

Отчет по лабораторной работе №15

Дисциплина: операционные системы

Сергеев Тимофей Сергеевич

Цель работы

Приобретение практических навыков работы с именованными каналами.

Выполнение лабораторной работы

- В программировании именованный канал или именованный конвейер (англ. named pipe) — один из методов межпроцессного взаимодействия, расширение понятия конвейера в Unix и подобных ОС.^[1]
- С помощью труб могут общаться только родственные друг другу процессы, полученные с помощью `fork()`. Именованные каналы FIFO позволяют обмениваться данными с абсолютно "чужим" процессом.^[2]

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?
- Сперва создаю файлы, с которыми буду работать, среди коотрых так же исходный файл из методичики для примера.

Рисунок 1:

```
ts Sergeev@ts Sergeev-VirtualBox:~$ touch common.h
ts Sergeev@ts Sergeev-VirtualBox:~$ emacs common.h
```

Рисунок 2:

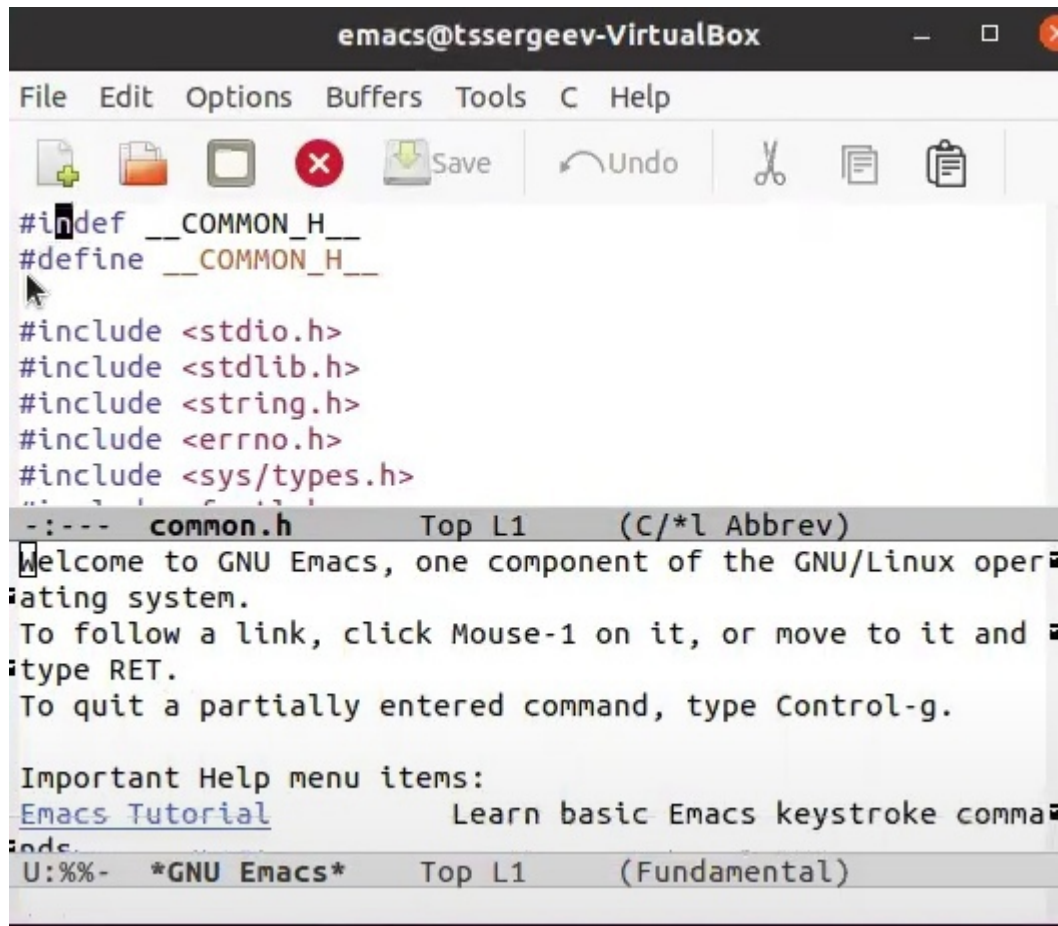
```
ts Sergeev@ts Sergeev-VirtualBox:~$ touch client.c
ts Sergeev@ts Sergeev-VirtualBox:~$ emacs client.c
```

Рисунок 3:

```
ts Sergeev@ts Sergeev-VirtualBox:~$ emacs client.c
ts Sergeev@ts Sergeev-VirtualBox:~$ touch client2.c
```

