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: Абушек Дмитрий Олегович

Группа: НФИбд-01-20

МОСКВА

2021 г.

Цель работы: приобретение практических навыков работы с именованными каналами.

Ход работы:

1.Изучили приведённые в тексте программы server.c и client.c и взяли данные примеры за образец.

2.Написали аналогичные программы, внеся следующие изменения:

- работает не 1 клиент, а несколько (например, два).
- клиенты передают текущее время с некоторой периодичностью (например, раз в пять секунд). Использовали функцию sleep() для приостановки работы клиента.
- сервер работает не бесконечно, а прекращает работу через некоторое время (например, 30 сек). Использовали функцию clock() для определения времени работы сервера.

```
[dmitryabushek@MacBook-Pro-Dmitrij ~ % mkdir lab15                              ]
[dmitryabushek@MacBook-Pro-Dmitrij ~ % cd lab15                                 ]
[dmitryabushek@MacBook-Pro-Dmitrij lab15 % touch common.h                       ]
[dmitryabushek@MacBook-Pro-Dmitrij lab15 % vi common.h                          ]
 dmitryabushek@MacBook-Pro-Dmitrij lab15 %
```

common.h:

```
#ifndef __COMMON_H__
#define __COMMON_H__

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>

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

#endif

~
```

server.c:

```
#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 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("server timeout, %li - seconds passed\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);
}
```

client.c:

```
#include "common.h"
#define MESSAGE "Hello Server!!!\n"

int
main()
{
        int msg, len, i;
        long int t;

        for (i=0; i<20; i++)
        {

                sleep(3);
                t=time(NULL);
                printf("FIFO Client...\n");

                if((msg = open(FIFO_NAME, O_WRONLY)) < 0)
                {
                        fprintf(stderr, "%s: Невозможно открыть FIFO (%s)\n",
                                __FILE__, strerror(errno));
                        exit(-1);
                }

                len = strlen(MESSAGE);

                if(write(msg, MESSAGE, len) != len)
                {
                        fprintf(stderr, "%s: Ошибка записи в FIFO (%s)\n",
                                __FILE__, strerror(errno));
                        exit(-2);
                }

                close(msg);
}
        exit(0);
}
```

client2.c:

```
#include "common.h"
#define MESSAGE "Hello Server!!!\n"

int
main()
{
        int writefd;
        int msglen;
        int count;
        long long int t;
        char message[10];

        for (count=0; count<=5; ++count)
        {
        sleep(5);
        t=(long long int) time(0);
        sprintf(message, "%lli", t);
        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);
        }
}
        close(writefd);
        exit(0);
}
~
```

Makefile:

```
dmitryabushek@MacBook-Pro-Dmitrij lab15 % touch makefile
dmitryabushek@MacBook-Pro-Dmitrij lab15 % vi makefile
```

Результат:

```
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
```

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: Я приобрел практические навыки работы с именованными каналами.

Ответы на 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.