Учреждение образования

«Белорусский Государственные университет

информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

Кафедра электронных вычислительных машин

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовому проекту

«Многопоточный FTP-клиент»

по дисциплине

«Системное программное обеспечение вычислительных машин»

Выполнил: Руководитель:

студент гр. 850504

Линкевич Владислав Николаевич Глоба А. А.

Минск, 2020

Содержание

[ВВЕДЕНИЕ 3](#_Toc27321280)

[1. ОБЗОР ИСТОЧНИКОВ 4](#_Toc27321281)

[1.1 Основные термины 5](#_Toc27321283)

[1.2 Выбор технологий для создания проекта 5](#_Toc27321284)

[2. СТРУКТУРНОЕ ПРОЕКТИРОВАНИE 5](#_Toc27321282)

[2.1 Общее структурное описание состава программного обеспечения 5](#_Toc27321283)

[2.2 Описание деления проекта на отдельные программы, модули или другие составные части 5](#_Toc27321284)

[2.3 Краткое описание сторонних программных компонент 5](#_Toc27321285)

[3. ФУНКЦИОНАЛЬНОЕ ПРОЕКТИРОВАНИЕ 6](#_Toc27321286)

[3.1 Общее описание функционирования программного обеспечения по обработке данных и получению результатов 6](#_Toc27321287)

[4. РАЗРАБОТКА ПРОГРАММНЫХ МОДУЛЕЙ 13](#_Toc27321289)

[5. РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ 13](#_Toc27321289)

[5.1 Минимальные программно-аппаратные требования к установке и запуску программы 13](#_Toc27321290)

[5.2 Краткое описание основных действий пользователя при использовании программы 13](#_Toc27321291)

[6 ТЕСТИРОВАНИЕ 16](#_Toc27321292)

[ЗАКЛЮЧЕНИЕ 18](#_Toc27321293)

[СПИСОК ИСТОЧНИКОВ 19](#_Toc27321294)

[ПРИЛОЖЕНИЯ 20](#_Toc27321295)

# ВВЕДЕНИЕ

Когда интернет только зарождался, но уже появились компьютерные сети, очень остро встал вопрос о потребности передачи файлов от одного компьютера к другому. В 1971 году каналы передачи данных не были такими надежными и такими быстрыми как сейчас, поэтому необходим был инструмент, который поможет обмениваться документами друг с другом на расстоянии. От этого инструмента требовались: простота использования и надежность при отправке и получении. Как вы уже наверное могли догадаться из названия курсового проекта — таким инструментом стал FTP-протокол.

FTP расшифровывается как File Transfer Protocol — протокол передачи файлов. Он отличается от других протоколов тем, что если в процессе передачи происходит какая-либо ошибка, то процесс останавливается и выводит сообщение для пользователя. Если же сообщений не было, то передача произошла успешно. Популярность FTP сохраняется и сегодня, благодаря широкой поддержке со стороны провайдеров и простоте использования.

Чаще всего пользователяем требуются следующие функции: быстрая пакетная загрузка файлов на сервер, быстрое скачивание, управление файлами и папками на хостинге.

По FTP-протоколу можно скачивать все что угодно: аудио- и видео- файлы, документы, изображения любых форматов и т. д. .

# ОБЗОР ИСТОЧНИКОВ

## Основные термины

1.1.1 FTP ([англ.](https://ru.wikipedia.org/wiki/Английский_язык) *File* *Transfer* *Protocol*) — [протокол передачи](https://ru.wikipedia.org/wiki/Протоколы_передачи_данных) [файлов](https://ru.wikipedia.org/wiki/Файл) по [сети](https://ru.wikipedia.org/wiki/Вычислительная_сеть), является одним из старейших прикладных протоколов, появившихся задолго до [HTTP](https://ru.wikipedia.org/wiki/HTTP), и даже до [TCP/IP](https://ru.wikipedia.org/wiki/TCP/IP), в 1971 году; в первое время он работал поверх протокола [NCP](https://ru.wikipedia.org/wiki/Network_Control_Protocol). Он и сегодня широко используется для распространения [ПО](https://ru.wikipedia.org/wiki/Программное_обеспечение) и доступа к удалённым [хостам](https://ru.wikipedia.org/wiki/Хост). В отличие от [TFTP](https://ru.wikipedia.org/wiki/Trivial_File_Transfer_Protocol), гарантирует передачу (либо выдачу ошибки) за счёт применения [квотиру-емого](https://ru.wikipedia.org/wiki/Квотирование) протокола.

Протокол построен на архитектуре «[клиент-сервер](https://ru.wikipedia.org/wiki/Клиент-сервер)» и использует разные сетевые соединения для передачи команд и данных между клиентом и сервером. Пользователи FTP могут пройти аутентификацию, передавая логин и пароль [открытым текстом](https://ru.wikipedia.org/wiki/Открытый_текст), или же, если это разрешено на сервере, они могут подключиться анонимно. Можно использовать протокол [SSH](https://ru.wikipedia.org/wiki/SSH) для безопасной передачи, скрывающей (шифрующей) логин и пароль, а также шифрующей содержимое.[1]