- Заполняю файл `common.h`

Рисунок 4:



```

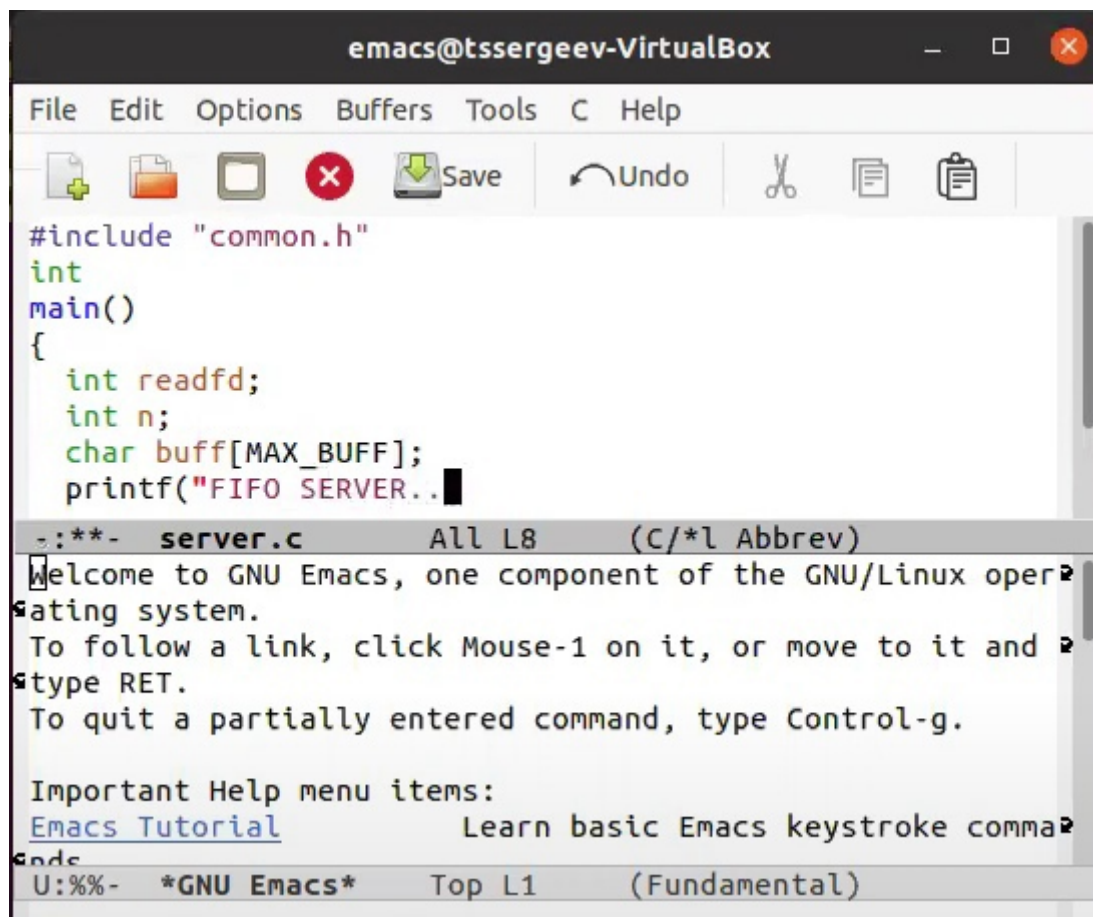
emacs@ts sergeev-VirtualBox
File Edit Options Buffers Tools C Help
[Icons: New, Open, Save, Undo, Cut, Copy, Paste]
#ifndef __COMMON_H__
#define __COMMON_H__
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <errno.h>
#include <sys/types.h>
...
-:--- common.h Top L1 (C/*l Abbrev)
Welcome to GNU Emacs, one component of the GNU/Linux operating system.
To follow a link, click Mouse-1 on it, or move to it and type RET.
To quit a partially entered command, type Control-g.

Important Help menu items:
Emacs Tutorial Learn basic Emacs keystroke commands.
U:%%- *GNU Emacs* Top L1 (Fundamental)

