

Лабораторная работа №14

Именованные каналы

Ариоке Габриэль Одафе.

Российский университет дружбы народов, Москва, Россия

Приобретение практических навыков работы с именованными каналами.

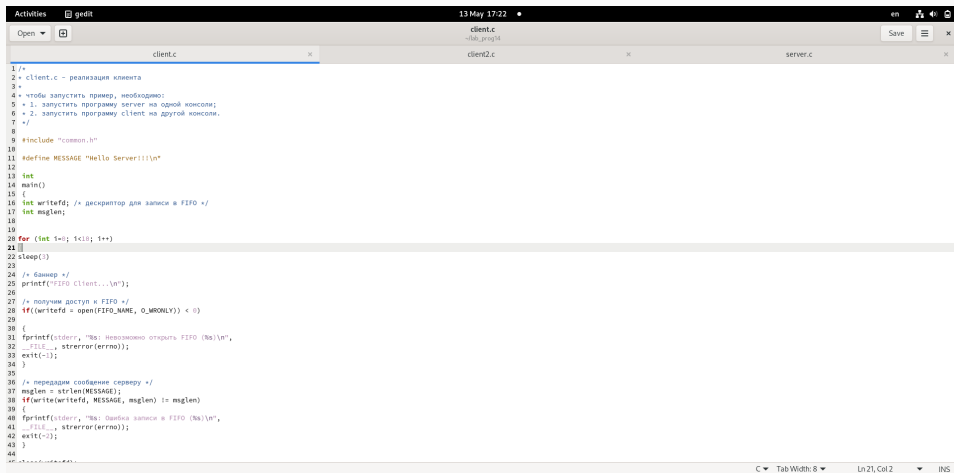
Изучите приведённые в тексте программы `server.c` и `client.c`. Взяв данные примеры за образец, напишите аналогичные программы, внося следующие изменения: 1. Работает не 1 клиент, а несколько (например, два). 2. Клиенты передают текущее время с некоторой периодичностью (например, раз в пять секунд). 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?

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: общедоступные (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)`.

Изучите приведённые в тексте программы `server.c` и `client.c`. Взяв данные примеры за образец, напишите аналогичные программы, внося 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. (рис. 1, 2)

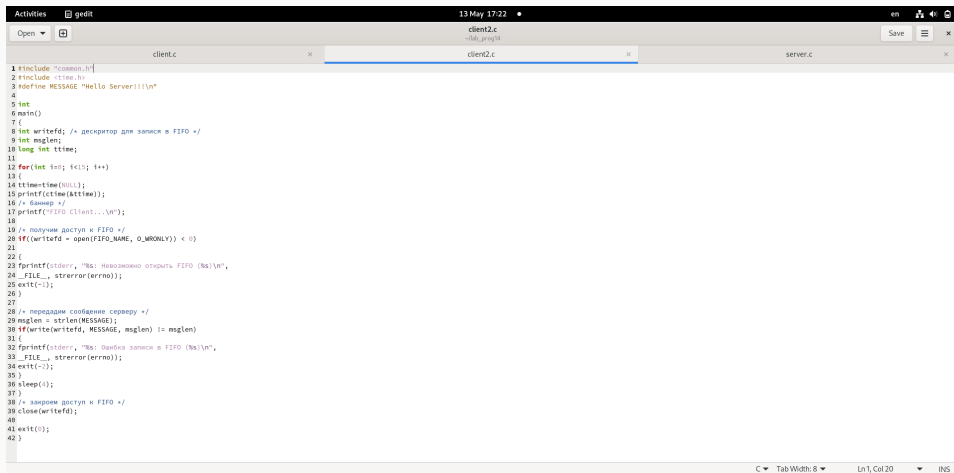
Выполнение лабораторной работы



```
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!!!"
12
13 int
14 main()
15 {
16     int writefd; /* дескриптор для записи в FIFO */
17     int msglen;
18
19
20     for (int i=0; i<10; i++)
21     {
22         sleep(3)
23
24         /* баннер */
25         printf("FIFO Client...\n");
26
27         /* получим доступ к FIFO */
28         if((writefd = open(FIFO_NAME, O_WRONLY)) < 0)
29         {
30             fprintf(stderr, "%s: Невозможно открыть FIFO (%s)\n",
31                     __FILE__, strerror(errno));
32             exit(-1);
33         }
34
35         /* передадим сообщение серверу */
36         msglen = strlen(MESSAGE);
37         if(write(writefd, MESSAGE, msglen) != msglen)
38         {
39             fprintf(stderr, "%s: Ошибка записи в FIFO (%s)\n",
40                     __FILE__, strerror(errno));
41             exit(-2);
42         }
43     }
44 }
```

