Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ Информатика и системы управления
КАФЕДРА Информационная безопасность (ИУ8)

Безопасность Систем Баз Данных

Отчет
по Лабораторной работе №2
“Знакомство с man-pages”

Выполнил:
Евула А. С.,
студент группы ИУ8-63

Проверил:
Зенькович С. А.,
старший преподаватель
кафедры ИУ8

Москва, 2021

ОГЛАВЛЕНИЕ

ЦЕЛЬ РАБОТЫ

Изучение man-pages.

ОСНОВНАЯ ЧАСТЬ

1. MAN-PAGES

это nix идеология "каждый программный продукт содержит документацию/справочную информацию/мануал"

2. MAN

Команда, предназначенная для форматирования и вывода справочных страниц.

Таблица 1- опции команды man

Параметр	Назначение
NAME	имя команды и однострочное описание ее назначения
SYNOPSIS	список опций и аргументов
DESCRIPTION	более подробное описание назначения и принципов работы команды
EXAMPLES	типовые примеры использования
OPTIONS	описания для каждой из опций
EXIT STATUS	значения, возвращаемые при завершении работы
ENVIRTONMENT	список всех переменных среды, затрагивающих программу
FILES	связанные с командой файлы
SEE ALSO	список связанных команд и функций.
HISTORY	история программы
BUGS	вероятные проблемы, известные баги
[OVERVIEW]	обзоры, описания
[DEFAULTS]	параметры по умолчанию (н.п. порядок поиска)

```
man [man options] [[section] page ...] ...
man -k [apropos options] regexp ...
man -K [man options] [section] term ...
man -f [whatis options] page ...
man -l [man options] file ...
man -w|-W [man options] page ...
```

3. CHMOD

Команда, предназначенная для изменения прав доступа к файлам и каталогам.

```
chmod [OPTION]... MODE[,MODE]... FILE...
chmod [OPTION]... OCTAL-MODE FILE...
chmod [OPTION]... --reference=RFILE FILE...
```

Таблица 2- перевод опции MODE

числ.	симв.
0	---
1	--x
2	-w-
3	-wx
4	r--
5	r-x
6	rw-
7	rwx

Таблица 3- варианты опции MODE

r	read	чтение файла
w	write	запись в файл
x	execute	выполнение файла

4. CHOWN

Команда, предназначенная для изменения владельца и/или группу для указанных файлов.
Поля OWNER и GROUP могут быть числовыми и символическими.

```
chown [OPTION]... [OWNER][:[GROUP]] FILE...
chown [OPTION]... --reference=RFILE FILE...
```

5. FSTAB

Конфигурационный файл, содержащий статическую информацию о файловых системах, разделах и устройствах хранения.
Хранится в `/etc/fstab`

Пример содержимого файла:

```
# Static information about the filesystems.
# See fstab(5) for details.

# <file system> <dir> <type> <options> <dump> <pass>
# /dev/sda6
UUID=5ef7bae4-2ca2-4e97-bc32-e5b2c5031f28       /               ext4
rw,relatime     0 1

# /dev/sda1
UUID=458C-49EF          /boot/efi       vfat
rw,relatime,fmask=0022,dmask=0022,codepage=437,iocharset=ascii,shor>
...
```

6. PROC

Псевдо-файловая система, которая используется в качестве интерфейса к структурам данных в ядре, чтобы избежать чтения и записи `/dev/kmem`. Большинство расположенных в ней файлов доступны только для чтения. Такая система позволяет придерживаться концепции - 'все есть файл'.

```
/proc/sys   // e.g. изменение/отображение параметров ядра
```

7. SIGNAL

Асинхронное уведомление процесса о каком-либо событии. Необходим для всзаимодействия между процессами. Когда сигнал послан процессу, операционная система прерывает выполнение процесса, при этом, если процесс установил собственный обработчик сигнала, операционная система запускает этот обработчик.

```
sighandler_t signal(int signum, sighandler_t handler);
```

8. SH

Интерпретатор, который выполняет команды, считанные из командной строки, стандартного ввода или указанного файла.

```
sh clion.sh     // e.g.
```

9. STDIO

Библиотека станадртных потоков ввода/вывода - 3 типа переменных, некоторые макросы и функции для выполнения ввода и вывода.