```

- Заполняю файл server.c

Рисунок 5:



```

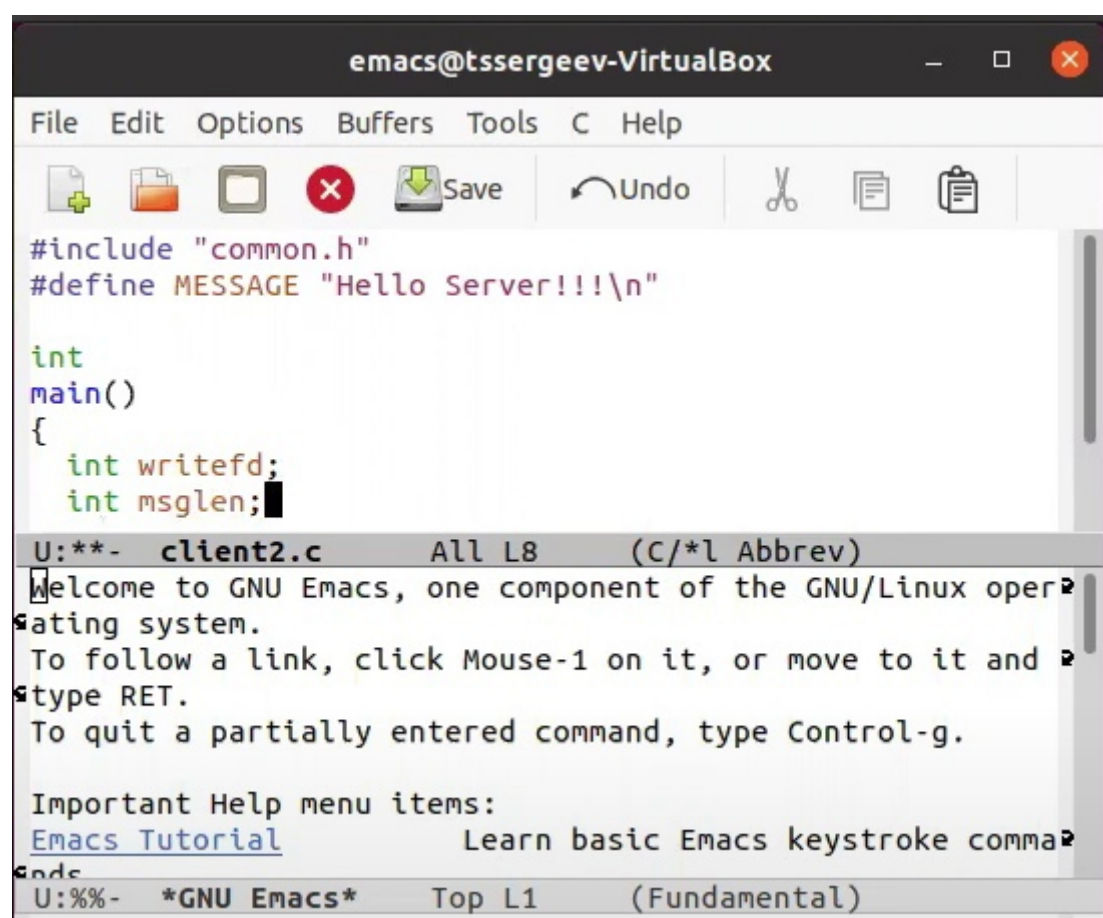
emacs@ts sergeev-VirtualBox
File Edit Options Buffers Tools C Help
[Icons: New, Open, Save, Undo, Cut, Copy, Paste]
#include "common.h"
int
main()
{
    int readfd;
    int n;
    char buff[MAX_BUFF];
    printf("FIFO SERVER..
...
-:**- server.c All L8 (C/*l Abbrev)
Welcome to GNU Emacs, one component of the GNU/Linux operating system.
To follow a link, click Mouse-1 on it, or move to it and type RET.
To quit a partially entered command, type Control-g.

Important Help menu items:
Emacs Tutorial Learn basic Emacs keystroke commands.
U:%%- *GNU Emacs* Top L1 (Fundamental)

```

- Заполняю файл client2.c

Рисунок 6:



- Создаю makefile и заполняю его

Рисунок 7:

```

emacs@ts sergeev-VirtualBox
File Edit Options Buffers Tools Makefile Help
[Icons: New, Open, Save, Undo, Cut, Copy, Paste, Find]
all: server client

server: server.c common.h
    gcc server.c -o server

client: client.c common.h
    gcc server.c -o server

U:**- makefile All L7 (GNUmakefile)
Welcome to GNU Emacs, one component of the GNU/Linux operating system.
To follow a link, click Mouse-1 on it, or move to it and type RET.
To quit a partially entered command, type Control-g.

Important Help menu items:
Emacs Tutorial          Learn basic Emacs keystroke commands
Read the Emacs Manual    View the Emacs manual using Info
\(Non\)Warranty          GNU Emacs comes with ABSOLUTELY NO WARRANTY
Copying Conditions     Conditions for redistributing and changing Emacs
More Manuals / Ordering Manuals How to order printed manuals from the FSF

Useful tasks:
U:%%- *GNU Emacs* Top L1 (Fundamental)