1.1.2 **«**Клиент — сервер**»** ([англ.](https://ru.wikipedia.org/wiki/Английский_язык) client–server) — вычислительная или сетевая архитектура, в которой задания или сетевая нагрузка распределены между поставщиками услуг, называемыми серверами, и заказчиками услуг, называемыми клиентами. Фактически [клиент](https://ru.wikipedia.org/wiki/Клиент_(информатика)) и [сервер](https://ru.wikipedia.org/wiki/Сервер_(программное_обеспечение)) — это [программное обеспечение](https://ru.wikipedia.org/wiki/Программное_обеспечение). Обычно эти программы расположены на разных [вычислительных машинах](https://ru.wikipedia.org/wiki/Вычислительная_машина) и взаимодействуют между собой через [вычислительную сеть](https://ru.wikipedia.org/wiki/Вычислительная_сеть) посредством

[сетевых протоколов](https://ru.wikipedia.org/wiki/Сетевой_протокол), но они могут быть расположены также и на одной машине. Программы-серверы ожидают от клиентских программ запросы и предоставляют им свои ресурсы в виде данных (например, [загрузка файлов](https://ru.wikipedia.org/wiki/Загрузка_файлов) посредством [HTTP](https://ru.wikipedia.org/wiki/HTTP), [FTP](https://ru.wikipedia.org/wiki/FTP), [BitTorrent](https://ru.wikipedia.org/wiki/BitTorrent_(протокол)), [потоковое мультимедиа](https://ru.wikipedia.org/wiki/Потоковое_мультимедиа) или работа с [базами данных](https://ru.wikipedia.org/wiki/База_данных)) или в виде сервисных функций (например, работа с  электронной почтой, общение посредством систем мгновенного обмена сообщениями или просмотр [web-страниц](https://ru.wikipedia.org/wiki/Web-страница) во [всемирной паутине](https://ru.wikipedia.org/wiki/Всемирная_паутина)). Поскольку  одна программа-сервер может выполнять запросы от множества программ-клиентов, её размещают на специально выделенной вычислительной машине, настроенной особым образом, как правило, совместно с другими программами-серверами, поэтому производительность этой машины должна быть высокой. Из-за особой роли такой машины в сети, специфики её оборудования и программного обеспечения, её также называют [сервером](https://ru.wikipedia.org/wiki/Сервер_(аппаратное_обеспечение)), а машины, выполняющие клиентские программы, соответственно, клиентами.[2]

1.1.3 Со́кет ([англ.](https://ru.wikipedia.org/wiki/Английский_язык) *socket* — разъём) — название [программного интер-фейса](https://ru.wikipedia.org/wiki/API) для обеспечения обмена данными между [процессами](https://ru.wikipedia.org/wiki/Процесс_(информатика)). Процессы при таком обмене могут исполняться как на одной [ЭВМ](https://ru.wikipedia.org/wiki/ЭВМ), так и на различныхЭВМ, связанных между собой [сетью](https://ru.wikipedia.org/wiki/Компьютерная_сеть). Сокет — [абстрактный](https://ru.wikipedia.org/wiki/Абстракция) объект, представляющий конечную точку соеди-нения.

Следует различать [клиентские](https://ru.wikipedia.org/wiki/Клиент_(информатика)) и [серверные](https://ru.wikipedia.org/wiki/Сервер_(аппаратное_обеспечение)) сокеты. Клиентские соке-ты грубо можно сравнить с конечными аппаратами [телефонной сети](https://ru.wikipedia.org/wiki/Телефонная_сеть_общего_пользования), а серверные — с [коммутаторами](https://ru.wikipedia.org/wiki/Телефонный_коммутатор). Клиентское приложение (например, [браузер](https://ru.wikipedia.org/wiki/Браузер)) использует только клиентские сокеты, а серверное (например, [веб-сервер](https://ru.wikipedia.org/wiki/Веб-сервер), которому браузер посылает запросы) — как клиентские, так и серверные сокеты.[3]

1.1.4 Хост (от [англ.](https://ru.wikipedia.org/wiki/Английский_язык) host —«хозяин, принимающий гостей») — любое [устройство](https://ru.wikipedia.org/wiki/Устройство), предоставляющее сервисы формата «[клиент-сервер](https://ru.wikipedia.org/wiki/Клиент-сервер)» в режиме сервера по каким-либо [интерфейсам](https://ru.wikipedia.org/wiki/Интерфейс) и уникально определённое на этих интерфейсах. В более частном случае под хостом могут понимать любой [компьютер](https://ru.wikipedia.org/wiki/Компьютер), [сервер](https://ru.wikipedia.org/wiki/Сервер_(аппаратное_обеспечение)), подключённыйк [локальной](https://ru.wikipedia.org/wiki/Локальная_сеть) или [глобальной](https://ru.wikipedia.org/wiki/Глобальная_сеть)

[сети](https://ru.wikipedia.org/wiki/Компьютерная_сеть).[4].

1.1.5 Порт ([англ.](https://ru.wikipedia.org/wiki/Английский_язык) port) — целое неотрицательное число, записываемое в заголовках [протоколов](https://ru.wikipedia.org/wiki/Протокол_передачи_данных) [транспортного ровня](https://ru.wikipedia.org/wiki/Транспортный_уровень) [модели OSI](https://ru.wikipedia.org/wiki/Сетевая_модель_OSI) ([TCP](https://ru.wikipedia.org/wiki/TCP), [UDP](https://ru.wikipedia.org/wiki/UDP), [SCTP](https://ru.wikipedia.org/wiki/SCTP), [DCCP](https://ru.wikipedia.org/wiki/DCCP)). Используется для определения [процесса](https://ru.wikipedia.org/wiki/Процесс_(информатика))-получателя пакета в пределах одного [хоста](https://ru.wikipedia.org/wiki/Хост).[5].

