

Лабораторная работа №15 по предмету
Операционные системы

Группа НПМбв-02-19

Нечаева Виктория Алексеевна

Содержание

Цель работы	5
Задание	6
Выполнение лабораторной работы	7
Выводы	13
Контрольные вопросы	14

Список таблиц

Список иллюстраций

1	Рисунок 1 - Как работает оригинальный FIFO	7
2	Рисунок 2	8
3	Рисунок 3 – client.c	9
4	Рисунок 4 – server.c	10
5	Рисунок 5 – common.h	11
6	Рисунок 6 – Что будет, есть прервать сервер	12

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, 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?

Выполнение лабораторной работы

Ниже – как работает версия из лабораторной работы.

```
vanechaeva@vanechaeva: ~/fifo
pclose'? [-Wimplicit-function-declaration]
43 | close(readfd); /* закроем FIFO */
    | ^~~~~~
server.c:45:4: warning: implicit declaration of function 'unlink' [-Wimplicit-fu
nction-declaration]
45 | if(unlink(FIFO_NAME) < 0)
    | ^~~~~~
gcc client.c -o client
client.c: In function 'main':
client.c:28:4: warning: implicit declaration of function 'write'; did you m
fwrite'? [-Wimplicit-function-declaration]
26 | if(write(writefd, MESSAGE, msglen) != msglen)
    | ^~~~~~
client.c:33:1: warning: implicit declaration of function 'close'; did you m
pclose'? [-Wimplicit-function-declaration]
33 | close(writefd);
    | ^~~~~~
vanechaeva@vanechaeva:~/fifo$ ./server
FIFO Server...
Hello Server!!!
vanechaeva@vanechaeva:~/fifo$
```

3-ipc-fifo.pdf

ота № 13. Именованные каналы

```
vanechaeva@vanechaeva:~$ ./client
bash: ./client: Нет такого файла или каталога
vanechaeva@vanechaeva:~$ cd fifo
vanechaeva@vanechaeva:~/fifo$ ./client
FIFO client...
vanechaeva@vanechaeva:~/fifo$
```

– описание результатов выполнения задания:
– листинги программ;
– результаты выполнения программ (текст от задания);
– выводы, согласованные с целью работы;
– ответы на контрольные вопросы.

13.6. Контрольные вопросы

Рис. 1: Рисунок 1 - Как работает оригинальный FIFO

Как работает версия с требованиями из задания (рис.2)

```
vanetchaeva@vanetchaeva: /tmp
vanetchaeva@vanetchaeva:~$ cd ..
vanetchaeva@vanetchaeva:~/home$ cd ..
vanetchaeva@vanetchaeva:/$ cd tmp
vanetchaeva@vanetchaeva:~/tmp$ rm fifo
vanetchaeva@vanetchaeva:~/fifo$


client client.c common.h Makefile server server.c
vanetchaeva@vanetchaeva:~/fifo$ ./server
FIFO Server...
2023-03-28 15:56:41 TTY1
2023-03-28 15:56:46 TTY1
2023-03-28 15:56:51 TTY1
2023-03-28 15:56:56 TTY1
2023-03-28 15:57:01 TTY1
2023-03-28 15:57:06 TTY1
2023-03-28 15:57:11 TTY1
vanetchaeva@vanetchaeva:~/fifo$ ./server
FIFO Server...
2023-03-28 15:57:35 TTY1
2023-03-28 15:57:40 TTY1
2023-03-28 15:57:43 TTY3
2023-03-28 15:57:45 TTY1
2023-03-28 15:57:48 TTY3
2023-03-28 15:57:50 TTY1
2023-03-28 15:57:53 TTY3
2023-03-28 15:57:55 TTY1
2023-03-28 15:57:58 TTY3
2023-03-28 15:58:00 TTY1
2023-03-28 15:58:03 TTY3
2023-03-28 15:58:05 TTY1
vanetchaeva@vanetchaeva:~/fifo$

vanetchaeva@vanetchaeva:~$ tty
/dev/pts/3
vanetchaeva@vanetchaeva:~$ ./client
bash: ./client: Нет такого файла или каталога
vanetchaeva@vanetchaeva:~$ cd fifo
vanetchaeva@vanetchaeva:~/fifo$ ./client
FIFO Client...
vanetchaeva@vanetchaeva:~/fifo$

vanetchaeva@vanetchaeva:~/fifo$ tty
/dev/pts/1
vanetchaeva@vanetchaeva:~/fifo$ ./client
FIFO Client...
vanetchaeva@vanetchaeva:~/fifo$
```

