Министерство образования Республики Беларусь Учреждение образования «Белорусский государственный университет

информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

Кафедра электронных вычислительных машин

Дисциплина: Системное программное обеспечение вычислительных машин

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовому проекту на тему

МНОГОПОТОЧНЫЙ FTP КЛИЕНТ

БГУИР КП 1-40 02 01 012 ПЗ

Студент: группы 850504, Линкевич В.Н.

Руководитель: ассистент каф. ЭВМ, Глоба А.А.

Минск 2020

СОДЕРЖАНИЕ

[Введение 3](#_Toc27321280)

[Обзор исотчников 4](#_Toc27321281)

[1.1 Основные термины 5](#_Toc27321283)

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

[2 Структурное проектирование 5](#_Toc27321282)

[3 Функциональное проектирование 6](#_Toc27321286)

[4 Разработка программных модулей 13](#_Toc27321289)

[5 Руководство пользователя 13](#_Toc27321289)

[6 Тестирование 16](#_Toc27321292)

[Заключение 18](#_Toc27321293)

[Список источников 19](#_Toc27321294)

[Приложениее 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 пользователям требуются следующие функции: быстрая пакетная загрузка файлов на сервер, быстрое скачивание, управление файлами и папками на хостинге.Liberation Sans

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

1. ОБЗОР ИСТОЧНИКОВ

## **Основные термины**

 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]

**«**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]

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]

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].

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].

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].

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 аргументов командной 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.

**2.** **СТРУКТУРНОЕ ПРОЕКТИРОВАНИE**

Структурная схема изображена в Приложении А.

Данная программа имеет 4 модуля: авторизация, обработка запроса пользователя, отправка команды на сервер, получение ответа от сервера.

В модуле «авторизация» осуществляется подключение к серверу и вход в личный кабинет пользователя.

В модуле «обработка запроса пользователя» пользователь может использовать следующие команды:

\* STOR

\* RETR

\* PWD

\* LIST

\* CWD

\* MKD

\* CDUP

\* HELP

\* QUIT

В модуле «отправка команды на сервер» осуществляется передача обработка команды с последующей передачей команды на сервер.

В модуле «получение ответа от сервера» происходит считывание и анализ данных которые были получены от сервера.

# **3** **ФУНКЦИОНАЛЬНОЕ ПРОЕКТИРОВАНИЕ**

В этом разделе рассмотрим функций написанные в данной курсовой работе.

Одна из главных функций в курсовой работе является:

void run(char\* ip, char\* port)

Данная функция запускается с старта программы и принимает в качестве параметров IP сервера, а также порт по которому будет происходить подключение. Здесь будет происходить подключение к серверу, авторизация пользователя и прием команд. Блок-схема данной функций представлена в Приложений Б.

int connectToHost(char\* ip, char\* port)

Данная функция служит для подключение клиента к серверу. На вход в функцию подается IP сервера и порт по которому происходит подключение. На выходе мы получаем дескриптор файла по которому происходит подключение, если у функций не удастся подключится она закроется выведя в терминале причину.

int userLogin(int sockfd)

Эта функция вызывается сразу после подключения к серверу. В данную функцию надо передать дескриптор файла, а на выход мы получаем 0 если смогли подключится к профилю и -1 если произошла ошибка. Здесь можно войти под своим личным аккаунтам для чего надо ввести логин или пароль с учетом реестра или возможно войти под общим аккаунтом anonymous в котором будут доступны ограниченные действия.

void sendCommand(int sock\_fd, const char\* cmd, const char\* info)

В этой функции осуществляется отправка команды на сервер. В качестве параметров она принимает дескриптор файла, саму команду а также второй параметр, если он нужен, иначе передаем пустую строку, на выходе мы ничего не принимаем, но при неудачной отправке сообщения программа закроется выTimes New Romanведя сообщение об ошибке.

int getReplyCode(int sockfd)

Следующая функция вызывается после sendCommand так как она читает код который возвращает сервер, а также при необходимости вызывает функцию перехода в пассивный режим. Данная функция получает дескриптор файла и возвращает код который вернул сервер или -1 если не удалось прочесть ответ сервера.

void enteringPassiveMode(char\* command)

Данная функция вводит клиент в пассивный режим. В функцию передается команда оправленная сервером.

# **4 РАЗРАБОТКА ПРОГРАММНЫХ МОДУЛЕЙ**

В данной главе будет рассмотрен функционал выполняемый программой и приведен алгоритм ее работы.

Работа начинается с определение количества параметров переданных в командной строке. Так если мы не передадим параметры программа завершится выдав сообщение о пустой командной строке. При одном параметре она запустит программу сделав порт подключения 21 что является стандартом для многих серверов. И при двух параметрах программа передаст их в качестве аргументов в функцию.

if (argc != 2 && argc != 3) {

printf("Usage: %s <host> [<port>]\n", argv[0]);

exit(-1);

} else if (argc == 2) { run(argv[1], "21"); }

else { run(argv[1], argv[2]); }

Далее мы заходим в функцию run в которой изначально вызываем connectToHost, где происходит подключения к хосту сервера, после удачного подключения мы входим в функцию userLogin где происходит вход в личный аккаунт. После входа в аккаунт нам доступен список команд которые мы можем отправить на сервер. В данной главе рассмотрим две наиболее важные команды LIST, STOR.