10. STDIN, STDOUT, STDERR

Стандартные потоки для ввода, вывода и вывода ошибок. По умолчанию стандартный ввод — чтение с клавиатуры, в то время как стандартный вывод и стандартный вывод ошибок печатаются на экране. Эти указатели можно использовать в качестве аргументов для функций.

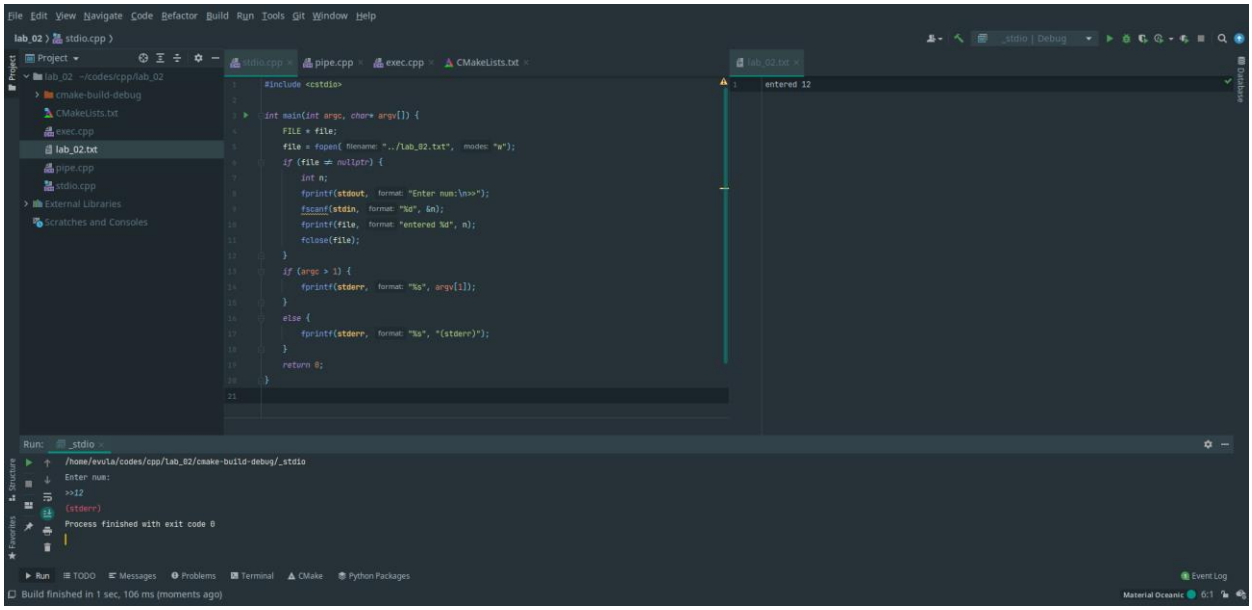

Рисунок 1- Пример взаимодействия с потоками

Исходный код в Приложение А.

11. PIPE

Однонаправленный канал межпроцессного взаимодействия, необходимый для связи нескольких команд путем перенаправления вывода одной команды (`stdout`) на вход (`stdin`) последующей. По сути функция создает два файловых дескриптора: для записи и для чтения. Если закрыть все дескрипторы, ссылающиеся на него, то канал уничтожится.

```
int pipe(int filedes[2]);
```

12. DUP

Метод создает дубликат файлового дескриптора. Возвращает -1 при ошибке. Старый и новый дескрипторы можно использовать друг вместо друга.

```
int dup(int oldfd, [int newfd]);
```