Рис. 2: Рисунок 2

На скринкасте в действии работа fifo на 43-ей минуте. Канал открыт на 30 секунд, запущены два клиента, каждый из них каждые 5 секунд отчитывается о работе серверу. По истечении канала закрывается и по очереди клиенты завершают работу - сначала тот, который пришел последним, потом первый.



```
GNU nano 6.2 client.c
#include "common.h"
#define MESSAGE_SIZE 50
#define MESSAGE "Hello Server!!!\n"
#define SLEEP_TIME 5

int main()
{
    int writefd; /* дескриптор для записи в FIFO */
    int msglen;
    /* баннер */
    printf("FIFO Client...\n");
    /* получим доступ к FIFO */
    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);
    char* tty_path = ttyname(STDOUT_FILENO);
    char tty_num = tty_path[strlen(tty_path) - 1];
    char message[MESSAGE_SIZE];
    while (1) {
        time_t t = time(NULL);
        struct tm* tm_info = localtime(&t);
        strftime(message, MESSAGE_SIZE, "%Y-%m-%d %H:%M:%S", localtime(&t));
        sprintf(message, "%s TTY%d", message, tty_num - '0');
        strcat(message, "\n");
        if(write(writefd, message, strlen(message)) != strlen(message))
        {
            fprintf(stderr, "%s: Ошибка записи в FIFO (%s)\n",
                __FILE__, strerror(errno));
            exit(-2);
        }
        sleep(SLEEP_TIME);
    }
    /* закроем доступ к FIFO */
    close(writefd);
    exit(0);
}
```

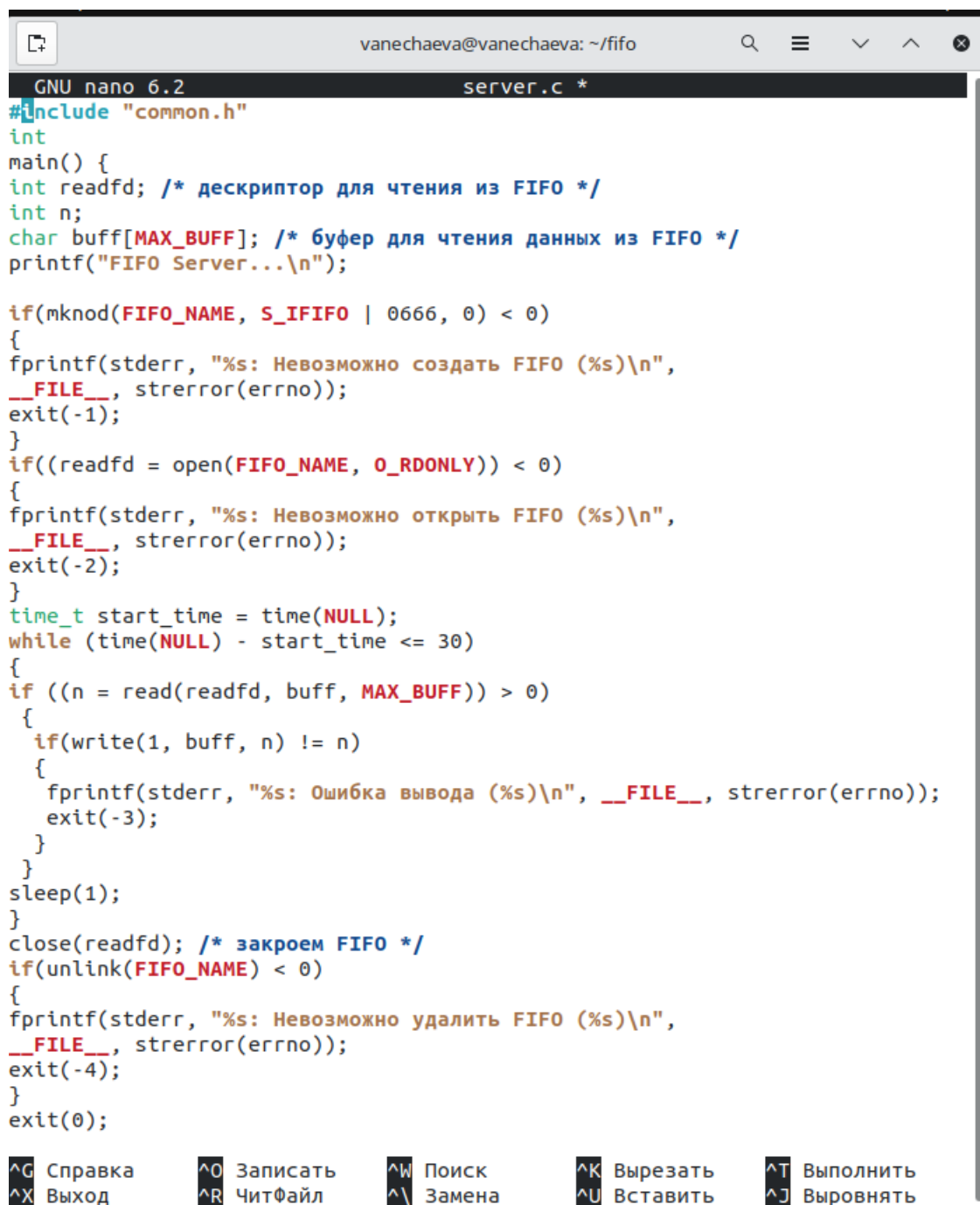
[Записана 41 строка]