1.1.6 Пото́к выполне́ния (тред; от [англ.](https://ru.wikipedia.org/wiki/Английский_язык) thread — нить) — наименьшая единица обработки, исполнение которой может быть [назначено](https://ru.wikipedia.org/wiki/Диспетчер_операционной_системы) [ядром операционной системы](https://ru.wikipedia.org/wiki/Ядро_операционной_системы). Реализация потоков выполнения и [процессов](https://ru.wikipedia.org/wiki/Процесс_(информатика)) в разных операционных системах отличается друг от друга, но в большинстве случаев поток выполнения находится внутри процесса. Несколько потоков выполнения могут существовать в рамках одного и того же процесса и совместно использовать ресурсы, такие как [память](https://ru.wikipedia.org/wiki/Разделяемая_память), тогда как процессы не разделяют этих ресурсов. В частности, потоки выполнения разделяют инструкции процесса (его код) и его контекст (значения переменных, которые они имеют в любой момент времени). В качестве аналогии потоки выполнения процесса можно уподобить нескольким вместе работающим поварам. Все они готовят одно блюдо, читают одну и ту же кулинарную книгу с одним и тем же рецептом и следуют его указаниям, причём не обязательно все они читают на одной и той же страниц.[6].

1.1.7 Многопото́чность (англ. *Multithreading*) — свойство платформы (например, [операционной системы](https://ru.wikipedia.org/wiki/Операционная_система), [виртуальной машины](https://ru.wikipedia.org/wiki/Виртуальная_машина) и т. д.) или [приложения](https://ru.wikipedia.org/wiki/Прикладное_программное_обеспечение), состоящее в том, что [процесс](https://ru.wikipedia.org/wiki/Процесс_(информатика)), порождённый в операционной системе, может состоять из нескольких [*потоков*](https://ru.wikipedia.org/wiki/Поток_выполнения), выполняющихся «[параллельно](https://ru.wikipedia.org/wiki/Параллельные_вычисления)», то есть без предписанного порядка во [времени](https://ru.wikipedia.org/wiki/Время). При выполнении некоторых задач такое разделение может достичь более эффективного использования [ресурсов вычислительной машины](https://ru.wikipedia.org/w/index.php?title=Вычислительные_ресурсы&action=edit&redlink=1). [7].

## Выбор технологий для создания проекта

Для разработки была выбрана IDE CLion 2020.1.1 – полнофункцональ-ная интегрированная среда разработки для написания, отладки, тестирова-ния и развертывания кода на любой платформе, разработанная компанией JetBrains. Она была использована в виду следующего: возможность напи-

сания и компиляции красивого и правильного кода на C/C++, наличие ин-

струментов для удобной отладки, интеграция с популярными системами контроля версий (Subversion, Git, GitHub, Mercurial, CVS и Perforce), фреймворками для юнит-тестирования Google Test, Boost.Test и Catch, поддерживает систему документирования Doxygen и инструменты для работы с базами данных [8].

Программа работает в виде консольного приложения, используя тер-минал операционной системы Linux, предназначенный для выполнения текстовых команд. Основная информация о подключении к серверу, а именно: IP-адрес хоста и порт передаются в качестве агрументов команд-ной строки. Выполнение команд происходит при удачной попытке под-ключения к серверу. Команды состоят из букв, цифр, символов, набирают-ся построчно, выполняются после нажатия клавиши Enter. Основной ин-струмент здесь клавиатура. Интерфейс командной строки встроен в ядро системы, он будет доступен, даже если графический интерфейс не запус-тится.

Преимущества: небольшой расход ресурсов, гибкость при составлении перечня действий из команд, возможность автоматического выполнения команд, возможность копировать и вставлять команды, выполнение с уда-лённого устройства.

# СТРУКТУРНОЕ ПРОЕКТИРОВАНИE

## Общее структурное описание состава программного обеспечения

Структурная схема изображена на рисунке 1.

Данная программа имеет 3 модуля: выбор раздела, выбор файловой системы ext4 (ext2, ext3, NTFS), процесс форматирования.

В модуле «выбор раздела» осуществляется выбор раздела твердотельного накопителя или жёсткого диска для его форматирования, переход к модулю «выбор файловой системы».

В модуле «выбор файловой системы» пользователю предоставляется возможность выбрать одну из перечисленных файловых систем (ext4, ext2, ext3, NTFS) для дальнейшего форматирования, осуществляется переход к последнему модулю.

В модуле «процесс форматирования» осуществляется форматирование раздела жёсткого диска или твердотельного накопителя в одну из выбранных файловых систем.

## Описание деления проекта на отдельные программы, модули или другие составные части

Модуль «выбор раздела» включает в себя список приведённых разделов жёсткого диска, твердотельного накопителя, данного персонального компьютера, на которых мы можем изменить файловую систему. Существует возможность отформатировать компьютер, на котором запущена данная программа. Перемещение по списку в консоли осуществляется треугольнообразным указателем по левой стороне списка, стрелочками вниз/вверх на клавиатуре. Выбор происходит по нажатии на Enter. Он в свою очередь вызывают переход в модуль «выбор файловой системы».

