Лабораторная работа № 14. Именованные каналы

14.1. Цель работы

Приобретение практических навыков работы с именованными каналами.

14.2. Указания к работе

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)`.

```
#include <sys/types.h>
#include <sys/stat.h>

int mkfifo(const char *pathname, mode_t mode);

```

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`):

```

mkfifo(FIFO_NAME, 0600);

```

В качестве маски доступа используется восьмеричное значение 0600, разрешающее процессу с аналогичными реквизитами пользователя чтение и запись. Можно также установить права доступа 0666.

Открываем созданный файл для чтения:

```
f = fopen(FIFO_NAME, O_RDONLY);
```

Ждём сообщение от клиента. Сообщение читаем с помощью функции `read()` и печатаем на экран. После этого удаляется файл `FIFO_NAME` и сервер прекращает работу.

Клиент открывает FIFO для записи как обычный файл:

```
f = fopen(FIFO_NAME, O_WRONLY);
```

Посылаем сообщение серверу с помощью функции `write()`.

Для создания файла FIFO можно использовать более общую функцию `mknod(2)`, предназначенную для создания специальных файлов различных типов (FIFO, сокеты, файлы устройств и обычные файлы для хранения данных).

```
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <unistd.h>

int mknod(const char *pathname, mode_t mode, dev_t dev);
```

Тогда, вместо

```
mkfifo(FIFO_NAME, 0600);
```

пишем

```
mknod(FIFO_NAME, S_IFIFO | 0600, 0);
```

Каналы представляют собой простое и удобное средство передачи данных, которое, однако, подходит не во всех ситуациях. Например, с помощью каналов довольно трудно организовать обмен асинхронными сообщениями между процессами.

14.3. Пример программы

14.3.1. Файл 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__ */
```

14.3.2. Файл server.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);
    }
```

```
38
39    /* читаем данные из FIFO и выводим на экран */
40    while((n = read(readfd, buff, MAX_BUFF)) > 0)
41      {
42        if(write(1, buff, n) != n)
43          {
44            fprintf(stderr, "%s: Ошибка вывода (%s)\n",
45                    __FILE__, strerror(errno));
46            exit(-3);
47          }
48      }
49
50    close(readfd); /* закроем FIFO */
51
52    /* удалим FIFO из системы */
53    if(unlink(FIFO_NAME) < 0)
54      {
55        fprintf(stderr, "%s: Невозможно удалить FIFO (%s)\n",
56                __FILE__, strerror(errno));
57        exit(-4);
58      }
59
60    exit(0);
61  }
```

14.3.3. Файл client.c

```
1   /*
2    * client.c - реализация клиента
3    *
4    * чтобы запустить пример, необходимо:
5    *   1. запустить программу server на одной консоли;
6    *   2. запустить программу client на другой консоли.
7    */
8
9   #include "common.h"
10
11  #define MESSAGE "Hello Server!!!\n"
12
13  int
14  main()
15  {
16    int writefd; /* дескриптор для записи в FIFO */
17    int msglen;
18
19    /* баннер */
20    printf("FIFO Client...\n");
21
22    /* получим доступ к FIFO */
23    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);
}
```

14.3.4. Файл Makefile

```
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
```

14.4. 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?

14.5. Содержание отчёта

1. Титульный лист с указанием номера лабораторной работы и ФИО студента.

2. Формулировка цели работы.
3. Описание результатов выполнения задания:
   - скриншоты (снимки экрана), фиксирующие выполнение лабораторной работы;
   - листинги (исходный код) программ (если они есть);
   - результаты выполнения программ (текст или снимок экрана в зависимости от задания).
4. Выводы, согласованные с целью работы.
5. Ответы на контрольные вопросы.

14.6. Контрольные вопросы

1. В чем ключевое отличие именованных каналов от неименованных?
2. Возможно ли создание неименованного канала из командной строки?
3. Возможно ли создание именованного канала из командной строки?
4. Опишите функцию языка C, создающую неименованный канал.
5. Опишите функцию языка C, создающую именованный канал.
6. Что будет в случае прочтения из fifo меньшего числа байтов, чем находится в канале? Большего числа байтов?
7. Аналогично, что будет в случае записи в fifo меньшего числа байтов, чем позволяет буфер? Большего числа байтов?
8. Могут ли два и более процессов читать или записывать в канал?
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)?
10. Опишите функцию strerror.