^G Справка	^O Записать	^W Поиск	^K Вырезать	^T Выполнить	^C Позиция
^X Выход	^R ЧитФайл	^_ Замена	^U Вставить	^J Выводить	^/ К строке

Рис. 3: Рисунок 3 – client.c

В client.c добавлены `SLEEP_TIME = 5` и `MESSAGE_SIZE = 50`, добавлен бесконечный цикл `while` оканчивающийся с закрытием сервера. В цикле определяется локальное время для вывода по каналу времени, чтобы было понятно, что кли-

енты обмениваются информацией по каналу каждые 5 секунд. С помощью `tty_path` и `tty_num` узнаем номер клиента (терминала) (рис.3)



```
GNU nano 6.2 server.c *
#include "common.h"
int
main() {
int readfd; /* дескриптор для чтения из FIFO */
int n;
char buff[MAX_BUFF]; /* буфер для чтения данных из FIFO */
printf("FIFO Server...\n");

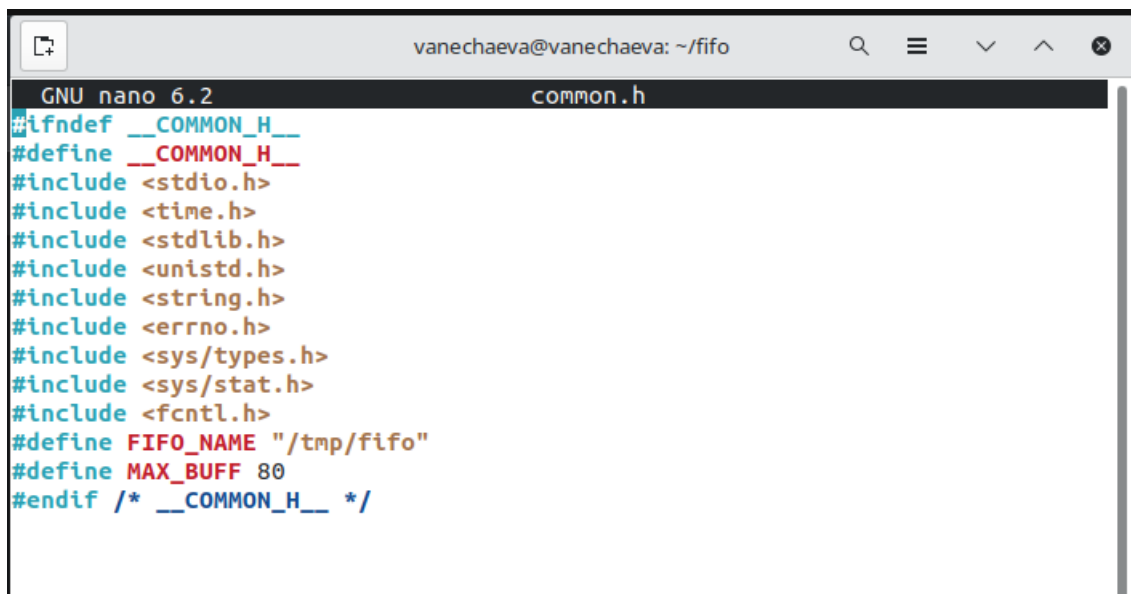
if(mknod(FIFO_NAME, S_IFIFO | 0666, 0) < 0)
{
fprintf(stderr, "%s: Невозможно создать FIFO (%s)\n",
__FILE__, strerror(errno));
exit(-1);
}
if((readfd = open(FIFO_NAME, O_RDONLY)) < 0)
{
fprintf(stderr, "%s: Невозможно открыть FIFO (%s)\n",
__FILE__, strerror(errno));
exit(-2);
}
time_t start_time = time(NULL);
while (time(NULL) - start_time <= 30)
{
if ((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);
}
}
sleep(1);
}
close(readfd); /* закроем FIFO */
if(unlink(FIFO_NAME) < 0)
{
fprintf(stderr, "%s: Невозможно удалить FIFO (%s)\n",
__FILE__, strerror(errno));
exit(-4);
}
exit(0);

^G Справка      ^O Записать
^X Выход        ^R ЧитФайл
^W Поиск       ^_ Замена
^K Вырезать    ^U Вставить
^T Выполнить   ^J Выровнять
```