Figure 1: Текст программы

Выполнение лабораторной работы



```
1 #include "common.h"
2 #include <time.h>
3 #define MESSAGE "Hello Server!!!!\n"
4
5 int
6 main()
7 {
8     int writefd; /* дескриптор для записи в FIFO */
9     int msglen;
10    long int ttime;
11
12    for(int i=0; i<15; i++)
13    {
14        ttime=time(NULL);
15        printf("ctime(&ttime));
16        /* БANNER */
17        printf("FIFO Client...\n");
18
19        /* получим доступ к FIFO */
20        if((writefd = open(FIFO_NAME, O_WRONLY)) < 0)
21
22        {
23            fprintf(stderr, "%s: Невозможно открыть FIFO (%s)\n",
24                __FILE__, strerror(errno));
25            exit(-1);
26        }
27
28        /* передадим сообщение серверу */
29        msglen = strlen(MESSAGE);
30        if(write(writefd, MESSAGE, msglen) != msglen)
31        {
32            fprintf(stderr, "%s: Ошибка записи в FIFO (%s)\n",
33                __FILE__, strerror(errno));
34            exit(-2);
35        }
36        sleep(4);
37    }
38    /* закроем доступ к FIFO */
39    close(writefd);
40
41    exit(0);
42 }
```

Figure 2: Текст программы

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? (рис. 3, 4, 5, 6)

Выполнение лабораторной работы

```
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67
```

```
24 if(mknod(FIFO_NAME, S_IFIFO | 0666, 0) < 0)
25 {
26     fprintf(stderr, "%s: Невозможно создать FIFO (%s)\n",
27         __FILE__, strerror(errno));
28     exit(-1);
29 }
30
31 /* откроем FIFO на чтение */
32 if((readfd = open(FIFO_NAME, O_RDONLY)) < 0)
33 {
34     fprintf(stderr, "%s: Невозможно открыть FIFO (%s)\n",
35         __FILE__, strerror(errno));
36     exit(-2);
37 }
38
39 clock_t now=time(NULL), clock_t start=time(NULL);
40 while (now-start<30)
41 {
42     /* читаем данные из FIFO и выводим на экран */
43     while((n = read(readfd, buff, MAX_BUFF)) > 0)
44     {
45         if(write(1, buff, n) != n)
46         {
47             fprintf(stderr, "%s: Ошибка вывода (%s)\n",
48                 __FILE__, strerror(errno));
49             exit(-3);
50         }
51     }
52     now=time(NULL);
53 }
54
55 close(readfd); /* закроем FIFO */
56
57 /* удалим FIFO из системы */
58 if(unlink(FIFO_NAME) < 0)
59 {
60     fprintf(stderr, "%s: Невозможно удалить FIFO (%s)\n",
61         __FILE__, strerror(errno));
62     exit(-4);
63 }
64
65 exit(0);
66 }
```

Figure 3: Текст программы

Выполнение лабораторной работы

```

1 all: server client
2
3 server: server.c common.h
4     gcc server.c -o server
5
6 client: client.c common.h
7     gcc client.c -o client
8
9 client2: client2.c common.h
10    gcc client2.c -o client
11
12 clean:
13     -rm server client *.o

```

Makefile

~lab_prog14

Save

Loading file "/home/pcaladi/lab_prog14/Makefile"...