Модуль «выбор файловой системы» включает в себя список вариантов файловых систем, на которые имеется возможность изменить текущую файловую систему выбранного раздела жёсткого диска или твердотельного накопителя. Передвижение по элементам списка осуществляется с помощью треугольнообразного указателя по левой стороне списка, а выбор той или иной файловой системы по нажатию на Enter. Варианты файловых систем, которые приведены в списке: ext2, ext3, ext4, NTFS. После выбора файловой системы происходит переход к заключительному модулю программы.

Модуль «процесс форматирования» представляет собой форматирование раздела жёсткого диска или твердотельного накопителя в одну из выбранных файловых систем. Процесс отображается в консоли процентным состоянием и оповещает о завершении процедуры форматирования.

* 1. Краткое описание сторонних программных компонент

При написании данной программы использовался пакет Gparted в частности команда mkfs. Gparted является редактором разделов жёстких дисков, твердотельных накопителей, т.е. графическим пользовательским интерфейсом Gtk2 для других инструментальных средств командной строки, с помощью которых можно создавать, удалять или изменять разделы на жёстких дисках или твердотельных накопителях. [Команда](https://ru.qwe.wiki/wiki/Command_(computing)) mkfs используется для  [форматирования](https://ru.qwe.wiki/wiki/Disk_formatting) блока устройства хранения с конкретной [файловой системой](https://ru.qwe.wiki/wiki/File_system) . Команда является частью [Unix](https://ru.qwe.wiki/wiki/Unix) и [Unix-подобных](https://ru.qwe.wiki/wiki/Unix-like)[операционных систем](https://ru.qwe.wiki/wiki/Operating_system) . В Unix, блок [запоминающего устройства](https://ru.qwe.wiki/wiki/Block_(data_storage))  должен быть отформатирован, прежде чем он может быть [установлен](https://ru.qwe.wiki/wiki/Mount_(computing)) и доступен через [иерархии файловой системы](https://ru.qwe.wiki/wiki/Unix_directory_structure).

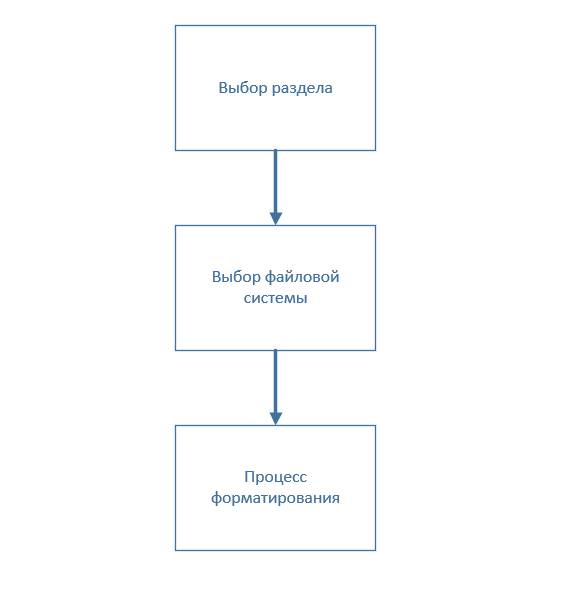


Рисунок 1 – структурная схема

# ФУНКЦИОНАЛЬНОЕ ПРОЕКТИРОВАНИЕ

## Общее описание функционирования программного обеспечения по обработке данных и получению результатов

Диаграмма классов приведена в приложении 1.

Класс DeviceChooser используется для выбора устройства (может быть жёсткий диск или твердотельный накопитель), на котором будет происходить форматирование раздела в ext4. Он включает в себя вектор со всеми устройствами и список выбранных устройств. Мы пытаемся открыть папку с устройствами, если успешно, то записываем в вектор allDevices все устройства, в противном случае возвращаем код ошибки DEVICE\_DIR\_NOT\_FOUND\_ERROR. В данном классе имеется функция addToFiltered(**const** string &exp), в котором происходит добавление всех устройств, соответствующих шаблону, приведённому в параметрах функции, в список filteredDevices, а также метод getFiltered(), с помощью которого мы имеем возможность получить список выбранных устройств.

#ifndef FORMATTER\_DEVICECHOOSER\_H

#define FORMATTER\_DEVICECHOOSER\_H

#include <vector>

#include <**set**>

#include <**string**>

#include <dirent.h>

#include "constants.h"

using std::vector;

using std::**string**;

using std::**set**;

**class** DeviceChooser {

vector<**string**> allDevices;

**set**<**string**> filteredDevices;

**public**:

DeviceChooser();

**void** addToFiltered(**const** **string** &exp);

vector<**string**> getFiltered();

};

**#include** **"**DeviceChooser.h**"**

DeviceChooser::DeviceChooser() {

allDevices = vector<string>();

filteredDevices = set<string>();

DIR \*dir = opendir("/dev/");

**if** (dir) {

dirent \*device;

**while** ((device = readdir(dir)) != **nullptr**) {

allDevices.emplace\_back(device->d\_name);

}

} **else** {

exit(DEVICE\_DIR\_NOT\_FOUND\_ERROR);

}

}

**void** DeviceChooser::addToFiltered(**const** string &exp) {

**for** (**const** **auto** &device : allDevices) {

**if** (device.find(exp) != string::npos) {

filteredDevices.insert(device);

}

}

}

vector<string> DeviceChooser::getFiltered() {

**return** vector<string>(filteredDevices.begin(), filteredDevices.end());

}

Класс ItemsSelector отвечает за выбор элемента. Имеет переменную **string** lastSelection, которая создана для хранения последнего выбранного элемента. Класс содержит функцию select(**const** vector<string> &data), которая из вектора приведённых данных позволяет выбрать нужный пользователю элемент, запоминает выбранный элемент в переменную lastSelection и возвращает его. Также имеется функция getLastSelection(), которая возвращает значение переменной lastSelection.

