

Лабораторная работа №15

Журавлев Георгий Иванович

Цель работы

Приобретение практических навыков работы с именованными каналами #
Ход работы.

1. Ознакомился с программами и на их основе написал свои, добавил 1 клиент.

1.1. common.h(header file)

```
1  #ifndef  __COMMON_H__
2  #define  __COMMON_H__
3  #include <stdio.h>
4  #include <stdlib.h>
5  #include <string.h>
6  #include <unistd.h>
7  #include <errno.h>
8  #include <sys/types.h>
9  #include <sys/stat.h>
10 #include <fcntl.h>
11 #include <time.h>
12 #define FIFO_NAME "Myfifo"
13 #define MAX_BUFF 80
14 #endif /* __COMMON_H__ */
15
```

1.2. server.c.

```

1  #include "common.h"
2
3  int main(){
4
5      int readfd;
6      char buff[MAX_BUFF];
7      int n;
8
9      printf("FIFO Server...\n");
10     if(mkfifo(FIFO_NAME,0777)==-1){
11         if(errno!=EEXIST){
12             printf("impossible to create fifo\n");
13             return 1;
14         };
15     }
16
17     printf("Open...\n");
18     if((readfd = open(FIFO_NAME,O_RDONLY))==-1){
19         printf("can't open");
20         return 2;
21     };
22     printf("opened\n");
23
24     while((n = read(readfd,buff,MAX_BUFF))>0){
25         if(write(1,buff,n)!=n){
26             printf("can't open");
27             return 3;
28         }
29     }
30     close(readfd);
31
32     if(unlink(FIFO_NAME)==-1){
33         printf("can't delete");
34         return 4;
35     }
36     return 0;
37 }

```

1.3. client1.c.

```
1  #include "common.h"
2  #define MESSAGE "Lab15,finally!!!\n"
3
4  int main()
5  {
6      int writefd;
7      int msglen;
8
9      printf("FIFO Client 1...\n");
10
11     if((writefd = open(FIFO_NAME, O_WRONLY))==-1)
12     {
13         printf("can't open");
14         return 1;
15     }
16
17     msglen = strlen(MESSAGE);
18
19     int n=0;
20     while(n<5){
21         if(write(writefd, MESSAGE, msglen) != msglen)
22         {
23             printf("can't write");
24             return 2;
25         }
26         printf("next message will appear in 5\n");
27         * seconds\n");
28         sleep(5);
29         n++;
30     }
31
32     close(writefd);
33     return 0;
34 }
```

1.4. client2.c.

```

1  #include "common.h"
2  #define MESSAGE "not yet\n"
3
4  int main()
5  {
6      int writefd;
7      int msglen;
8
9      printf("FIFO Client 2...\n");
10
11     if((writefd = open(FIFO_NAME, O_WRONLY))==-1)
12     {
13         printf("can't open");
14         return 1;
15     }
16
17     msglen = strlen(MESSAGE);
18     int n=0;
19     while(n<5){
20         if(write(writefd, MESSAGE, msglen) != msglen)
21         {
22             printf("can't write");
23             return 2;
24         }
25         printf("next message will appear in 5 seconds\n");
26         sleep(5);
27         n++;
28     }
29
30     close(writefd);
31     return 0;
32 }

```

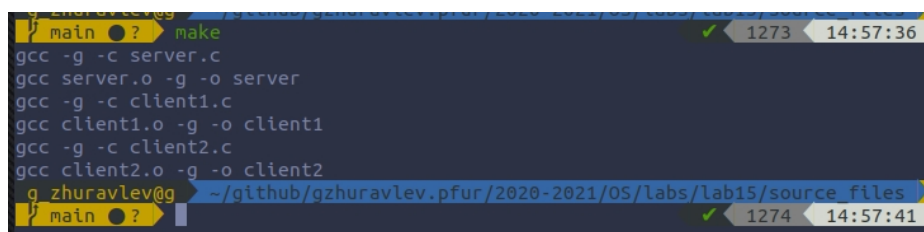
1.5. Makefile

```

1  CC=gcc
2  CFLAGS= -g -c
3  CompileFlags= -g -o
4
5  all: server client1 client2
6
7  server: server.o
8      $(CC) server.o $(CompileFlags) server
9
10 client1: client1.o
11     $(CC) client1.o $(CompileFlags) client1
12
13 client2: client2.o
14     $(CC) client2.o $(CompileFlags) client2
15
16 server.o: server.c common.h
17     $(CC) $(CFLAGS) server.c
18
19 client1.o: client1.c common.h
20     $(CC) $(CFLAGS) client1.c
21
22 client2.o: client2.c common.h
23     $(CC) $(CFLAGS) client2.c
24
25 clean:
26     rm -rf *.o client1 client2 server

