Лабораторная работа № 14

Именованные каналы

Шулуужук Айраана Вячеславовна НПИбд-02-22

Содержание

Список иллюстраций

Список таблиц

1 Цель работы

Приобретение практических навыков работы с именованными каналами.

2 Задание

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 (напри- мер, 30 сек). Используйте функцию clock() для определения времени работы сервера.

3 Теоретическое введение

3.1 Этапы разработки приложений

Одним из видов взаимодействия между процессами в операционных системах яв- ляется 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 меха- низм 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 кли-

ент–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

Создадим нужный каталог и необходимые файлы (рис. 4.1)

```
[avshuluuzhuk@fedora os]$ mkdir lab14
[avshuluuzhuk@fedora os]$ cd lab14
[avshuluuzhuk@fedora lab14]$ touch common.h
[avshuluuzhuk@fedora lab14]$ touch server.c
[avshuluuzhuk@fedora lab14]$ touch client.c
[avshuluuzhuk@fedora lab14]$ touch Makefile
[avshuluuzhuk@fedora lab14]$
```

Рис. 4.1: создание файлов и каталога

Внесем тексты программ в файлы (рис. 4.2)

```
/*
 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__ */
```

Рис. 4.2: файл commom.h

В файле server.c введем дополнительную функцию clock для определения вре-

мени работы сервера (рис. 4.3)

```
        }
/* функция clock для определения времени работы сервера */

        clock_t now=time(NULL), start=time(NULL);
        while(now-start<30)
        {
                /* 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));
                        }
                }
                now=time(NULL)
        }
        printf("server timeout, %li - second passed\n",(now-start));
        close(readfd); /* закроем FIFO */
```

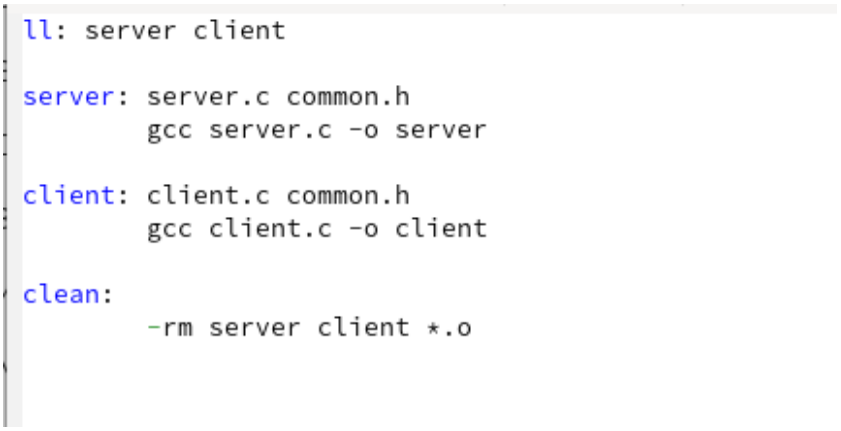
Рис. 4.3: файл server.c

В файле client.c введем некоторые изменения, клиенты передают текущее время с некоторой периодичностью (например, раз в пять секунд). Используем функцию sleep (рис. 4.4)

```
*/

#include "common.h"

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

int
main()
{
        int msg, len, i; /* дескриптор для записи в FIFO */
        long int i;
        for(i=0; i<20; i++)
        {
                /*функция sleep для приостановки работы клиента*/
                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);
                }
                /* передадим сообщение серверу */
                len = strlen(MESSAGE);
                if(write(msg, MESSAGE, len) != len)
                {
                        fprintf(stderr, "%s: Ошибка записи в FIFO (%s)\n",
                                __FILE__, strerror(errno));
                        exit(-2);
                }
                /* закроем доступ к FIFO */
                close(msg);
        }
        exit(0);
}
```

Рис. 4.4: файл client.c

Создаем файл Makefile (рис. 4.5)



```
ll: 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. 4.5: Makefile

Скомпилируем программу, используюя Makefile (рис. 4.6)

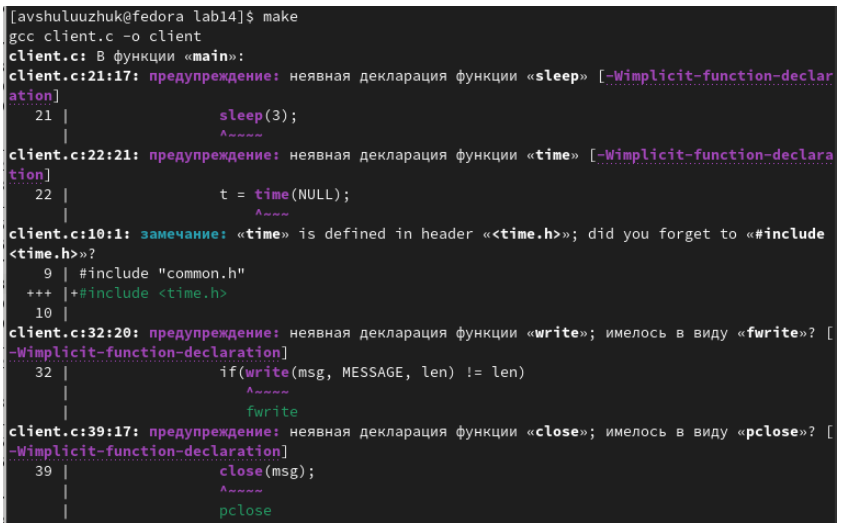


Рис. 4.6: компилирование

Запустим командные файлы server и client (рис. 4.7)

Рис. 4.7: запуск командных файлов

5 Выводы

В ходе выполнения работы мы приобрели практические навыки работы с именованными каналами.