

Отчёт о выполнении лабораторной работы №15

Именованные каналы

Факультет Физико-Математических и Естественных Наук

Дисциплина: *Операционные системы*

Работу выполняла: *Живцова Анна*

1032201673

НКН60-01-20

Москва. Дисплейный класс РУДН. 2021г.

Цель работы

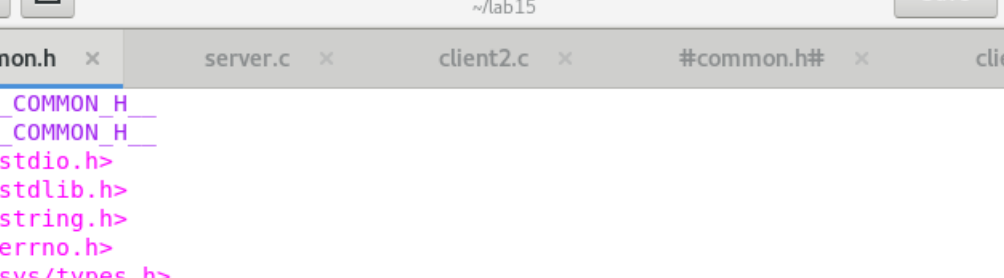
Приобретение практических навыков работы с именованными каналами.

Выполнение работы

Изучила приведённые в тексте программы `server.c` и `client.c`. Взяв данные примеры за образец, написала аналогичные программы, внося следующие изменения:

1. Работает не 1 клиент, а 2.
2. Клиенты передают текущее время с некоторой периодичностью.

- The image shows two terminal windows side-by-side. The left window is titled 'aazhivcova@aazhivcova:~/lab...' and shows a server process running. It prints 'FIFO Server...' and then a series of timestamps from 'Mon May 17 08:55:47 2021' to 'Mon May 17 08:56:17 2021'. The right window is titled 'aazhivcova@aazhivcova:~/lab...' and shows a client process running. It prints '^C' and then a series of timestamps from 'Mon May 17 08:55:47 2021' to 'Mon May 17 08:56:17 2021'. Both windows show file paths like '/server' and '/client1'. The right window also shows a file path '/client2'. The left window shows a 'server timeout' message and a '30 seconds passed' message. The right window shows a 'client1.c: Невозможно открыть FI' message and a 'FIFO (No such file or directory)' message. The left window shows a 'rtkit-daemon' message. The right window shows a 'private-...' message. The left window shows a 'b394122b7ae' message. The right window shows a 'c40a40860ac' message. The left window shows a 'rtkit-daemon' message. The right window shows a 'private-...' message. The left window shows a 'b394122b7ae' message. The right window shows a 'c40a40860ac' message.



```
Activities Text Editor May 17 08:58
common.h
~/lab15
Save
common.h x server.c x client2.c x #common.h# x client1.c x

#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>
#include<sys/un.h>
#include<sys/socket.h>
#include<unistd.h>

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

#endif

C/ObjC Header Tab Width: 8 Ln 8, Col 3 INS
```

client1

```

#include "common.h"

int main()
{
    for (int i=0; i<10;i++){
        int writefd;
        int msglen;
        long int ttime;
        ttime=time(NULL);
#define MESSAGE ctime(&ttime)
        printf("FIFO Client1...\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);
        }
        sleep(5);
        close(writefd);
    }
}

```

client2

```

for (int i=0; i<10; i++){
    int writefd;
    int msglen;

    long int ttime;
    ttime=time(NULL);
#define MESSAGE ctime(&ttime)
    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);
    }
    sleep(2);
    close(writefd);
}
exit(0);
}

```

server

```

#include "common.h"
int
main()
{
    int readfd;
    int n;
    char buff[MAX_BUFF];
    printf("FIFO Server...\n");
    if(mknod(FIFO_NAME, S_IFIFO|0666,0)<0)
    {
        fprintf(stderr,"%s: Невозможно создать FIFO (%s)\n",__FILE__,
strerror(errno));
        exit(-1);
    }
    if((readfd=open(FIFO_NAME, O_RDONLY))<0)
    {
        fprintf(stderr,"%s: Невозможно открыть FIFO (%s)\n",__FILE__,
strerror(errno));
        exit(-2);
    }
    clock_t currentTime=time(NULL), begin=time(NULL);
    while(currentTime-begin< 30)
    {
        if((n=read(readfd, buff, MAX_BUFF))>0)
        {
            if(write(1, buff, n)!=n)

```

server

```

            if(write(1, buff, n)!=n)
            {
                fprintf(stderr,"%s: Ошибка вывода (%s)
\n",__FILE__,strerror(errno));
                exit(-3);
            }
        }
        currentTime=time(NULL);
    }
    printf("\nserver timeout\n%u seconds passed", (currentTime-begin));
    close(readfd);
    if(unlink(FIFO_NAME)<0)
    {
        fprintf(stderr,"%s: Невозможно удалить FIFO (%s)\n",__FILE__, strerror(errno));
        exit(-4);
    }
    exit(0);
}

```

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. `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 `bufy` 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.

Библиография

<https://habr.com/ru/post/122108/>

https://www.opennet.ru/docs/RUS/linux_parallel/node17.html

http://rus-linux.net/MyLDP/consol/An_introduction_to_pipes_in_Linux.html

Вывод

Приобрела практические навыки работы с именованными каналами.