13. FORK

Создает процесс-потомок, который отличается от родительского только идентификатором процесса и идентификатором родительского процесса. При неудаче родительскому процессу возвращается -1

```
pid_t fork(void);
```

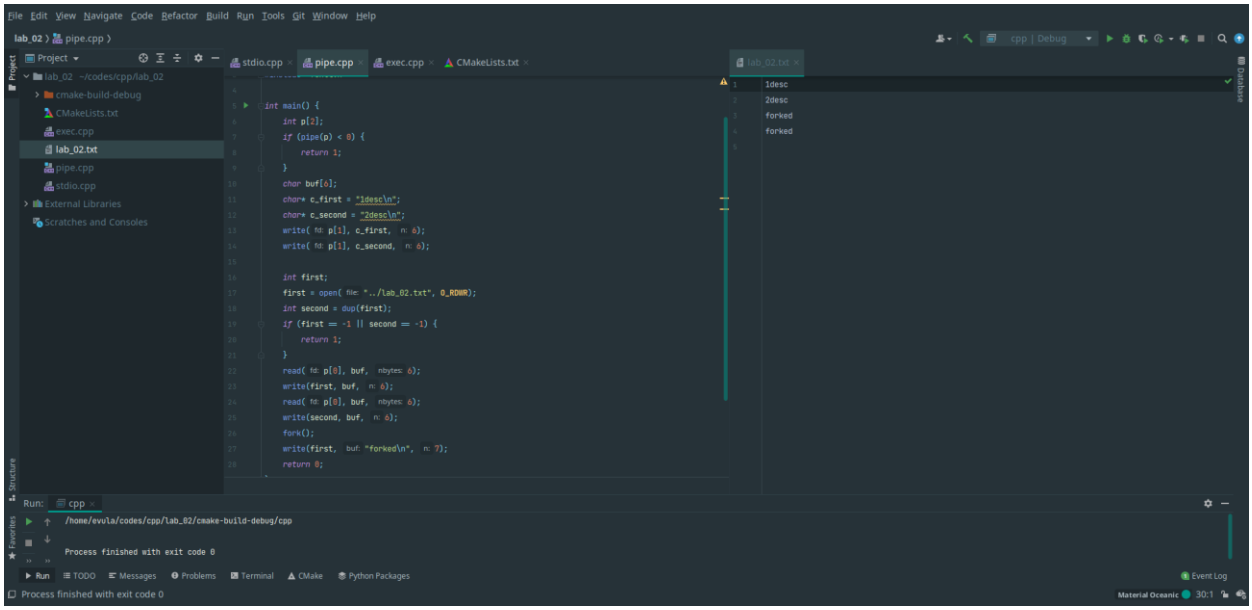

Рисунок 2- Пример использования функций pipe, dup, fork

Исходный код в Приложение Б.

14. EXEC

Заменяет текущий процесс новым, созданым из обычного исполняемого файла. Этот файл является либо исполняемым объектным файлом, либо файлом данных для интерпретатора. Все файлы вызывающей программы остаются открытыми. Они также являются доступными новой программе. Вызов `exec` происходит таким образом, что переданная в качестве аргумента программа загружается в память вместо старой, которая вызвала `exec`.