```

1.6. Компиляция.



```

gzhuravlev@ ~/github/gzhuravlev.pfur/2020-2021/OS/labs/lab15/source_files
main ●? make
gcc -g -c server.c
gcc server.o -g -o server
gcc -g -c client1.c
gcc client1.o -g -o client1
gcc -g -c client2.c
gcc client2.o -g -o client2
gzhuravlev@ ~/github/gzhuravlev.pfur/2020-2021/OS/labs/lab15/source_files
main ●?

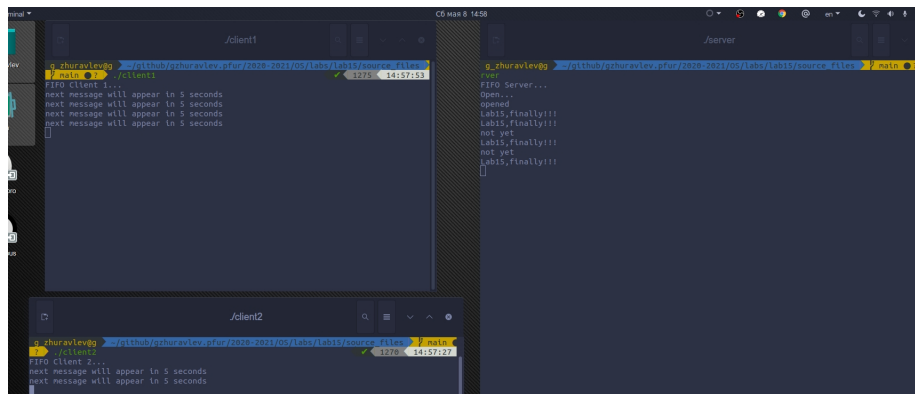
```

2. Добавил функцию задержки `sleep(5)` (сообщение появляется 5 раз)

2.1. Реализация.

```
int n=0;
while(n<5){
    if(write(writefd, MESSAGE, msglen) != msglen)
    {
        printf("can't write");
        return 2;
    }
    printf("next message will appear in 5
seconds\n");
    sleep(5);
    n++;
}
```

2.2. В действии.



```
g_zhuravlev@ ~/github/zhuravlev_rfm/2020-2021/05/lab5/lab5/source_files: ./server
FIFO Server...
Open...
opened
Lab5,Finally!!!
not get
Lab5,Finally!!!
not get
Lab5,Finally!!!

g_zhuravlev@ ~/github/zhuravlev_rfm/2020-2021/05/lab5/lab5/source_files: ./client1
FIFO client 1...
next message will appear in 5 seconds
next message will appear in 5 seconds
next message will appear in 5 seconds
next message will appear in 5 seconds

g_zhuravlev@ ~/github/zhuravlev_rfm/2020-2021/05/lab5/lab5/source_files: ./client2
FIFO client 2...
next message will appear in 5 seconds
next message will appear in 5 seconds
```

2.3. Завершение.

```
g_zhuravlev@g_zhuravlev.pfur/2020-2021/OS/labs/lab15/source_files
$ ./client1
FIFO Client 1...
next message will appear in 5 seconds
next message will appear in 5 seconds
next message will appear in 5 seconds
next message will appear in 5 seconds
next message will appear in 5 seconds
$ ./client2
FIFO Client 2...
next message will appear in 5 seconds
next message will appear in 5 seconds
next message will appear in 5 seconds
next message will appear in 5 seconds
next message will appear in 5 seconds
$ ./server
FIFO Server...
Open...
opened
Lab15,finally!!!
Lab15,finally!!!
not yet
Lab15,finally!!!
not yet
Lab15,finally!!!
not yet
Lab15,finally!!!
not yet
not yet
not yet
$
```

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

Ответ: Ошибка.

Вывод.

Благодаря этой лабораторной работе, я приобрел практические навыки работы с именованными каналами.

Контрольные вопросы.

1. Именованные каналы отличаются от не именованных наличием идентификатора канала, который пре...
2. нет
3. да, например с помощью функции `mkfifo(FIFO_NAME,MODE)` в терминале.
4. `int pipe(int fd[2])` - 2 файловых дескриптора(чтение и запись).
5. `mkfifo(FIFO_NAME,MODE)`
6. Смотря на пример из лаб15 -> произойдет ошибка при чтении.
7. Смотря на пример из лаб15 -> произойдет ошибка при записи.
8. При технологии FIFO да, но это будет неудобно.(лучший вариант - один на чтение, один на запись.)
9. `write` имеет следующую логику: `write(fd, buffer, count)`, где `buffer` - записываемые файлы; `count` - байты; `fd` - file descriptor; ->
`write(1,buff,n)` - 1-fd запись(0-чтение, 1-запись); `buff`- записываемые данные; `n` - кол-во записываемых данных.

10. `strerror()` возвращает указатель на сообщение об ошибке, связанное с номером ошибки. (`errno` - number of error).