

РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

Факультет физико-математических и естественных наук

Кафедра прикладной информатики и теории вероятностей

ОТЧЕТ

ПО ЛАБОРАТОРНОЙ РАБОТЕ № 15

дисциплина: операционные системы

Студент: Пиняева Анна Андреевна

Группа: НФИбд-02-20

МОСКВА

2021

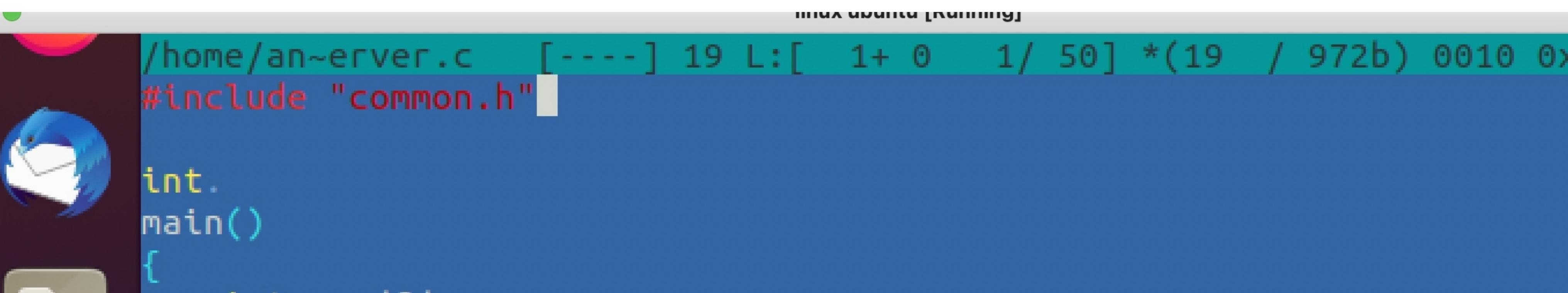
Цель работы

Приобретение практических навыков работы с именованными каналами.

Ход работы

1. Ознакомилась с программами и на их основе написала свои, добавила 1 клиент.

Рисунок 1 (server.c):



The screenshot shows a terminal window with the following content:

```
/home/an~erver.c [---] 19 L:[ 1+ 0 1/ 50 ] *(19 / 972b) 0010 0x  
#include "common.h"  
  
int.  
main()  
{
```

The terminal window has a dark blue background and light blue text. The title bar of the window says "Изучение языка программирования". On the left side of the terminal, there is a vertical docked panel with icons, including one that looks like a mail icon.

```
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 now=time(NULL), start=time(NULL);
while(now-start<30)
{
    while ((n=read(readfd,buff,MAX_BUFF))>0)
    {
```

Рисунок 2 (server.c):

The screenshot shows a terminal window with the following details:

- Terminal title: annapinyaeva@annapinyaeva-VirtualBox: ~/work/2020-2021...
- Code editor interface with tabs and search bar.
- Code content (server.c):

```
#!/home/an~erver.c [-M--] 61 L:[ 24+12 36/ 50] *(802 / 997b) 0044 0x02C [*]
{
    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);
if(unlink(FIFO_NAME)<0)
{ <-><----->
```

```
<----->fprintf(stderr,"%s:Невозможно удалить FIFO (%s)\n",
<----->      __FILE__,strerror(errno));
<----->exit(-4);
}
exit(0);
}
```

Рисунок 3 (client.c):

```
/home/an~lient.c [---] 1 L:[ 1+23 24/ 24] *(453 / 453b) <EOF> [*]
#include "common.h"
#define MESSAGE "Hello Server!!!\n"

int main()
{
int writefd;
int msglen;
printf("FIFO client...\n");
if((writefd=open(FIFO_NAME,O_WRONLY))<0)
{
    perror("Error opening FIFO");
    exit(1);
}
msglen = strlen(MESSAGE);
write(writefd,MESSAGE,msglen);
close(writefd);
}
```

```
fprintf(stderr,"%s:It is impossible to open FIFO (%s)\n",
        __FILE__,strerror(errno));
exit(-1);
}

msglen=strlen(MESSAGE);
if(write(writefd,MESSAGE,msglen)!=msglen)
{
fprintf(stderr,"%s: Error writing in FIFO (%s)\n",
<----->__FILE__,strerror(errno));
exit(-2);
}
close (writefd);
exit(0);
}
```

