Российский университет дружбы народов

Факультет физико-математических и естественных наук

Отчёт по лабораторной работе №15

тема "Именованные файлы"

Подготовила: Голощапова Ирина Борисовна

Студентка группы НФИбд-01-20

Цель работы

Приобретение практических навыков работы с именованными каналами

Библиография

Именованные файлы в Linux

goto-linux

FIFO именованные файлы

Выполнение лабораторной работы

1. Изучила приведённые в тексте программы server.c и client.c. Взяв данные примеры за образец.

Рис.1 "Создание и радактирование файлов"

Файл Client.c:

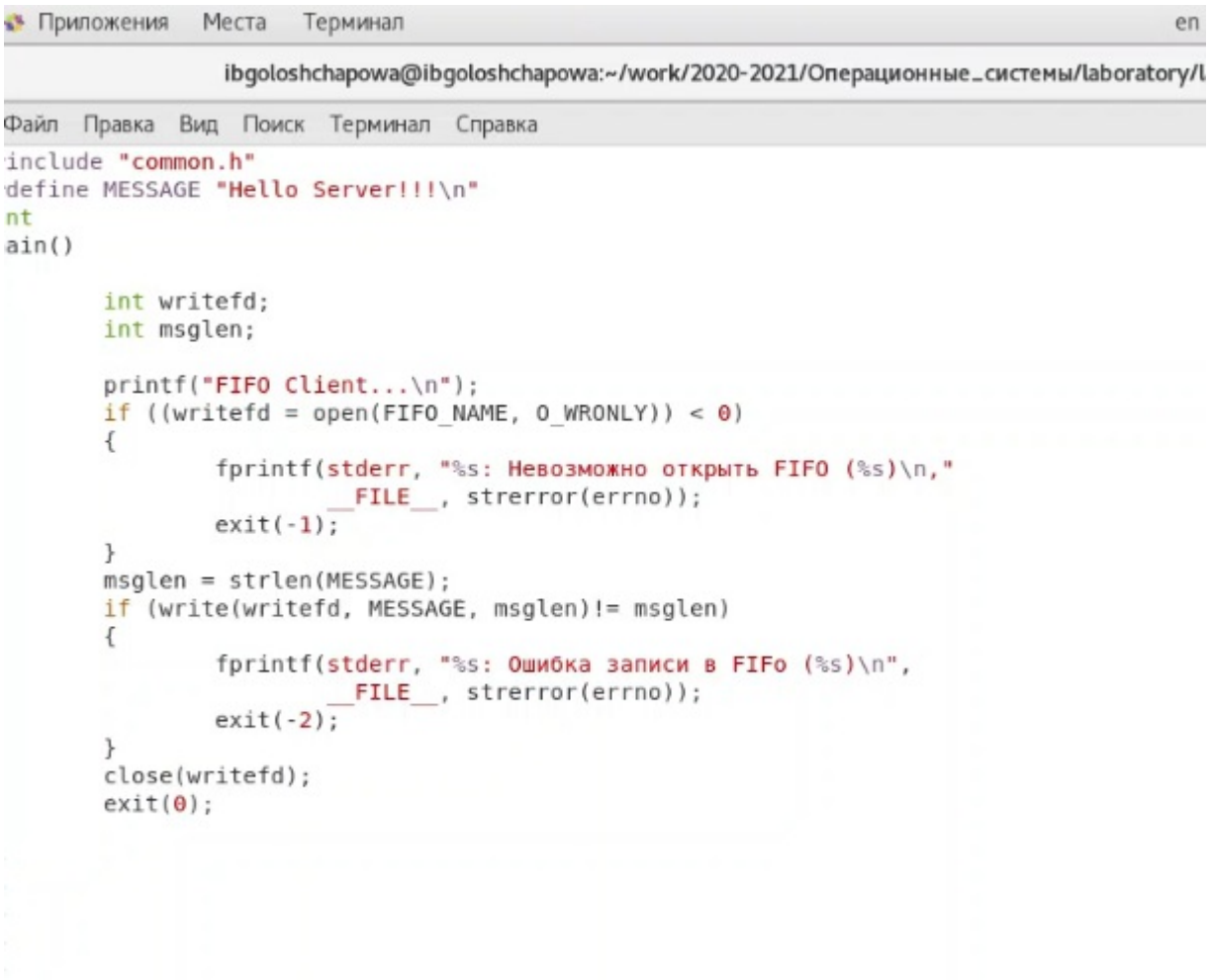

Рис.2 "*Client.c*"

Файл Server.c:

ibgoloshchapowa@ibgoloshchapowa:~/work/2020-2021/Операционные_системы/labora

Файл Правка Вид Поиск Терминал Справка

```
#include "common.h"
int
main()
{
        int readfd;
        int n;
        char buff[MAX_BUFF];
        printf("FIFO Server...\n");
        if (mknod(FIFO_NAME, S_FIFO | 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__, srterror(errno));
                exit(-2);
        }
        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);
        if(unlink (FIFO_NAME) < 0)
        {
                fprintf(stderr, "%s: Невозможно удалить FIFO (%s)\n",
                        __FILE__, strerror(errno));
                exit(0);
        }
}
~
~
~
~
~
"server.c" [Новый] 37L, 803C записано
```

Рис.3 "Server.c"

Файл common.h:

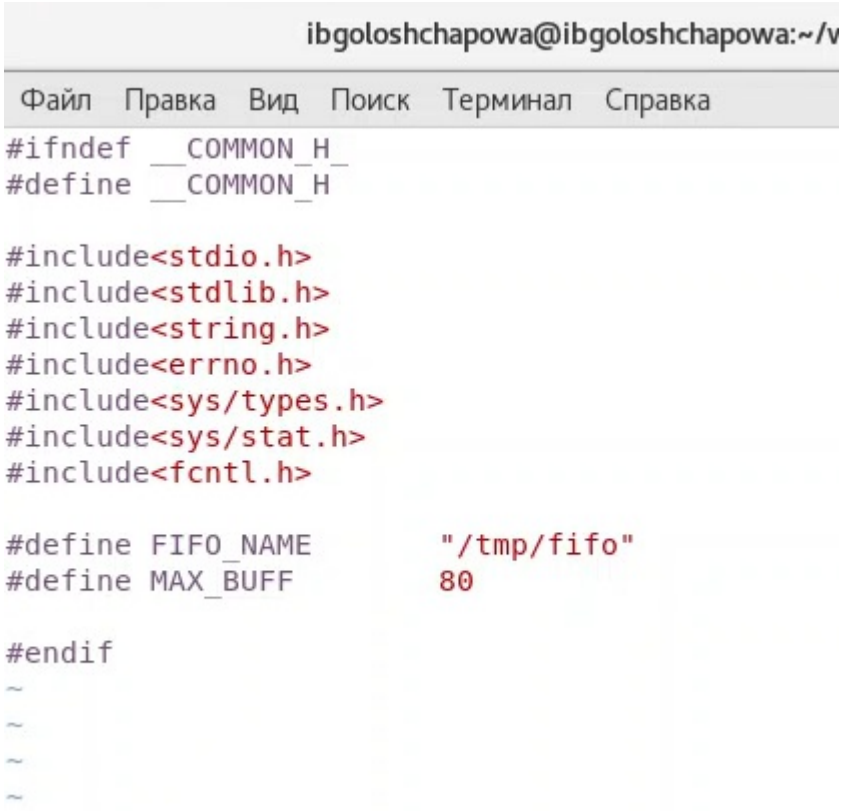

Рис.4 "common.h"

Файл MakeFile:

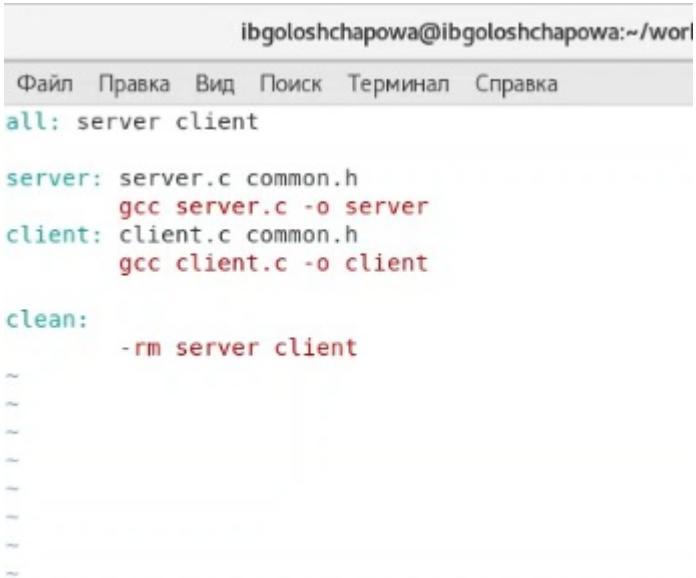

Рис.5 "MakeFile"

Написала аналогичные программы, внеся следующие изменения:

1. Работает не 1 клиент, а несколько (например, два).
2. Клиенты передают текущее время с некоторой периодичностью (например, раз в пять секунд). Используйте функцию sleep() для приостановки работы клиента.
3. Сервер работает не бесконечно, а прекращает работу через некоторое время (например, 30 сек). Используйте функцию clock() для определения времени работы сервера. Что будет в случае, если сервер завершит работу, не закрыв канал

Листинг новой программы:

Файл Client.c:

```
#define MESSAGE "Hello Server!!!\n"
int main()
{
        int writefd;
        int msglen;
        char message[10];
        long long int T;
        int 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: FIFO can't be openned(%s)\n", __FILE__,
                exit(-1);
                }
                msglen = strlen(MESSAGE);
                if (write(writefd, MESSAGE, msglen)!= msglen)
                {
                        fprintf(stderr,"%s: FIFO can't be written(%s)\n", __F
                        exit(-2);
                }
        }
        close(writefd);
        exit(0);
}
```

Рис.6 "Client.c"

Файл Server.c:

ibgoloshch

Файл Правка Вид Поиск Терминал Справка

```
#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))
        {
                fprintf(stderr, "%s:FIFO cannot be created(%s\n)",__FILE__,st
                exit(-1);
        }
        if ((readfd=open(FIFO_NAME, O_RDONLY)) < 0)
        {
                fprintf(stderr, "%s: FIFO cannot be opened(%s)\n", __FILE__,s
                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: DATA cannot be written(%s
                                exit(-3);
                        }
                }
                now=time(NULL);
        }
        printf("\n-----\nserver timeout/n%u seconds passed!\n-----\n",now-sta
        close(readfd);
        if (unlink(FIFO_NAME)<0)
        {
                fprintf(stderr, "%s:FIFO cannot be delete(%s)\n", __FILE__,st
                exit(-4);
        }
        exit(0);
}
~
```

Рис.7 "Server.c"

Ввод команды для запуска программ (для запуска необходимо открыть два терминала):

```
[ibgoloshchapowa@ibgoloshchapowa lab15]$ ./client
FIFO Client...
```

Рис.1 "запуск программ"

```
[ibgoloshchapowa@ibgoloshchapowa lab15]$ ./server
FIFO Server...
server.c: Невозможно создать FIFO (File exists)
[ibgoloshchapowa@ibgoloshchapowa lab15]$
```

Рис.1 "запуск программ2"

Выводы

В ходе лабораторной работы я приобрела 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.