

РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

Факультет физико-математических и естественных наук

Кафедра прикладной информатики и теории вероятностей

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ № 15

дисциплина: операционные системы

Студент: Соболевский Денис Андреевич

Группа: НФИбд-02-20

МОСКВА

2021 г.

Цель работы:

Приобретение практических навыков работы с именованными каналами.

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:

- общепользовательские (именованные каналы, сигналы);
- System V Interface Definition (SVID — разделяемая память, очередь сообщений, семафоры)
- 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)`.

```
#include <sys/types.h>
#include <sys/stat.h>
int mkfifo(const char *pathname, mode_t mode);
```

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`):

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

Подробнее с данным типом каталогов можно ознакомиться в статье ["Каналы FIFO"](#)^[1].

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 работы:

Задание 1

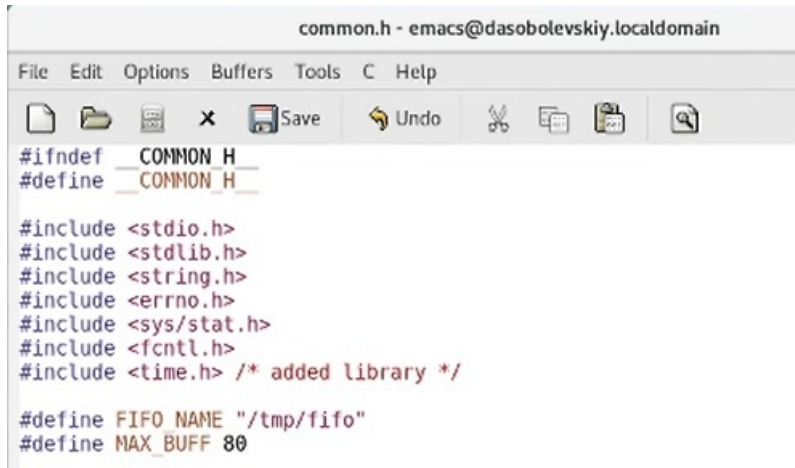
1. Создадим предложенные в лабораторной работе файлы с кодами при помощи текстового редактора `emacs`. Таким образом, создаем файлы `common.h` (рисунк 2), `server.c` (рисунк 3), `client.c` (рисунк 4) и `Makefile` (рисунк 5) с внесенными в них коррективками, как того от нас

требуют задания.

Рисунок 1:

```
[dasobolevskiy@dasobolevskiy ~]$ cd
[dasobolevskiy@dasobolevskiy ~]$ emacs common
[dasobolevskiy@dasobolevskiy ~]$ emacs common.h
[dasobolevskiy@dasobolevskiy ~]$ emacs server.c
[dasobolevskiy@dasobolevskiy ~]$ emacs client.c
[dasobolevskiy@dasobolevskiy ~]$ emacs Makefile
[dasobolevskiy@dasobolevskiy ~]$ gcc -c server.c
[dasobolevskiy@dasobolevskiy ~]$ gcc -o server server.c
[dasobolevskiy@dasobolevskiy ~]$ gcc -c client.c
[dasobolevskiy@dasobolevskiy ~]$ gcc -o client client.c
```

Рисунок 2:

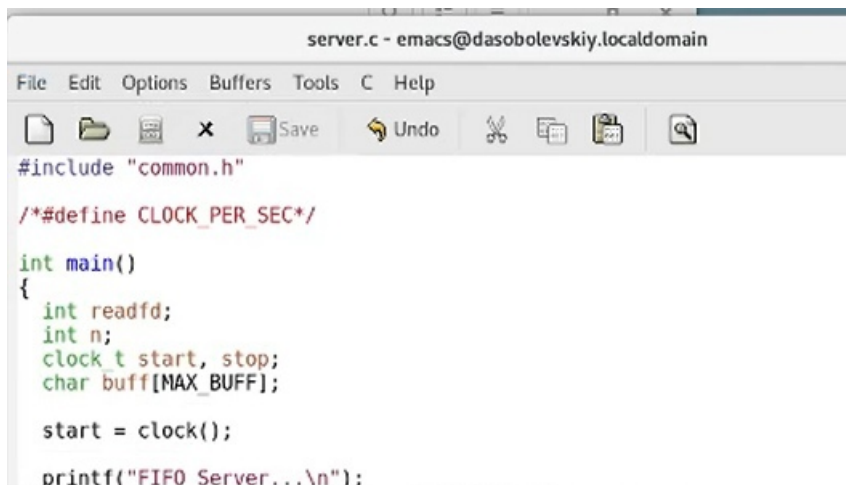


```
common.h - emacs@dasobolevskiy.localdomain
File Edit Options Buffers Tools C Help
#ifndef COMMON_H
#define COMMON_H

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <errno.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <time.h> /* added library */

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

Рисунок 3:



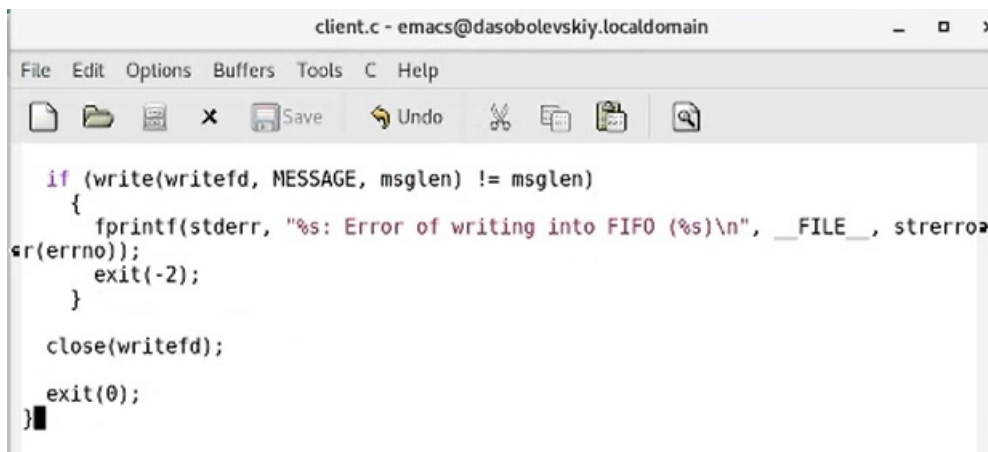
```
server.c - emacs@dasobolevskiy.localdomain
File Edit Options Buffers Tools C Help
#include "common.h"

