~]$ gcc client2.c -o client2
[tgabriel@tagbriel ~]$
```

tgabriel@tagbriel:~	tgabriel@tagbriel:~
File Edit View Search Terminal Help	File Edit View Search Terminal Help
[tgabriel@tagbriel ~]\$ touch client.c	[tgabriel@tagbriel ~]\$./client
[tgabriel@tagbriel ~]\$ touch Makefile	Sat Jun 12 14:34:39 2021
[tgabriel@tagbriel ~]\$ emacs	Sat Jun 12 14:34:39 2021
[tgabriel@tagbriel ~]\$ make	Sat Jun 12 14:34:39 2021
gcc server.c -o server	Sat Jun 12 14:34:39 2021
gcc client.c -o client	Sat Jun 12 14:34:39 2021
[tgabriel@tagbriel ~]\$ touch client2.c	Sat Jun 12 14:34:39 2021
[tgabriel@tagbriel ~]\$ emacs	[tgabriel@tagbriel ~]\$
[tgabriel@tagbriel ~]\$ gcc client2.c -o client2	
[tgabriel@tagbriel ~]\$./server	
FIFO Server...	
Hello Server!!!	
Hello Server!!!	
Hello Server!!!	
Hello Server!!!	
Hello Server!!!	
Hello Server!!!	

server timeout	
30 seconds passed!	

[tgabriel@tagbriel ~]\$	

[illegible]

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

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, 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 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.