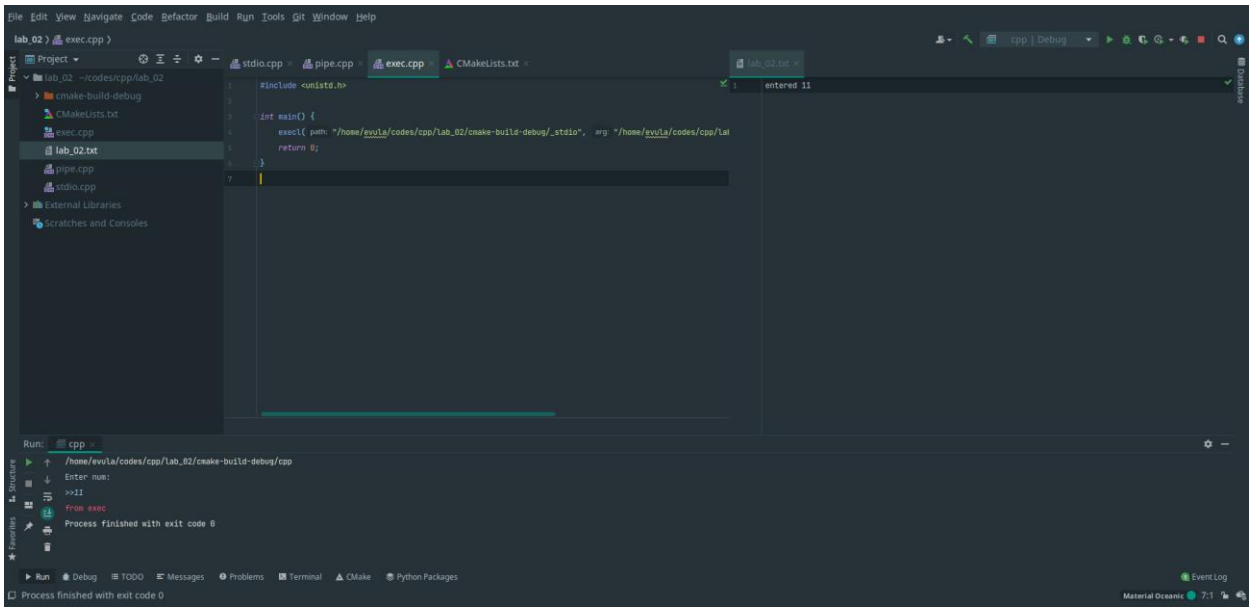

Рисунок 3- Пример использования exec

Исходный код в Приложение В.

ВЫВОДЫ

Были получены знания об идеологии man-pages, некоторых Unix-командах, потоках и нескольких полезных методах в C++.

ПРИЛОЖЕНИЕ А

```
#include <cstdio>

int main(int argc, char* argv[]) {
    FILE * file;
    file = fopen("../lab_02.txt", "w"); // открыть файл для записи
    if (file != nullptr) {
        int n;
        fprintf(stdout, "Enter num:\n>>");
        fscanf(stdin, "%d", &n);    // считать число
        fprintf(file, "entered %d", n); // записать строку в файл
        fclose(file);
    }
    // вывод в stderr
    if (argv) {
        fprintf(stderr, "%s", *argv);
    }
    else {
        fprintf(stderr, "(stderr)");
    }
    return 0;
}
```

ПРИЛОЖЕНИЕ Б

```
#include <cstdio>
#include <unistd.h>
#include <fcntl.h>

int main() {
    int p[2];
    if (pipe(p) < 0) {
        return 1;
    }
    char buf[6];
    char* c_first = "1desc\n";
    char* c_second = "2desc\n";
    write(p[1], c_first, 6);    // передать по каналу
    write(p[1], c_second, 6);

    int first;
    first = open("../lab_02.txt", O_RDWR);  // открыть файл
    int second = dup(first);    // сделать копию дескриптора
    if (first == -1 || second == -1) {
        return 1;
    }
    read(p[0], buf, 6); // чтение из канала
    write(first, buf, 6);   // запись в файл
    read(p[0], buf, 6);
    write(second, buf, 6);
    fork(); // создать 2 процесса (2 раза сделать запись "forked")
    write(first, "forked\n", 7);
    return 0;
}
```

ПРИЛОЖЕНИЕ В

```
#include <unistd.h>

int main() {
    // запустить процесс из первого примера, в качестве аргументов передается
строка
    execl("/home/evula/codes/cpp/lab_02/cmake-build-debug/_stdio", "from
exec", NULL);
    return 0;
}
```