Начнем с обработки команды LIST. Данная команда возвращает информацию о файле или каталоге, если она указана, то возвращается информация о текущем рабочем каталоге.

После ввода пользователем команды она обрабатывается и вызывается функция вызывается функция list в которую мы передаем дескриптор файла. Функция сразу отправляет серверу команду перехода в пассивный режим в котором происходит изменение TPC-порта и подключение к серверу после чего мы оправляем саму команду LIST и ждем ответа сервера 150 что значит что файл находится в хорошем состоянии и вот-вот начнется отправка, после чего подключаемся к данным передаваемым сервером. После извлечение информации закрываем соединение и получаем ответ от сервера 250 что значит о том что наша команда успешна выполнена.

В случае запроса STOR после обработки команды создается новый поток для выполнения данной функций.

if (pthread\_create(&id,NULL, stor, NULL)!=0) {

errorReport("Thread create error.");

}

Из-за того что эта команда выполняется в отдельном потоке мы можем дальше работать с программой не смотря на то как долго будет выполняться функция. Это имеет большие преимущества т.к. при передачи на сервер файла больших размеров может произойти подвисание программы, т. к. главный поток занят отправкой файла он не может параллельно принимать другие запросы. А так мы можем отправлять очень большие файлы, что может привести к долгому времени ожидания отправки файла.

При созданий потока он сразу переводит клиент в пассивный режим. После чего подключается к новому сокету и отправляет команду на передачу данных на сервер. При получении кода 125 ,что означает подключение к данным уже открыто, начинается передача данных. После передачи процесс закрывает сокет и если получает от код 226 значит все прошло успешно. После того как поток отработал он удаляется.

# **5 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ**

В данном приложений будет рассмотрено использование курсового проекта. Данное приложение является простым в освоении.

Изначально нам надо открыть командную строку в папке где находится исполняемый файл. Далее вводим команду ./(название исполняемого файла) через пробел вводим IP адрес сервера, и при желаний можно через пробел ввести порт, если порт не будет введен будет выбран 21 порт.

После успешного подключения к серверу на консоль будет написано «Username:», в данном поле надо вести свои логин или anonymous. При входе как анонимный пользователь вам не надо вводить пароль поэтому можете начинать отправлять команды на сервер. Если вы все же входите в свой аккаунт после удачного ввода логина нужно вести пароль.

Если все сделано правильно в командной строке должно появится [Client command]. На этом этапе мы можем отправлять серверу команды:

1. «RETR» - Скачивание файла с сервер. В качестве второго аргумента надо передавать называние файла который хотим скачать.
2. «STOR» - Загрузка файла на сервер. В качестве второго аргумента надо передать название файла который хотим загрузить на сервер. Важно чтобы файл который мы хотим загрузить находился в той же директорий что и исполняемый фаил.
3. «PWD» - Получение рабочей директории.
4. «LIST» - Возвращает информацию о каталоге.
5. «CWD» - Изменяет рабочую директорию. В качестве второго параметра передаем название директорий.
6. «MKD» - Создаем папку. В качестве второго параметра передаем название папки.
7. «CDUP» - Переходим в родительский каталог.
8. «HELP» - Вызывает список команд и их краткое описание. Так же вызывается когда не введена никакая другая команда.
9. «QUIT» - Разъединение с сервером и выход из программы.

# **6 ТЕСТИРОВАНИЕ**

В данной главе проведено тестирование приложение

Таблица 1 – Тестирование запросов

|  |  |  |  |
| --- | --- | --- | --- |
| Тип запроса | Действие | Ожидаемый  результат | Успешность  тестирования |
| 1 | 2 | 3 | 4 |
| «RETR» | Ввод в консоль команды «retr name.txt» и нажатие «ENTER» | Скачивание с сервера файла name.txt | Успешно |
| «STOR» | Ввод в консоль команды «stor name.txt» и нажатие «ENTER» | Загрузка на сервер файла name.txt | Успешно |
| «PWD» | Ввод в консоль команды «pwd» и нажатие «ENTER» | Вывод в консоль рабочего каталога. | Успешно |
| «LIST» | Ввод в консоль команды «list» и нажатие «ENTER» | Вывод в консоль информации о рабочем каталоге. | Успешно |
| «CWD» | Ввод в консоль команды «cwd NewFolder» и нажатие «ENTER» | Переход в каталог под названием NewFolder. | Успешно |
| «MKD» | Ввод в консоль команды «mkd NewFolder» и нажатие «ENTER» | Создание новой папки NewFolder в рабочем каталоге. | Успешно |
| «CDUP» | Ввод в консоль команды «cdup» и нажатие «ENTER» | Переход в родительский каталог. | Успешно |