Рисунок 4 (client2.c):

```
linux ubuntu [Running]
/home/an-ient2.c  [---]  0 L:[ 1+ 0    1/ 29] *(0    / 620b) 0035 0x023 [*]
#include "common.h"
#define MESSAGE "Hello Server!!!\n"
```

```
int
main()
{
int writefd;
int msglen;
int count;
char message[10];
long long int;
for (count=0; count<=5; ++count) {
sleep(5);
T=(long long int) time(0);
sprintf(message,"%lli",T);
message[9]='\n';
printf("FIFO Client...\n");
if ((writefd=open(FIFO_NAME,O_WRONLY))<0)
{ fprintf(stderr, "%s: It is impossible to open FIFO (%s)\n",
__FILE__,strerror(errno));
exit(-1);}
msglen=strlen(MESSAGE);
if(write(writefd,MESSAGE,msglen)!=msglen)
{ fprintf(stderr,"%s: Error writing in FIFO (%s)\n",
__FILE__,strerror(errno));}
close(writefd);
}
```

```
<----->__FILE__, strerror(errno));  
    exit(-2);}  
close(writefd);
```

Рисунок 5 (common.h):

```
/home/an~ommon.h [----] 21 L:[ 1+12 13/ 15] *(240 / 265b) 0010 0x00A [*]  
#ifndef __COMMON_H__  
#define __COMMON_H__  
  
#include <stdio.h>  
#include <stdlib.h>  
#include <string.h>  
#include <errno.h>  
#include <sys/types.h>  
#include <sys/start.h>  
#include <fcntl.h>  
  
#define FIFO_NAME    "/tmp/fifo"  
#define MAX_BUFF    80  
  
#endif /* __COMMON_H__ */
```

Рисунок 6 (client2.c):

linux ubuntu [Running]

```
int
main()
{
int writefd;
int msglen;
int count;
char message[10];
long long int;
for (count=0; count<5; ++count) {
```

```
for (count=0; count<=5; ++count) {  
    sleep(5);  
    T=(long long int) time(0);  
    sprintf(message,"%lli",T);  
    message[9]='\n';  
    printf("FIFO Client...\\n");  
    if ((writefd=open(FIFO_NAME,O_WRONLY))<0)  
    { fprintf(stderr, "%s: It is impossible to open FIFO (%s)\\n",  
             __FILE__,strerror(errno));  
        exit(-1);}  
    msglen=strlen(MESSAGE);  
    if(write(writefd,MESSAGE,msglen)!=msglen)  
    { fprintf(stderr, "%s: Error writing in FIFO (%s)\\n",  
             <----->__FILE__,strerror(errno));  
        exit(-2);}  
    close(writefd);  
    exit(0);  
}
```

2. Получили следующий результат:

```
annapinyaeva@annapinyaeva-VirtualBox:~/work/2020-2021/os/lab15$ ./client.c
```

```
FIFO Client...
```

```
annapinyaeva@annapinyaeva-VirtualBox:~/work/2020-2021/os/lab15$ ./client2.c
```

```
FIFO Client...
```



Рисунок 8:

```
annapinyaeva@annapinyaeva-VirtualBox:~/work/2020-2021/os/lab15$ ./server.c
FIFO Server...
Hello Server!!!
-----
server timeout
30 seconds passed!
-----
annapinyaeva@annapinyaeva-VirtualBox:~/work/2020-2021/os/lab15$ ./server.c
FIFO Server...
Hello Server!!!
-----
server timeout
30 seconds passed!
-----
```

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: приобретены 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 односторонняя 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.