МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГООБРАЗОВАНИЯ«РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ»

Факультет физико-математических и естественных наук Кафедра информационных технологий

ОТЧЕТ

по лабораторной работе 14

ТЕМА «Именованные каналы**»** по дисциплине **«** Операционные системы**»**

Выполнил/ла

Студент/ка группы НПИбд-02-21

Студенческий билет No 1032205421

Студент/ка: Стелина Петрити

Цель работы

Приобретение практических навыков работы с именованными каналами.

Последовательность выполнения работы

Изучите приведённые в тексте программы server.c и client.c. Взяв данные примеры за образец, напишите 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

4.коды:

```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 start = time(NULL);

 while(time(NULL)-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);
         }
     }
 }
 close(readfd);
 if(unlink(FIFO_NAME) <0)
   {
     fprintf(stderr, "%s: Can not delete FIFO(%s)\n",
             __FILE__, strerror(errno));
     exit(-4);
   }
 exit(0);
}
```

```
U:---  server.c        All L9      (C/*l
```

4.1.server.c

```c
#include "common.h"

#define MESSAGE "Hello Server!!!\n"

int
main()
{
  int writefd; /* дескриптор для записи в FIFO */
  int msglen;

  printf("FIFO Client...\n");

  for(int i=0; i<4; i++)
    {
      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);
    }

 /* закроем доступ к FIFO */
  close(writefd);

  exit(0);
}
```

```
U:---  client.c        All L1      (C/*l -2
```

4.2.client.c

```c
#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 <unistd.h>
#include <time.h>

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

#endif /* __COMMON_H__ */
```

4.3.common.h

```
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
```

```
-:---  Makefile       All L1     (GNUmakefile)
```

4.4.Makefile

- В файл common.h unistd.h и time.h необходимые для работы кодов других файлов
- В файл server.c цикл while для контроля за временем работы сервера.
- В файле client.c цикл для количества сообщений и команду sleep, чтобы приостановить работу клиента на 5 секунд.

2. Исполнение файлов

1.команда make all, и скомпилировал необходимые файлы

```
[inna@fedora lab14]$ make all
gcc server.c -o server
gcc client.c -o client
```

1.1 make all command

*2.выполнения файлов**client.c; server.c*

```
[inna@fedora lab14]$ ./client
FIFO Server...
```

2.1.client.c

```
[inna@fedora lab14]$ ./server
FIFO Server...
Hello Server!!!
Hello Server!!!
Hello Server!!!
Hello Server!!!
```

2.2. server.c

Выводы

Во время этой лабораторной работы я приобрела практические навыки работы с именованными каналами

Контрольные вопросы

Контрольные вопросы

1.В чем ключевое отличие именованных каналов от неименованных?

Могут использоваться неродственными процессами. Они дают вам, по сути, те же возможности, что и неименованные каналы, но с некоторыми преимуществами, присущими обычным файлам. Именованные каналы используют специальную запись в директории для управления правами доступа.

2.Возможно ли создание неименованного канала из командной строки?

Существует возможность. Именованные каналы создают соединения между двумя процессами и могут использоваться как обычные файлы.

3.Возможно ли создание именованного канала из командной строки? Да, возможно

4.Опишите функцию языка C, создающую неименованный канал.

Неименованный канал - средство взаимодействия между связанными процессами. Родительский процесс создает канал при помощи системного вызова: int pipe(int fd[2]);

Массив из двух целых чисел является выходным параметром этого системного вызова. Если вызов выполнился нормально, то этот массив содержит два файловых дескриптора.

fd[0] -дескриптор для чтения из канала,

fd[1] - дескриптор для записи в канал.

Когда процесс порождает другой процесс, дескрипторы родительского процесса наследуются дочерним процессом, и, таким прокладывается трубопровод между двумя процессами.

5.Опишите функцию языка C, создающую именованный канал.

Файлы именованных каналов создаются функцией mkfifo()mknod

int mkfifo(const char *pathname, mode_t mode);, где первый параметр − путь, где будет располагаться FIFO, второй параметр определяет режим работы с FIFO

mknod (namefile, IFIFO | 0666, 0), где namefile − имя канала, 0666 − к каналу разрешен доступ на запись и на чтение любому запросившему процессу.

6.Что будет в случае прочтения из fifo меньшего числа байтов, чем находится в канале? Большего числа байтов?

меньшего числа байтов, чем находится в канале или FIFO, возвращается требуемое число байтов, остаток сохраняется для последующих чтений.

большего числа байтов, чем находится в канале или FIFO, возвращается доступное число байтов.

7.Аналогично, что будет в случае записи в fifo меньшего числа байтов, чем позволяет буфер? Большего числа байтов?

При записи большего числа байтов, чем это позволяет канал или FIFO, вызов *write(2)* блокируется до освобождения требуемого места. Если процесс пытается записать данные в канал, не открытый ни одним процессом на чтение, процессу генерируется сигнал , а вызов *write(2)* возвращает 0 с установкой ошибки.

Запись числа байтов, меньшего емкости канала или FIFO, гарантированно атомарно.

8.Могут ли два и более процессов читать или записывать в канал? Могут

9.Опишите функцию write (тип возвращаемого значения, аргументы и логику работы).

Что означает 1 (единица) в вызове этой функции в программе server.c (строка 42)?

Функция записывает length байтов из buffer. Эта операция двоичная без buffer. Реализуется как непосредственный вызов . Функция write мы посылаем сигнал клиенту или серверу.

10.Опишите функцию strerror

Функция strerror() возвращает указатель на строку, которая описывает код ошибки, (Например, если errnum равен EINVAL, возвращаемое описание будет "Недопустимый аргумент".) Эта строка не должен быть изменен приложением, но может быть изменен последующим вызовом strerror() или strerror_l().