Продолжение Таблицы 1

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 2 | 3 | 4 |
| «HELP» | Ввод в консоль команды «help» или «?» и нажатие «ENTER» | Вывод краткой информации о всех командах | Успешно |
| «QUIT» | Ввод в консоль команды «quit» и нажатие «ENTER» | Отсоединение от сервера и выход из программы | Успешно |

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

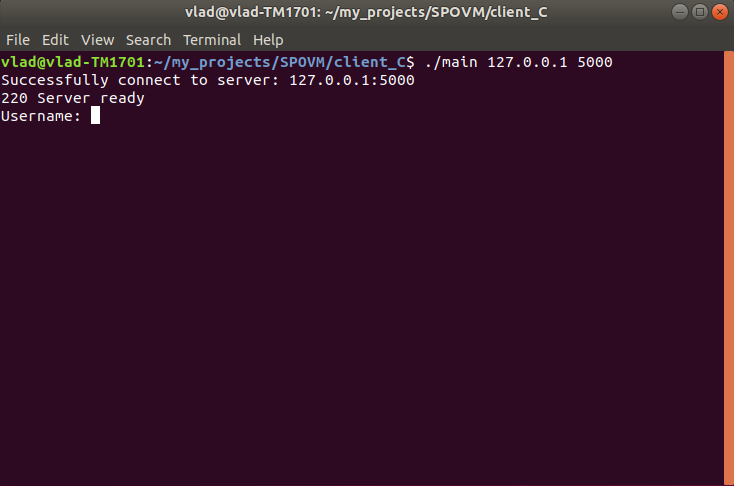
Запуск клиента отображено на рисунке 1

Рисунок 6.1 — Запуск клиента

Вход в аккаунт изображен на рисунке 2

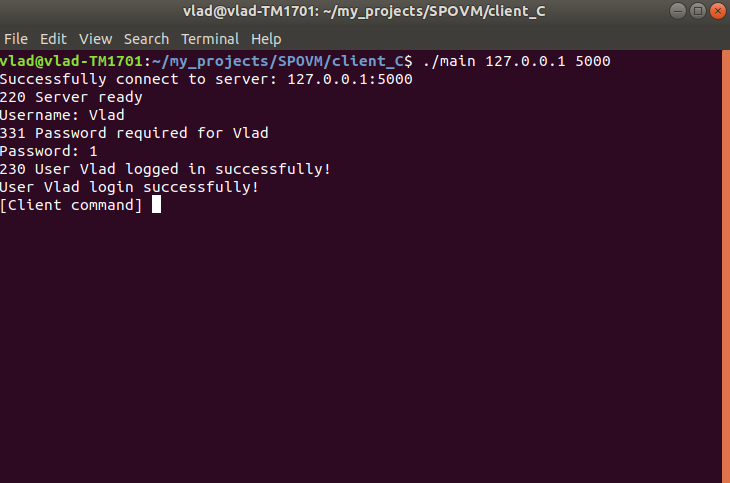


Рисунок 6.2 — вход в аккаунт

## Тестирование команды «RETR» изображено на рисунке 2

## 

## Рисунок 6.3 — скачивание файла с сервера

## Тестирование команды «STOR» изображено на рисунке 4

### 

### Рисунок 6.4 — загрузка файла на сервер

### Тестирование команды «PWD» изображено на рисунке 5

### 

### Рисунок 6.5 — отображение рабочего каталога

### Тестирование команды «LIST» изображено на рисунке 6

### 

### Рисунок 6.6 — получение информации текущего каталога

### Тестирование команды «CWD» изображено на рисунке 7

### 

### Рисунок 6.7 — переход в каталог под названием NewFolder

### Тестирование команды «MKD» изображено на рисунке 8

##### 

##### Рисунок 6.8 — создание новой папки NewFolder

### Тестирование команды «CDUP» изображено на рисунке 9

### 

### Рисунок 6.9 — перемещение в родительский каталог

### Тестирование команды «HELP» изображено на рисунке 10

### 

### Рисунок 6.10 — использование команды «help\?»

### Тестирование команды «QUIT» изображено на рисунке 11

### 

### Рисунок 6.11 — выход из приложения

# Результаты тестирования команд полностью подтверждают работоспособность программы, следовательно, приложение соответствует исходным требованиям.

ЗАКЛЮЧЕНИЕ

В результате работы над этим проектом была разработан многопоточный

FTP клиент. В проекте было реализована многопоточность и простой консольный интерфейс. Так же были выполнены все условия, поставленные темой курсового проекта. В результате тестирования проблем выявлено не было. Был получен опыт по работе с системой контроля версий git и по оформлению документации к проектам.

# **С****ПИСОК 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)

ПРИЛОЖЕНИЕ А

(Обязательное) Структурная схема программы

# 

ПРИЛОЖЕНИЕ Б

(Обязательное)

Блок-схема алгоритма

**ПРИЛОЖЕНИЕ В**

(обязательное)

Ведомость курсового проекта