Отчет

Лабораторная работа №14

Павлова Варвара Юрьевна НПМбд-02-21

Содержание

Список иллюстраций

Список таблиц

Цель работы

Приобретение практических навыков работы с именованными каналами.

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. Что будет в случае, если сервер завершит работу, не закрыв канал?

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.

Выполнение лабораторной работы

1. Создаю необходимые файлы с помощью команды touch.(рис. [-@fig:001])

```
[vypavlova@fedora ~]$ touch common.h server.c client.c Makefile
```

Рис. 0.1: создание файлов

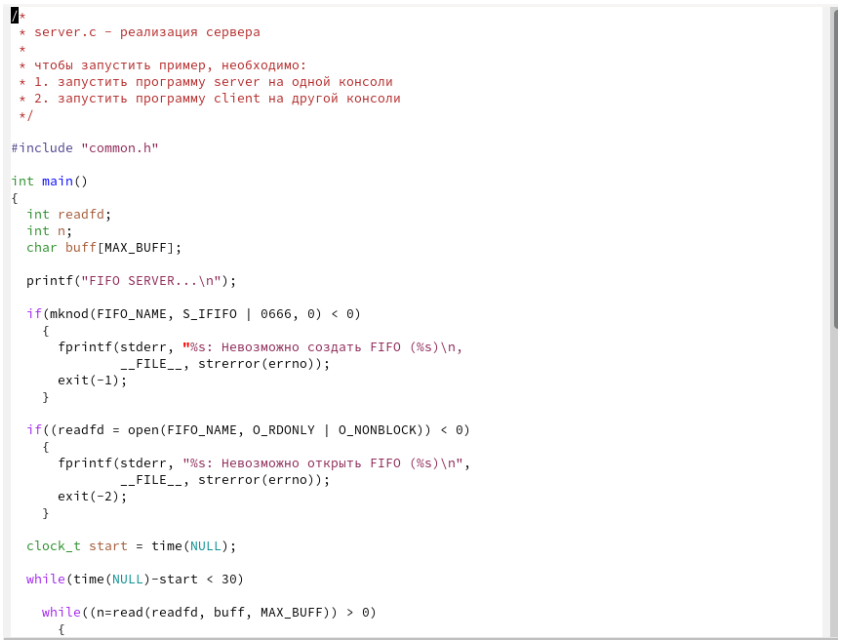
2. В файл common.h добавляю стандартные заголовочные файлы time.h и unistd.h (рис. [-@fig:002])

```
/*
 * 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>
#include <time.h>
#include <unistd.h>

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

#endif /* __COMMON_H__ */
```

Рис. 0.2: common.h

3. В файл server.c добавляю цикл while для контроля за временем работы сервера.

Разница между текущим временем и временем начала работы не должна превышать 30 секунд. (рис. [-@fig:003]) (рис. [-@fig:004])



```
/*
 * 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;
  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 | O_NONBLOCK)) < 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)
      {
```

Рис. 0.3: server.c

```
  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);
          }
      }

  closw (readfd);

  if(unlink(FIFO_NAME) < 0)
    {
      fprintf(stderr, "%s: Невозможно удалить FIFO (%S)\n",
              __FILE__, strerror(errn0));
      exit(-4);
    }
  exit(0);
}
```

Рис. 0.4: server.c

4. В файл client.c добавляю цикл, который отвечает за количество сообщений о текущем времени и команду sleep(5) для приостановки работы клиента на 5 секунд. (рис. [-@fig:005]) (рис. [-@fig:006])

```c
/*
 * client.c
 *
 * Чтобы запустить пример, необходимо:
 * 1. запустить программу server на одной консоли;
 * 2. запустить программу client на другой консоли.
 */
#include "common.h"

int main()
{
  int writefd;
  int msglen;

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

  int i=0;
  for (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);
          break;
        }
      long int ttime=time(NULL);
      char* text=ctime(&ttime);
```

Рис. 0.5: client.c

```c
      long int ttime=time(NULL);
      char* text=ctime(&ttime);

      msglen = strlen(text);
      if(write(writefd, text, msglen) != msglen)
        {
          fprintf(stderr, "%s: Ошибка записи в FIFO (%s)\n",
                  __FILE__, strerror(errno));
          exit(-2);
        }
      sleep(5);
    }

  close(writefd);
  exit(0);
}
```

Рис. 0.6: client.c

3. Файл для сборки - Makefile - не изменяю. (рис. [-@fig:007])

```
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. 0.7: Makefile

4. Выполняю команду make all и компилирую необходимые файлы. (рис. [-@fig:008])

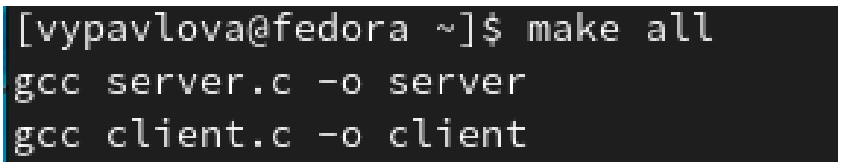


Рис. 0.8: компиляция

5. Проверяю работу написанного кода. Открываю 3 консоли и запускаю в одном из них ./server(рис. [-@fig:009]) , а в двух других - ./client. (рис. [-@fig:010]) (рис. [-@fig:011])

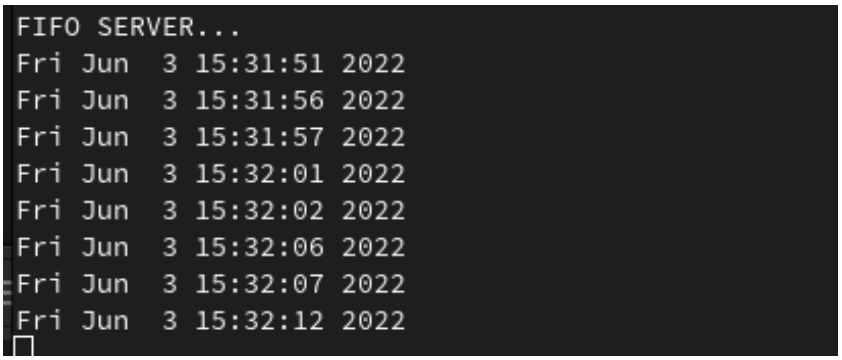


Рис. 0.9: ./server

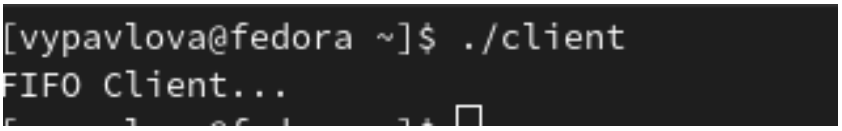


Рис. 0.10: ./client

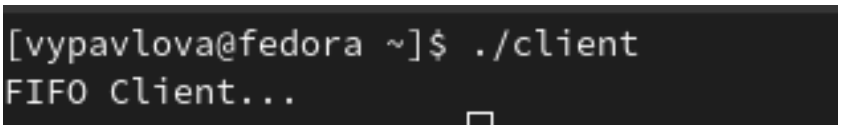


Рис. 0.11: ./client

6. Если сервер завершит работу, не закрыв канал, то при следующей попытке запуска будет выдана ошибка “Невозможно создать FIFO”.

Выводы

Выполняя данную лабораторную работу я приобрела практические навыки работы с именованными каналами.

Список литературы