#ifndef FORMATTER\_ITEMSSELECTOR\_H

#define FORMATTER\_ITEMSSELECTOR\_H

#include <**string**>

#include <vector>

using std::vector;

using std::**string**;

**class** ItemsSelector {

**string** lastSelection;

**public**:

**const** **string** &getLastSelection();

**string** select(**const** vector<**string**> &data);

};

**#include** **<**ncurses.h**>**

**#include** **"**ItemsSelector.h**"**

**const** string &ItemsSelector::getLastSelection() {

**return** lastSelection;

}

string ItemsSelector::select(**const** vector<string> &data) {

**int** choice = 0;

initscr();

curs\_set(0);*//убиваем прямоугольничек с курсором*

keypad(stdscr, TRUE);

**for** (**auto** &item : data) *//Проходим по всем элементам меню*

{

printw(" %s\n\r", item.c\_str());

}

**bool** moving = **true**;

**while** (moving) {

**for** (**int** i = 0; i < data.size(); ++i) {

**if** (i == choice) {

mvaddch(i, 0, L'>');

} **else** {

mvaddch(i, 0, ' ');

}

}

*//Получаем нажатие пользователя*

**int** key = getch();

**switch** (key) {

**case** KEY\_UP:

**if** (choice) *//Если возможно, переводим указатель вверх*

choice--;

**break**;

**case** KEY\_DOWN:

**if** (choice < data.size() - 1) *//Если возможно, переводим указатель вниз*

choice++;

**break**;

**case** 10:

moving = **false**;

**break**;

**default**:

printw("%d\r\n", key);

**break**;

}

}

clear();

endwin();

lastSelection = data[choice];

**return** lastSelection;

}

Класс Ext4Formatter является наследником абстрактного класса Formatter и переопределяет виртуальную функцию format (**string** device). Данная функция принимает раздел устройства и форматирует его. Класс создан для работы с файловой системой ext4.

#ifndef FORMATTER\_EXT4FORMATTER\_H

#define FORMATTER\_EXT4FORMATTER\_H

#include "formatter.h"

**class** Ext4Formatter : **public** Formatter {

**public**:

**void** format(**string** device) override;

};

**#include** **<**iostream**>**

**#include** **"**Ext4Formatter.h**"**

**void** Ext4Formatter::format(string device) {

string command = "sudo mkfs -t ext4 /dev/" + device;

system(command.c\_str());

}

Класс Ext3Formatter является наследником абстрактного класса Formatter и переопределяет виртуальную функцию format (**string** device). Данная функция принимает раздел устройства и форматирует его. Класс создан для работы с файловой системой ext3.

#ifndef FORMATTER\_EXT4FORMATTER\_H

#define FORMATTER\_EXT4FORMATTER\_H

#include "formatter.h"

**class** Ext3Formatter : **public** Formatter {

**public**:

**void** format(**string** device) override;

};

**#include** **<**iostream**>**

**#include** **"**Ext3Formatter.h**"**

**void** Ext3Formatter::format(string device) {

string command = "sudo mkfs -t ext3 /dev/" + device;

system(command.c\_str());

}

Класс NTFSFormatter является наследником абстрактного класса Formatter и переопределяет виртуальную функцию format (**string** device). Данная функция принимает раздел устройства и форматирует его. Класс создан для работы с файловой системой NTFS.

#ifndef FORMATTER\_EXT4FORMATTER\_H

#define FORMATTER\_EXT4FORMATTER\_H

#include "formatter.h"

**class** NTFSFormatter : **public** Formatter {

**public**:

**void** format(**string** device) override;

};

**#include** **<**iostream**>**

**#include** **"**NTFSFormatter.h**"**

**void** Ext3Formatter::format(string device) {

string command = "sudo mkfs -t NTFS /dev/" + device;

system(command.c\_str());

}

Класс Fat32Formatter является наследником абстрактного класса Formatter и переопределяет виртуальную функцию format (**string** device). Данная функция принимает раздел устройства и форматирует его. Класс создан для работы с файловой системой fat32.

#ifndef FORMATTER\_EXT4FORMATTER\_H

#define FORMATTER\_EXT4FORMATTER\_H

#include "formatter.h"

**class** Fat32Formatter : **public** Formatter {

**public**:

**void** format(**string** device) override;

};

**#include** **<**iostream**>**

**#include** **"**Fat32Formatter.h**"**

**void** Fat32Formatter::format(string device) {

string command = "sudo mkfs -t fat32 /dev/" + device;

system(command.c\_str());

}

Класс Formatter является абстрактным классом и содержит чисто виртуальную функцию virtual **void** format (**string** device) = 0. Данный класс создан для облегчения работы с различными файловыми системами.