Рис. 4: Рисунок 4 – `server.c`

В `server.c` задано время работы сервера – 30 секунд, в цикле `while(time(NULL)...)`

(рис.4)



```
GNU nano 6.2 common.h
#ifndef __COMMON_H__
#define __COMMON_H__
#include <stdio.h>
#include <time.h>
#include <stdlib.h>
#include <unistd.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. 5: Рисунок 5 – common.h

В common.h добавлены <unistd.h> и <time.h> (рис.5)

```
vanechaeva@vanechaeva:~/fifo$ ./server
FIFO Server...
2023-03-28 15:56:41 TTY1
2023-03-28 15:56:46 TTY1
2023-03-28 15:56:51 TTY1
2023-03-28 15:56:56 TTY1
2023-03-28 15:57:01 TTY1
2023-03-28 15:57:06 TTY1
2023-03-28 15:57:11 TTY1
vanechaeva@vanechaeva:~/fifo$ ./server
FIFO Server...
2023-03-28 15:57:35 TTY1
2023-03-28 15:57:40 TTY1
2023-03-28 15:57:43 TTY3
2023-03-28 15:57:45 TTY1
2023-03-28 15:57:48 TTY3
2023-03-28 15:57:50 TTY1
2023-03-28 15:57:53 TTY3
2023-03-28 15:57:55 TTY1
2023-03-28 15:57:58 TTY3
2023-03-28 15:58:00 TTY1
2023-03-28 15:58:03 TTY3
2023-03-28 15:58:05 TTY1
vanechaeva@vanechaeva:~/fifo$ nano client.c
vanechaeva@vanechaeva:~/fifo$ nano server.c
vanechaeva@vanechaeva:~/fifo$ nano common.h
vanechaeva@vanechaeva:~/fifo$ ./server
FIFO Server...
2023-03-28 16:00:56 TTY1
^X2023-03-28 16:01:01 TTY1
^C
vanechaeva@vanechaeva:~/fifo$ ./server
FIFO Server...
server.c: Невозможно создать FIFO (File exists)
vanechaeva@vanechaeva:~/fifo$
```

```
vanechaeva@vanechaeva:~/fifo$ ls
fifo
snap-private-tmp
systemd-private-3df44524159f4d02a036d727d2d0b8aa-colord.service-ht3LLW
systemd-private-3df44524159f4d02a036d727d2d0b8aa-fwupd.service-zWDWLq
systemd-private-3df44524159f4d02a036d727d2d0b8aa-haveged.service-30FcYt
systemd-private-3df44524159f4d02a036d727d2d0b8aa-ModemManager.service-HvMQT5
systemd-private-3df44524159f4d02a036d727d2d0b8aa-power-profiles-daemon.service-U
WF-Cw
systemd-private-3df44524159f4d02a036d727d2d0b8aa-switcheroo-control.service-vqLZ
WX
systemd-private-3df44524159f4d02a036d727d2d0b8aa-systemd-logind.service-Ydi8Wg
systemd-private-3df44524159f4d02a036d727d2d0b8aa-systemd-oomd.service-ox0VhA
systemd-private-3df44524159f4d02a036d727d2d0b8aa-systemd-resolved.service-jAsr3W
systemd-private-3df44524159f4d02a036d727d2d0b8aa-upower.service-g3AvXn
tracker-extract-3-files.1000
tracker-extract-3-files.128
VMware0n0
vanechaeva@vanechaeva:~/tmp$
```