/*#define CLOCK_PER_SEC*/

int main()
{
    int readfd;
    int n;
    clock_t start, stop;
    char buff[MAX_BUFF];

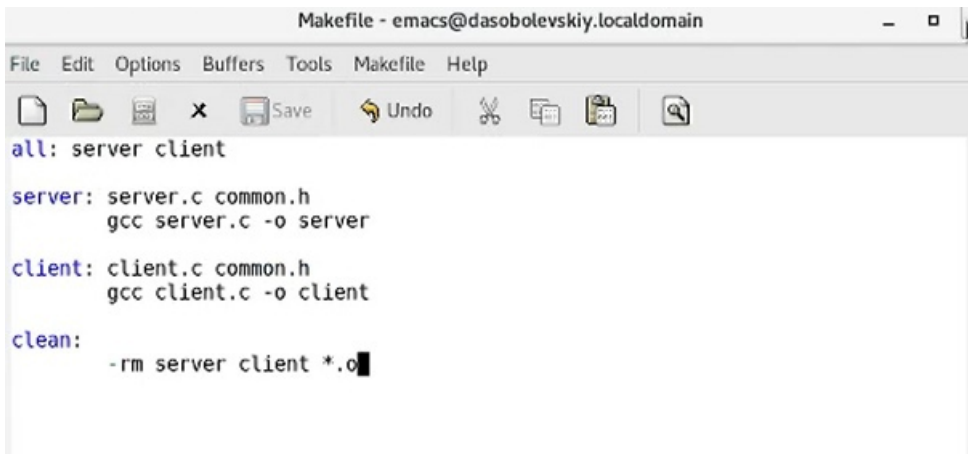
    start = clock();
    printf("FIFO Server...\n");
```

Рисунок 4:



```
client.c - emacs@dasobolevskiy.localdomain
File Edit Options Buffers Tools C Help
if (write(writefd, MESSAGE, msglen) != msglen)
{
    fprintf(stderr, "%s: Error of writing into FIFO (%s)\n", __FILE__, strerror(errno));
    exit(-2);
}
close(writefd);
exit(0);
}
```

Рисунок 5:



```
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
```

Какие коррективы были внесены в первоначальный код:

- в файле `server.c` появилась функция `clock()`, помогающая подсчитывать кол-во времени, затраченное на выполнение алгоритма, подробнее с данной функцией можно ознакомиться статье [Измерение времени выполнения блока кода на C/C++](#)^[2];
- в файле `client.c` вывод текущей даты и времени осуществляем 5 раз с интервалом в 5 секунд - `sleep(5)` ;

Далее компилируем наши программы `server.c` и `client.c` при помощи компилятора `gcc` (*рисунок 1*). Система не выдает нам ошибок, следовательно все реализовано верно.

2. Проверяем работу программ. Откроем три терминала, в одном из них первую очередь запускаем `server.c` , а в оставшихся двух - два `./client.c` .

Видим, что оба клиента выводят дату и время, можно заметить что каждый из них делает это с интервалов в 5 секунд, а интервалы между выводами раздых клиентов равно разнице во времени из запуска, в нашем случае - 3 секунды. Отсюда и итоговое время выполнения работы: 0.33 sec.

Вывод:

Приобрел практические навыки работы с именованными каналами.

Библиография:

[1]: [Каналы FIFO](#)

[2]: [Измерение времени выполнения блока кода на C/C++](#)