#ifndef FORMATTER\_FORMATTER\_H

#define FORMATTER\_FORMATTER\_H

#include <**string**>

using std::**string**;

**class** Formatter {

**public**:

virtual **void** format(**string** device) = 0;

};

1. [РАЗРАБОТКА ПРОГРАММНЫХ МОДУЛЕЙ](#_Toc27321289)

Схема алгоритма в приложении 3

В качестве примера можно привести алгоритм движения треугольнообразного указателя по спискам устройств или разделов, а также по списку файловых систем:

1. Создаём переменные moving и choice, присваиваем им значения true и 0 соответственно. Мoving переменная создаётся для дальнейшего выхода из цикла, а choice для движения по элементам списка.
2. Заходим в цикл while, попадаем сразу же в цикл for по всем элементам списка. Проверяем, если choice равняется индексу строки в списке, то отрисовываем треугольнообразный указатель, если нет, то отрисовываем пробел.
3. Получаем нажатие пользователя и заходим в switch по нажатию: если же код нажатия равен коду стрелочки вверх на клавиатуре, то проверяем, возможно ли передвинуть choice вверх(т.е. сравниваем его с 0), если возможно, уменьшаем и выходим из switch, если же код нажатия равен коду стрелочки вниз на клавиатуре, то проверяем, возможно ли передвинуть choice вниз (т.е. сравниваем его с максимальным индексом списка), если возможно, увеличиваем и выходим из switch, если же код нажатия равен 10 – код enter на клавиатуре, то присваиваем значению переменной moving false и выходим из switch и while.

**int** choice = 0;

**bool** moving = **true**;

**while** (moving) {

**for** (**int** i = 0; i < data.size(); ++i) {

**if** (i == choice) {

mvaddch(i, 0, L'>');

} **else** {

mvaddch(i, 0, ' ');

}

}

**int** key = getch();

**switch** (key) {

**case** KEY\_UP:

**if** (choice)

choice--;

**break**;

**case** KEY\_DOWN:

**if** (choice < data.size() - 1)

choice++;

**break**;

**case** 10:

moving = **false**;

**break**;

}

}

# РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ

## Минимальные программно-аппаратные требования к установке и запуску программы

Для запуска и успешного взаимодействия с программой необходим персональный компьютер с установленной операционной системой Linux, компьютерная мышь.

Для успешного запуска необходимо наличие всех классов в проекте, а также загруженный пакет mkfs. Запуск осуществляется консольной командой ./formatter.

## 

## 5.2 Краткое описание основных действий пользователя при использовании программы

При запуске программы появляется список устройств и разделов, разделы которых можно форматировать, и треугольнообразный указатель с левой стороны, с помощью которого пользователь может перемещаться по данному списку. Выбор осуществляется по нажатию на Enter.

Описание модулей:

1. Модуль «Выбор раздела»

Вид модуля предоставлен на рисунке 5.1.

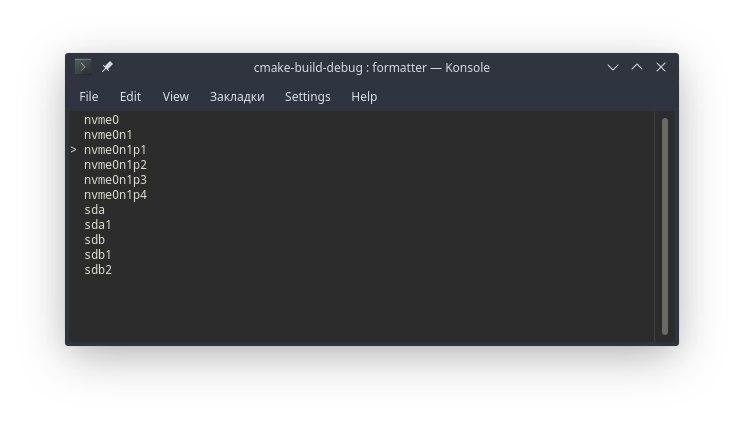


Рисунок 5.1 – модуль «выбор раздела»

При выборе раздела или устройства пользователь переходит к следующему модулю. Однако стоит заметить, что очень важно проверить накопитель, который пользователь хочет отформатировать, ведь среди вышеперечисленных вариантов есть и операционная система компьютера.

1. Модуль «Выбор файловой системы»

При переходе к данному модулю программы появляется список поддерживаемых файловых систем, и треугольнообразный указатель с левой стороны, с помощью которого пользователь может перемещаться по данному списку. Выбор осуществляется по нажатию на Enter. После выбора осуществляется переход к 3 модулю программы.

Вид модуля представлен на рисунке 5.2.

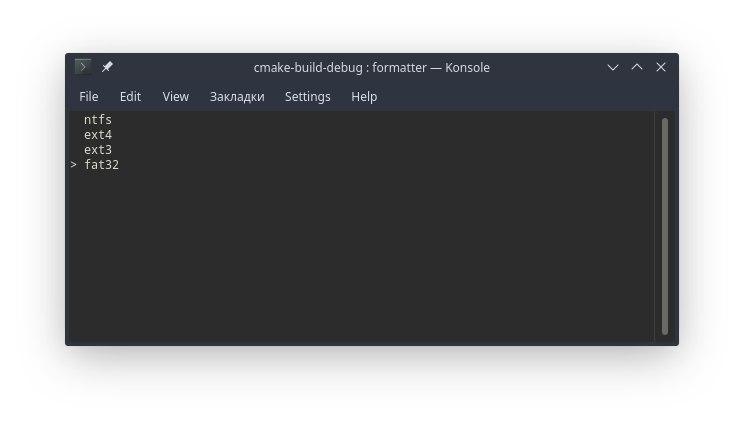


Рисунок 5.2 – модуль «выбор файловой системы»

1. Модуль «Процесс форматирования»

Вид модуля представлен на рисунках 5.3, 5.4.