```
vanechaeva@vanechaeva:~/fifo$ tty
/dev/pts/1
vanechaeva@vanechaeva:~/fifo$ ./client
FIFO Client...
vanechaeva@vanechaeva:~/fifo$ ./client
FIFO Client...
vanechaeva@vanechaeva:~/fifo$
```

Рис. 6: Рисунок 6 – Что будет, если прервать сервер

Если в терминале прервать исполнение server, то в каталоге /tmp останется файл пайпа fifo, наличие которого мешает вновь запустить сервер. Поэтому его надо удалить руками (рис.6)

Выводы

В ходе лабораторной работы мною были приобретены практические навыки работы с именованными каналами.

Контрольные вопросы

1. В чем ключевое отличие именованных каналов от неименованных?

Именованные каналы (FIFO) существуют в файловой системе и могут использоваться для обмена данными между процессами, как между связанными процессами, так и между несвязанными процессами. Неименованные каналы (pipe) используются только для обмена данными между связанными процессами и существуют только в пределах одного процесса.

2. Возможно ли создание неименованного канала из командной строки?

Создание неименованного канала из командной строки не предусмотрено, т.к. это требует выполнения нескольких системных вызовов и настройки прав доступа.

3. Возможно ли создание именованного канала из командной строки?

Да, создание именованного канала из командной строки возможно (mknod)

4. Опишите функцию языка C, создающую неименованный канал.

Для создания неименованного канала в языке C используется функция `pipe()`. Она принимает в качестве аргумента массив из двух целочисленных дескрипторов файлов: `pipefd[0]` для чтения из канала и `pipefd[1]` для записи в канал. Функция возвращает 0 в случае успешного выполнения и -1 в случае ошибки.

5. Опишите функцию языка C, создающую именованный канал.

Для создания именованного канала в языке C используется функция `mkfifo()`. Она принимает в качестве аргумента путь к создаваемому каналу и права доступа (`mode`). Функция возвращает 0 в случае успешного выполнения и -1 в случае ошибки.

6. Что будет в случае прочтения из `fifo` меньшего числа байтов, чем находится в канале? Большого числа байтов?

Если при чтении из `FIFO` меньше байтов, чем есть в канале, то процесс блокируется до тех пор, пока в канале не появятся новые данные. Если при чтении больше байтов, чем есть в канале, то процесс блокируется до тех пор, пока другой процесс не запишет новые данные в канал.

7. Аналогично, что будет в случае записи в `fifo` меньшего числа байтов, чем позволяет буфер? Большого числа байтов?

Если при записи в `FIFO` меньше байтов, чем позволяет буфер, то данные записываются в буфер частично. Если при записи больше байтов, чем позволяет буфер, то процесс блокируется до тех пор, пока другой процесс не прочтет часть данных из канала и не освободит место в буфере.

8. Могут ли два и более процессов читать или записывать в канал?

Два и более процессов могут читать и записывать в канал. При этом каждый процесс будет иметь свой собственный дескриптор канала, который будет использоваться для чтения или записи данных.

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)?

Функция `write()` используется для записи данных в файловый дескриптор. Она принимает три аргумента: дескриптор файла, указатель на буфер, содержащий данные, и количество байтов, которые нужно записать. Функция возвращает количество записанных байтов в случае успешного выполнения и -1 в случае ошибки.

В программе `server.c` (строка 42) значение 1 в вызове функции `write()` означает, что нужно записать в канал 1 байт из буфера `buffer`.

10. Опишите функцию `strerror`.

`strerror()` используется для получения строки с описанием ошибки, связанной с кодом ошибки `errno`. Она принимает в качестве аргумента код ошибки и возвращает строку с описанием ошибки, если такое описание есть в системе. Если описание ошибки не найдено, функция возвращает строку “Unknown error”.