Лабораторная работа № 14

Операционные системы

Шуваев Сергей Александрович

Содержание

Список иллюстраций

Список таблиц

1 Цель работы

Приобретение практических навыков работы с именованными каналами

2 Задание

Изучить приведённые в тексте программы server.c и client.c. Взяв данные примеры за образец, напишем 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). Используя функцию clock() для определения времени работы сервера. 4. Что будет в случае, если сервер завершит работу, не закрыв канал?

3 Теоретическое введение

Одним из видов взаимодействия между процессами в операционных системах является обмен сообщениями. Под сообщением понимается последовательность байтов, передаваемая от одного процесса другому. В операционных системах типа UNIX есть 3 вида межпроцессорных взаимодействий: общеюниксные (именованные каналы, сигналы), System V Interface Definition (SVID — разделяемая память, очередь сообщений, семафоры) и BSD (сокеты). Для передачи данных между неродственными процессами можно использовать механизм именованных каналов (named pipes). Данные передаются по принципу FIFO (First In First Out) (первым записан — первым прочитан), поэтому они называются также FIFO pipes или просто FIFO. 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. Файлы именованных каналов создаются функцией mkfifo(3).Первый параметр — имя файла, идентифицирующего канал, второй параметр — маска прав доступа к файлу. После создания файла канала процессы, участвующие в обмене данными, должны открыть этот файл либо для записи, либо для чтения. При закрытии файла сам канал продолжает существовать. Для того чтобы закрыть сам канал, нужно удалить его файл, например с помощью вызова unlink(2). Рассмотрим работу именованного канала на примере системы клиент–сервер. Сервер создаёт канал, читает из него текст, посылаемый клиентом, и выводит его на терминал. Вызов функции mkfifo() создаёт файл канала (с именем, заданным

макросом FIFO_NAME)

4 Выполнение лабораторной работы

```
/*
 * common.h - заголовочный файл со стандартными определениями
 */

#ifndef __COMMON_H__
#define __COMMON_H__

#include <stdio.h>
#include <stdlib.h>
```

Рис. 4.1: Изучите приведённые в тексте программы server.c и client.c. Взяв данные примеры за образец, напишите аналогичные программы, внеся следующие изменения

```
#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__ */
```

Рис. 4.2: Работает не 1 клиент, а несколько (например, два)

```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);
    }
```

Рис. 4.3:

```
  /* читаем данные из FIFO и выводим на экран */
  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); /* 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);
}
```

Рис. 4.4: Клиенты передают текущее время с некоторой периодичностью (например, раз в пять секунд). Используйте функцию sleep() для приостановки работы клиента.

```
#include "common.h"

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

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

  /* баннер */
  printf("FIFO Client...\n");

  /* получим доступ к FIFO */
  if((writefd = open(FIFO_NAME, O_WRONLY)) < 0)
```

Рис. 4.5:

```
    {
      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);
}
```

Рис. 4.6: Сервер работает не бесконечно, а прекращает работу через некоторое время (например, 30 сек). Используйте функцию clock() для определения времени работы сервера.

```
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
```

Рис. 4.7:

5 Выводы

Я приобрел практические навыки работы с именованными каналами