```

- Использую команду make

Рисунок 8:

```

ts sergeev@ts sergeev-VirtualBox:~$ make
gcc server.c -o server
In file included from server.c:1:
common.h:1:2: error: invalid preprocessing directive #indef; did you mean #ifdef?
1 | #indef __COMMON_H__
  | ^~~~~~
  | ifdef
In file included from server.c:1:
common.h:14:2: error: #endif without #if
14 | #endif
    | ^~~~~~
server.c: In function 'main':
server.c:10:7: warning: implicit declaration of function 'mknod' [-Wimplicit-function-declaration]
10 |     if (mknod(FIFO_NAME, S_FIFO | 0666, 0) < 0)
    |         ^~~~~~
server.c:10:23: error: 'S_FIFO' undeclared (first use in this function); did you mean 'S_IFIFO'?
10 |     if (mknod(FIFO_NAME, S_FIFO | 0666, 0) < 0)
    |                         ^~~~~~

```

- Запускаю процесс

Рисунок 9:

```
FIFO Server...
Hello Server!!!
Hello Server!!!
Hello Server!!!
Hello Server!!!
Hello Server!!!
Hello Server!!!
Hello Server!!!
Hello Server!!!
Hello Server!!!
Hello Server!!!
server timeout, 30 - seconds passed
```

Рисунок 10:

```
FIFO Client...
FIFO Client...
FIFO Client...
FIFO Client...
FIFO Client...
FIFO Client...
FIFO Client...
FIFO Client...
FIFO Client...
FIFO Client...
FIFO Client...
client.c: Невозможно открыть FIFO (No such file or directory)
```

Выводы

В ходе выполнения поставленных задач я приобрел простейшие навыки работы с именованными каналами.

Библиография

- [1] https://ru.wikipedia.org/wiki/Именованный_канал#:~:text=В%20программировании%20именованный%20канал%20или,обмениваться%20данными%2C%20даже%20если%20программы

- [2] https://www.opennet.ru/docs/RUS/linux_parallel/node17.html

Контрольные вопросы

1. 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.

2. Создание неименованного канала из командной строки невозможно.

3. Создание именованного канала из командной строки возможно.

```
4. int read(int pipe_fd, void *area, int cnt);
```

```
int write(int pipe_fd, void *area, int cnt);
```

Первый аргумент этих вызовов - дескриптор канала, второй - указатель на область памяти, с которой происходит обмен, третий - количество байт. Оба вызова возвращают число переданных байт (или -1 - при ошибке).

```
5. int mkfifo (const char *pathname, mode_t mode) ;
```

```
mkfifo(FIFO_NAME, 0600) ;
```

Первый параметр — имя файла, идентифицирующего канал, второй параметр маска прав доступа к файлу. Вызов функции `mkfifo()` создаёт файл канала (с именем, заданным макросом `FIFO_NAME`).

6. При чтении меньшего числа байтов, чем находится в канале, возвращается требуемое число байтов, остаток сохраняется для последующих чтений. При чтении большего числа байтов, чем находится в канале или FIFO возвращается доступное число байтов.

7. При записи большего числа байтов, чем это позволяет канал или FIFO, вызов `write(2)` блокируется до освобождения требуемого места. При этом атомарность операции не гарантируется. Если процесс пытается записать данные в канал, не открытый ни одним процессом на чтение, процессу генерируется сигнал. Запись числа байтов, меньшего емкости канала или FIFO, гарантированно атомарно. Это означает, что в случае, когда несколько процессов одновременно записывают в канал, порции данных от этих процессов не перемешиваются.

8. В общем случае возможна много направленная работа процессов с каналом, т.е. возможна ситуация, когда с одним и тем же каналом взаимодействуют два и более процесса, и каждый из взаимодействующих каналов пишет и читает информацию в канал. Но традиционной схемой организации работы с каналом является однонаправленная организация, когда канал связывает два, в большинстве случаев, или несколько взаимодействующих процесса, каждый из которых может либо читать,

либо писать в канал.

9. `Write` - Функция записывает `length` байтов из буфера `buffer` в файл, определенный дескриптором файла `fd`. Эта операция чисто 'двоичная' и без буферизации. Реализуется как непосредственный

вызов DOS. С помощью функции `write` мы посылаем сообщение клиенту или серверу.

10. Строковая функция `strerror` - функция языков C/C++, транслирующая код ошибки, который обычно хранится в глобальной переменной `errno`, в сообщение об ошибке, понятном человеку. Ошибки эти возникают при вызове функций стандартных Си-библиотек. Возвращенный указатель ссылается на статическую строку с ошибкой, которая не должна быть изменена программой. Дальнейшие вызовы функции `strerror` перезапишут содержание этой строки. Интерпретированные сообщения об ошибках могут различаться, это зависит от платформы и компилятора.