Makefile Tab Width: 8 Ln 1, Col 1 INS

Figure 4: Текст программы

Выполнение лабораторной работы

```

1 /*
2 * common.h - заголовочный файл со стандартными определениями
3 */
4
5 #ifndef __COMMON_H__
6 #define __COMMON_H__
7
8 #include <stdio.h>
9 #include <stdlib.h>
10 #include <string.h>
11 #include <errno.h>
12 #include <sys/types.h>
13 #include <sys/stat.h>
14 #include <fcntl.h>
15
16 #define FIFO_NAME "/tmp/fifo"
17 #define MAX_BUFF 88
18
19 #endif /* __COMMON_H__ */

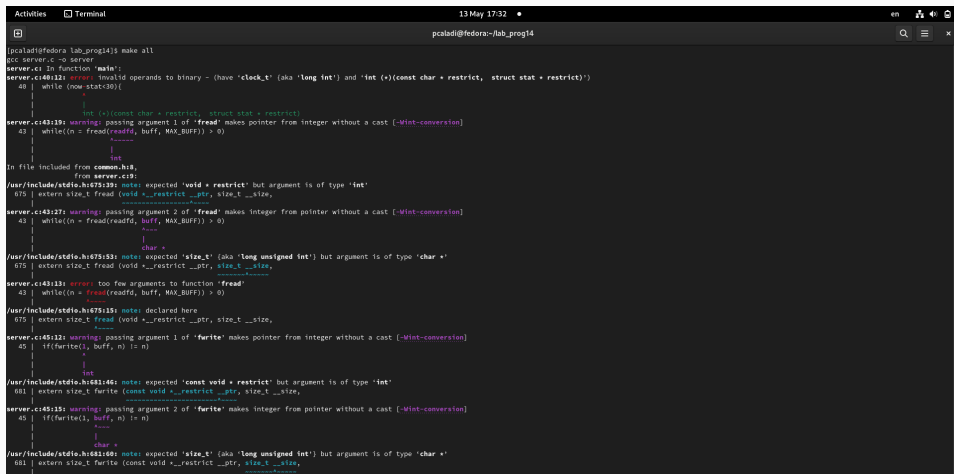
```

Loading file "/home/pcaladi/lab_prog14/common.h"...

C/ObjC Header Tab Width: 8 Ln 1, Col 1 INS

Figure 5: Компиляция

Выполнение лабораторной работы



```
pcaladi@fedora:~/lab_prog14$ make all
gcc server.c -o server
server.c: In function 'main':
server.c:40:12: error: invalid operands to binary - (have 'clock_t' (aka 'long int') and 'int (*)(const char * restrict, struct stat * restrict)')
   40 | while (now-stat<30){
      |         ^
      |         |
      |         int (*)(const char * restrict, struct stat * restrict)
server.c:43:19: warning: passing argument 1 of 'fread' makes pointer from integer without a cast [-Wint-conversion]
   43 | while((n = fread(readfd, buff, MAX_BUFFER)) > 0)
      |                   ^~~~~~
      |                   |
      |                   int
In file included from common.h:8,
                 from server.c:8:
/usr/include/stdio.h:675:39: note: expected 'void * restrict' but argument is of type 'int'
   675 | extern size_t fread (void *__restrict __ptr, size_t __size,
      |                       ^~~~~~
server.c:43:17: warning: passing argument 2 of 'fread' makes integer from pointer without a cast [-Wint-conversion]
   43 | while((n = fread(readfd, buff, MAX_BUFFER)) > 0)
      |                   ^~~~~~
      |                   |
      |                   char *
/usr/include/stdio.h:675:53: note: expected 'size_t' (aka 'long unsigned int') but argument is of type 'char *'
   675 | extern size_t fread (void *__restrict __ptr, size_t __size,
      |                       ^~~~~~
server.c:43:13: error: too few arguments to function 'fread'
   43 | while((n = fread(readfd, buff, MAX_BUFFER)) > 0)
      |                   ^~~~~~
/usr/include/stdio.h:675:15: note: declared here
   675 | extern size_t fread (void *__restrict __ptr, size_t __size,
      |                       ^~~~~~
server.c:45:12: warning: passing argument 1 of 'fwrite' makes pointer from integer without a cast [-Wint-conversion]
   45 | if(fwrite(1, buff, n) != n)
      |         ^
      |         |
      |         int
/usr/include/stdio.h:681:46: note: expected 'const void * restrict' but argument is of type 'int'
   681 | extern size_t fwrite (const void *__restrict __ptr, size_t __size,
      |                       ^~~~~~
server.c:45:15: warning: passing argument 2 of 'fwrite' makes integer from pointer without a cast [-Wint-conversion]
   45 | if(fwrite(1, buff, n) != n)
      |                   ^~~~~~
      |                   |
      |                   char *
/usr/include/stdio.h:681:60: note: expected 'size_t' (aka 'long unsigned int') but argument is of type 'char *'
   681 | extern size_t fwrite (const void *__restrict __ptr, size_t __size,
```

Figure 6: Результат

В процессе выполнения лабораторной работы я приобрел практические навыки работы с именованными каналами.

1. Лабораторная работа № 14. Именованные каналы [Электронный ресурс]. URL: <https://esystem.rudn.ru/>.

Спасибо за внимание!