Лабораторная работа № 15.

Кальсин З. А.

[1]RUDN University, Moscow, Russian Federation

Цель работы

Цель работы

приобретение практических навыков работы с именованными кана

Выполнение лабораторной работы

Выполнение лабораторной работы

Ход работы:

1.Изучил приведённые в тексте программы server.c и client.c и взял данные

примеры за образец.

common.h:

```
/** 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__ */
```

Слайд 1

server.c:

```c
/*
* server.c - реализация сервера
*
* чтобы запустить пример, необходимо:
*  1. запустить программу server на одной консоли;
*  2. запустить программу client на другой консоли.
*/
#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);
    }
  /* читаем данные из FIFO и выводим на экран */
  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);
        }
```

Слайд 2

```
          exit(-3);
        }
      }
  now=time(NULL);
    }
  printf("\n\nсервер будет остановлен\n%u секунд спустя\n\n", now-start);
  close(readfd);
  /* закроем FIFO */
  /* удалим FIFO из системы */
  if(unlink(FIFO_NAME) < 0)
    {
      fprintf(stderr, "%s: Невозможно удалить FIFO (%s) \n",
              __FILE__, strerror(errno));
      exit(-4);
    }
  exit(0);
}
```

Слайд 3

client.c:

```c
/*
* client.c - реализация клиента
*
* чтобы запустить пример, необходимо:
*  1. запустить программу server на одной консоли;
*  2. запустить программу client на другой консоли.
*/
#include "common.h"
#define MESSAGE "Hello Server!!!\n"
int
main()
{
  int writefd; /* дескриптор для записи в FIFO */
  int msglen; /* баннер */
  char message[10];
  int count;
  long long int T;
  for (count=0; count<=5; ++count){
      sleep(5);
      T = (long long int) time(0);
      message[9] = '\n';
  printf("FIFO Client...\n");
  /* 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);
}
```

Слайд 4

2.Написал 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). Использовала функцию sleep() для приостановки работы клиента.

- сервер работает не бесконечно, а прекращает работу через некоторое время (например, 30 сек). Использовала функцию clock() для определения времени работы

сервера.

Слайд 5

common.h:

```c
/** 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__ */
```

Слайд 6

server.c:

```c
/*
* server.c - реализация сервера
*
* чтобы запустить пример, необходимо:
*  1. запустить программу server на одной консоли;
*  2. запустить программу client на другой консоли.
*/
#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);
    }
  /* читаем данные из FIFO и выводим на экран */
  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);
        }
```

Слайд 7

```
          exit(-3);
        }
      }
  now=time(NULL);
    }
  printf("\n\nсервер будет остановлен\n%u секунд спустя\n\n", now-start);
  close(readfd);
  /* закроем FIFO */
  /* удалим FIFO из системы */
  if(unlink(FIFO_NAME) < 0)
    {
      fprintf(stderr, "%s: Невозможно удалить FIFO (%s) \n",
              __FILE__, strerror(errno));
      exit(-4);
    }
  exit(0);
}
```

Слайд 8

client.c:

Слайд 9

```c
/*
* client.c - реализация клиента
*
* чтобы запустить пример, необходимо:
*  1. запустить программу server на одной консоли;
*  2. запустить программу client на другой консоли.
*/
#include "common.h"
#define MESSAGE "Hello Server!!!\n"
int
main()
{
  int writefd; /* дескриптор для записи в FIFO */
  int msglen; /* баннер */
  char message[10];
  int count;
  long long int T;
  for (count=0; count<=5; ++count){
      sleep(5);
      T = (long long int) time(0);
      message[9] = '\n';
  printf("FIFO Client...\n");
  /* 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);
}
```

client2.c:

Слайд 10

Слайд 11

```
client.c:32:6: предупреждение: неявная декларация функции «write»; имелось в виду «fwrite»? [-Wimplicit-function-declaration]
   32 |   if(write(writefd, MESSAGE, msglen) !=msglen)
      |      ^~~~~
      |      fwrite
client.c:40:3: предупреждение: неявная декларация функции «close»; имелось в виду «pclose»? [-Wimplicit-function-declaration]
   40 |   close (writefd);
      |   ^~~~~
      |   pclose
zakaljsin@dk8n78 ~/lab15 $ ./server
FIFO Server...
Hello Server!!!
Hello Server!!!
Hello Server!!!
Hello Server!!!
Hello Server!!!
Hello Server!!!


сервер будет остановлен
30 секунд спустя

zakaljsin@dk8n78 ~/lab15 $ ./server
FIFO Server...
```

Слайд 12

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.

Слайд 13

Вывод: приобрел практические навыки работы с именованными каналами.

Вывод: приобрел практические навыки работы с именованными каналами.

Ответы на контрольные вопросы:

Ответы на контрольные вопросы:

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.

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.