Титульный лист

РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

Факультет физико-математических и естественных наук

Лабораторная работа 15

По дисциплине "Операционные системы"

Выполнил:

Студент группы НПВбм-01-19

Студенческий билет №: 1032193844

Саидов Ахият Магомадович

Руководитель: Валиева Татьяна Рефатовна

Москва 2023

Цель работы

Приобретение практических навыков работы с именованными каналами.

Начало работы

Я изучил приведенные в тексте программы server.c и client.c. Взяв данные примеры за образец, напишем аналогичные программы, внеся следующие изменения.

- Для этого создадим *common.h, server.c, client.c, Makefile*.

```
[AMSaidov@amsaidov ~]$ mkdir lab15
[AMSaidov@amsaidov ~]$ cd lab15
[AMSaidov@amsaidov lab15]$ touch common.h
[AMSaidov@amsaidov lab15]$ touch server.c
[AMSaidov@amsaidov lab15]$ touch client.c
[AMSaidov@amsaidov lab15]$ touch Makefile
```

Рисунок 1

- Запишем в файл common.h следующий код:

```
/*
* 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>
#define FIFO_NAME "/tmp/fifo"
#define MAX_BUFF 80
#endif /* __COMMON_H__ */
```

Рисунок 2

- Запишем в файл server.c следующий код:

```c
#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 входа (%s)\n",
__FILE__, strerror(errno));
exit(-3);
}
}
now=time(NULL);
}
printf("\n-----\nserver timeout\n%u seconds passed!\n-----\n",now-start);
close(readfd);
```

Рисунок 3

- Запишем в файл client.c следующий код:

```c
#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);
}
```

Рисунок 4

- Запишем в файл Makefile следующий код:

```
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
```

Рисунок 5

- Проверим на работоспособность.

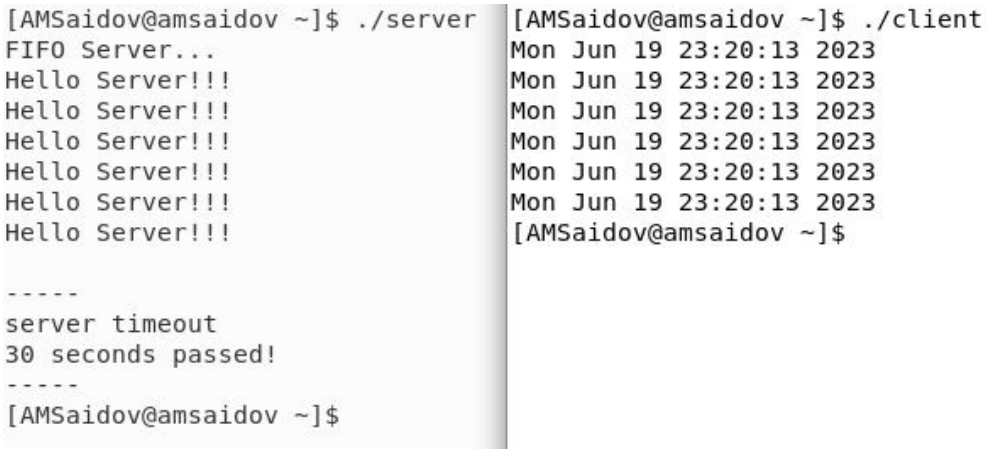

Рисунок 6

Вывод

Мы приобрели практические навыки работы с именованными каналами.

Контрольные вопросы

1. У именованных каналов есть идентификатора канала, а у неименованных его нет
2. Возможно создание неименованного канала из командной строки, но только с созданием временного канала с индикатором.
3. Да. При помощи mknod.
4. #include int fd[2];

   pipe(fd);

   /* возвращает 0 в случае успешного завершения, -1 - в случае ошибки;*/ Это значит, что функция возвращает два файловых дескриптора: fd[0] и fd[l]
5. #include <sys/types.h>

   #include <sys/stat.h>

   int mkfifo(const char *pathname, mode_t mode);
6. При чтении меньшего числа байтов, чем находится в канале или FIFO, возвращается требуемое число байтов, остаток сохраняется для последующих чтений. 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 требуется в программе
7. Запись числа байтов, меньшего числа битов у канала или FIFO, в случае, когда несколько процессов одновременно записывают в канал, порции данных от этих процессов не перемешиваются. При записи большего числа байтов, чем это позволяет канал или FIFO, вызов write(2) блокируется до освобождения занятой нами до этого памяти.
8. Могут ли два и более процессов читать или записывать в канал? Ответ: Да. Если у buff достаточное количество памяти.
9. Функция записывает length памяти из буфера buffer в файл, определенный дескриптором файла fd. Эта операция чисто ‘двоичная’ и без буферизации. При единице возвращает действительное число байтов. При -1 сообщение об ошибке.

10. Интерпретирует номер ошибки, передаваемый в функцию в качестве аргумента — errornum, в понятное для человека текстовое сообщение (строку). Откуда берутся эти ошибки? Ошибки эти возникают при вызове функций стандартных Си-библиотек. То есть хорошим тоном программирования будет — использование этой функции в паре с другой, и если возникнет ошибка, то пользователь или программист поймет, как исправить ошибку, прочитав сообщение функции strerror. 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.