В данном модуле происходит основная и главная часть всей программы-форматирование выбранного устройства или его раздела в выбранную файловую систему. Данный процесс сопровождается словесным описанием, а по готовности сообщает об этом пользователю. На 1 этапе требуется ввести пароль безопасности (т.е. для корректной работы программы необходимо иметь права доступа администратора). На втором этапе выводится информация о завершении процесса форматирования или же происходит ожидание окончания процесса.

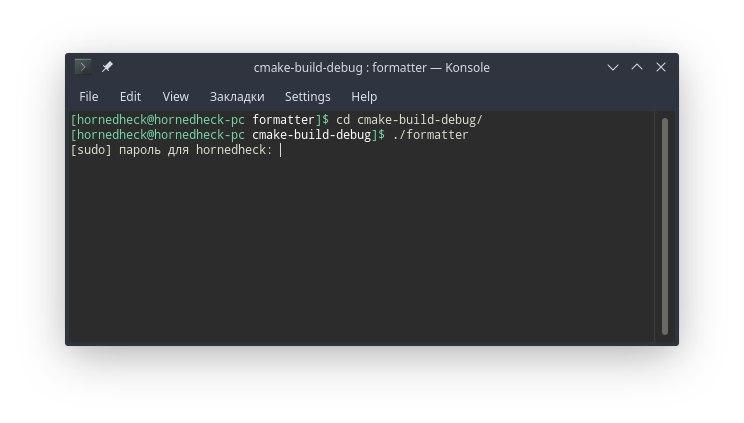


Рисунок 5.3 – модуль «процесс форматирования этап 1»

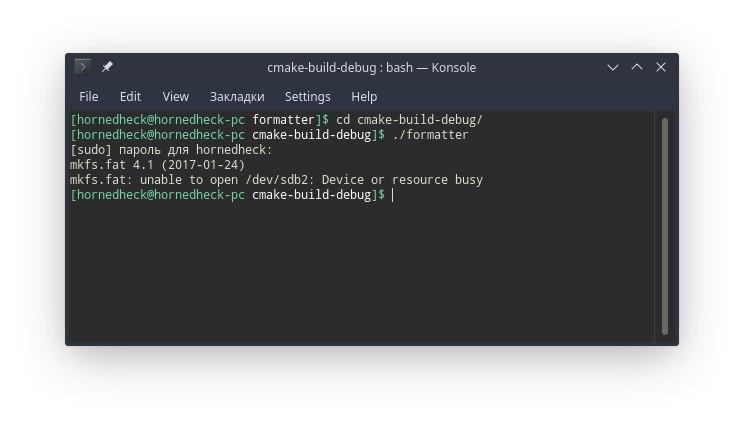


Рисунок 5.4 – модуль «процесс форматирования этап 2»

# ТЕСТИРОВАНИЕ

Тестирование проводилось на 3 персональных компьютерах с различными дистрибутивами Linux и на 2 твердотельных накопителях .

Проведем попытки запуска программы при отсутствии правильного пути в командной строке:

При запуске исполняемого файла появляется окно и оповещение о том, что нет такого файла или каталога.

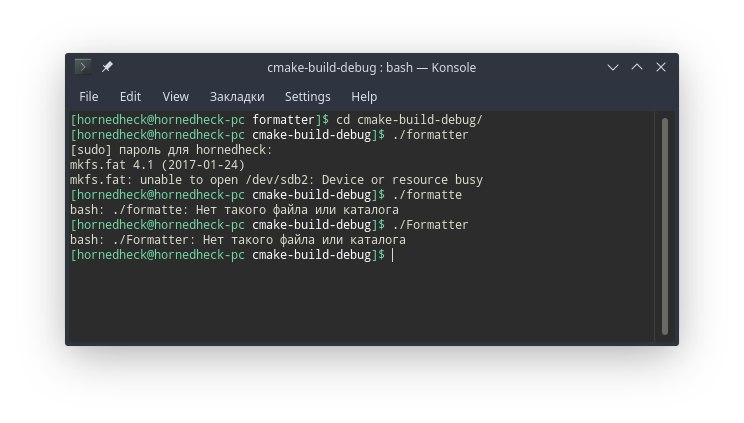


Рисунок 6.1 – ошибка при отсутствии корректного пути

Если устройство смонтировано, то форматирование невозможно

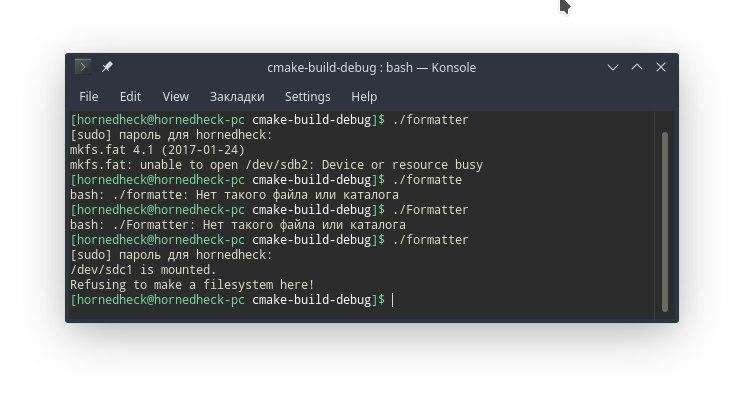


Рисунок 6.2 – ошибка при попытке отформатировать смонтированное устройство

Результат форматирования можно наблюдать на рисунке 6.3 во вкладке управление разделами KDE:

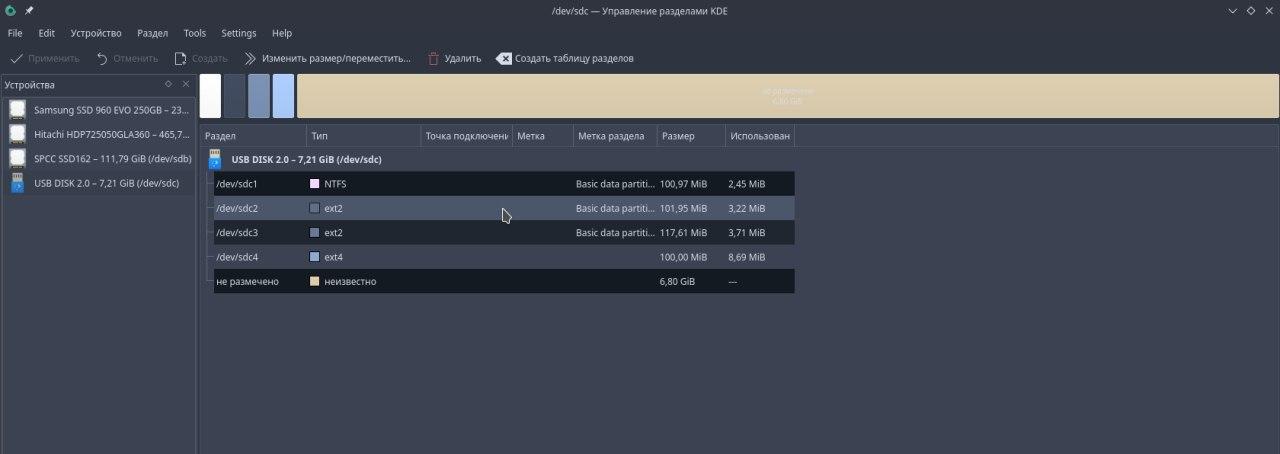


Рисунок 6.3 – результат работы программы

# ЗАКЛЮЧЕНИЕ

В процессе выполнения курсового проекта овладела практическими навыками проектирования и разработки законченного, отлаженного и протестированного программного продукта с использованием методик объектно-ориентированного программирования, углубила знания по оформлению документации на программный продукт. Было разработано приложение, которое изменяет файловую систему на жёстком диске, твердотельном накопителе. Разработанное приложение удовлетворяет условиям технического задания. Но можно отметить, что данное приложение можно совершенствовать: добавить, например, больше файловых систем для форматирования.

Были проведены необходимые тесты продукта, а также продемонстрированно применение программы.

# 

# СПИСОК ИСТОЧНИКОВ

1. FTP - протокол [Электронный ресурс]. – электронные данные. – Режим доступа: https://ru.wikipedia.org/wiki/FTP
2. Клиент-серверная архитектура [Электронный ресурс]. – электронные данные.– Режим доступа: [https://ru.wikipedia.org/wiki/](https://ru.wikipedia.org/wiki/Сокет_(программный_интерфейс))Клиент\_—\_сервер
3. Сокет\_(программный\_интерфейс) [Электронный ресурс]. – электронные данные.– Режим доступа: [https://ru.wikipedia.org/wiki/](https://ru.wikipedia.org/wiki/Сокет_(программный_интерфейс))Сокет\_(программный\_интерфейс)
4. Хост [Электронный ресурс]. – электронные данные.– Режим доступа: [https://ru.wikipedia.org/wiki/](https://ru.wikipedia.org/wiki/Сокет_(программный_интерфейс))Хост
5. Порт [Электронный ресурс]. – электронные данные.– Режим доступа: <https://ru.wikipedia.org/wiki/Порт_(компьютерные>\_сети)
6. Поток выполнения [Электронный ресурс]. – электронные данные.– Режим доступа: https://ru.wikipedia.org/wiki/Поток\_выполнения
7. Многопоточность [Электронный ресурс]. – электронные данные. – Режим доступа: <https://ru.wikipedia.org/wiki/Многопоточность>
8. CLion [Электронный ресурс]. – электронные данные. – Режим доступа: <https://www.jetbrains.com/ru-ru/clion/>
9. Mkfs . – За пределами проекта «Linux® с нуля. Пер. с англ. Ромоданов Н.А – М.:ДМК Пресс,2014. – 746с.:ил.
10. Mkfs. – Администрирование GNU/Linux с нуля. 2 изд. перераб. и доп. –СПб.:БХВ – Петербург,2010. – 576 с.:ил.+(Дистрибутивы на CD-ROM) –(Системный администратор)

# 

# 