

МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего профессионального образования
«Московский государственный технический университет радиотехники, электроники и
автоматики»

МГТУ МИРЭА

Институт информационных технологий (ИТ)

«наименование факультета»

Кафедра базовая автоматизированных систем организационного управления (АСОУ) №239 МГТУ
МИРЭА при ФГУП НИИ «Восход»

«наименование кафедры»

КУРСОВАЯ РАБОТА

по дисциплине

Программирование 1

<Наименование дисциплины>

Тема курсовой работы: Приложение для получения удалённого содержимого с
сетевого сервера

<Название темы курсовой работы/проекта>

Студент группы ИББО-03-14

<код группы>

<подпись студента>

Максимов А. Ю.

<ФИО студента>

Руководитель курсовой работы

ассистент

<должность, звание,
уч.степень>

<подпись руководителя>

Хлебников А.А.

< ФИО руководителя >

Рецензент (при наличии)

<должность, звание,
уч.степень>

<подпись рецензента>

< ФИО рецензента>

Работа представлена к защите

«__»_____201_ г.

<подпись студента>

«Допущен к защите»

«__»_____201_ г.

<подпись руководителя>

МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего профессионального образования
«Московский государственный технический университет радиотехники, электроники и автоматики»
МГТУ МИРЭА

Институт информационных технологий (ИТ)

«наименование факультета»

Кафедра базовая автоматизированных систем организационного управления (АСОУ) №239 МГТУМИРЭА при ФГУП
НИИ «Восход»

«наименование кафедры»

УТВЕРЖДАЮ

Заведующий

кафедрой _____ Н.В. Волков

«___» _____ 201__ г.

ЗАДАНИЕ

на выполнение курсовой работы

по дисциплине

«Программирование 1»

«Название дисциплины»

Студент	Максимов Алексей Юрьевич	Группа

1. Тема курсовой работы

Приложение для получения удалённого содержимого с сетевого сервера

2. Исходные данные

- Описание приложения wget

2. Перечень вопросов, подлежащих разработке, и обязательного графического материала:

- разработка текстового интерфейса и параметров запуска программы.
- разработка приложения для получения удалённого содержимого с сетевого сервера.

4. Срок представления к защите курсового проекта (работы): до «___» _____ 201__ г.

Задание на курсовой проект
(работу) выдал

«___» _____ 201__ г.

<подпись руководителя
проекта>

Хлебников А.А.

<Ф.И.О. руководителя
проекта>

Задание на курсовой проект
(работу) получил

«___» _____ 201__ г.

<подпись студента-
исполнителя проекта>

Максимов А.Ю.

<Ф.И.О. студента-
исполнителя проекта>

Мониторинг процесса выполнения курсового проекта (работы)

№ э т а п а	Наименование этапа курсового проекта, работы	Этап курсового проекта, работы выполнил и представил результаты руководителю проекта (работы), <i>дата и подпись исполнителя</i>	Работу по этапу курсового проекта (работы) принял на рассмотрение, <i>дата и подпись руководителя</i>	Рекомендации и замечания по этапу курсового проекта (работы) выдал исполнителю, <i>дата и подпись руководителя</i>	Оценка выполнения этапа курсового проекта, (работы) (в <i>соответствии с балльно- рейтинговой системой</i>)	Комментарии руководителя курсового проекта (работы)
1	Разработка задания на КР					
2	Анализ задачи					
3	Разработка алгоритма задачи					
4	Разработка структуры проекта					
5	Разработка интерфейса пользователя					
6	Разработка схемы программных взаимодействия единиц, группируя по модулям					
7	Разработка приложения					
8	Разработка пояснительной записки					
9	Защита курсовой работы					

Реферат

Объект исследования – программирование на языке C/C++

Предмет исследования – работа с сетевым протоколом HTTP/1.1 и сокетами в языке программирования C/C++.

Цель работы: закрепление и углубление практических знаний по программированию на языке C/C++, ориентированных на работу с сокетами (socket).

В процессе работы проводилась разработка алгоритмов, тестирование и отладка как самого приложения, так и отдельных его модулей.

Область возможного практического применения: создание крупных коммерческих анализаторов открытого содержимого сети Интернет.

ЛИСТ ЗАМЕЧАНИЙ

[illegible]

Содержание

Реферат.....	4
Введение.....	7
Описание программного обеспечения wget.....	9
Основная часть.....	10
Обоснование выбора средств и методов для выполнения задания.....	10
<iostream>.....	11
<netdb.h>.....	12
<sys/socket.h>.....	13
<arpa/inet.h>.....	14
Интерфейс командной строки.....	15
Реализация связи по протоколу HTTP/1.1 на примере реализации программы wget.....	17
Заключение.....	19
Приложение.....	21

Введение

Широкое распространение **языка C** на различных типах компьютеров (иногда называемых аппаратными платформами) привело, к сожалению, ко многим вариациям языка. Они были похожи, но несовместимы друг с другом. Это было серьезной проблемой для разработчиков программ, нуждавшихся в написании совместимых программ, которые можно было бы выполнять на нескольких платформах. Стало ясно, что необходима стандартная версия C. В 1983г. ANSI (Американский Национальный Комитет Стандартов) сформировал технический комитет X3J11 для создания стандарта языка C (чтобы "обеспечить недвусмысленное и машинно-независимое определение языка"). В 1989 стандарт был утвержден. ANSI скооперировался с ISO (Международной Организацией Стандартов), чтобы стандартизовать C в международном масштабе; совместный стандарт был опубликован в 1990 году и назван ANSI/ISO 9899:1990. Этот стандарт совершенствуется до сих пор и поддерживается большинством фирм разработчиков компиляторов.

Бьерн Страуструп высвободил объектно-ориентированный потенциал C путем перенесения возможностей классов *Simula 67* в C. Первоначально новый язык носил имя "C с классами" и только потом стал называться C++. Язык C++ достиг популярности, будучи разработанным в Bell Labs, позже он был перенесен в другие индустрии и корпорации. Сегодня это один из наиболее популярных языков программирования в мире. C++ наследует как хорошие, так и плохие стороны C.

Бьерн Страуструп: "Я придумал C++, записал его первоначальное определение и выполнил первую реализацию. Я выбрал и сформулировал критерии проектирования C++, разработал его основные возможности и отвечал за судьбу предложений по расширению языка в комитете по стандартизации C++", - пишет автор самого популярного языка программирования. - "Язык C++ многим обязан языку C, и язык C остается подмножеством языка C++ (но в C++ устранены несколько серьезных брешей системы типов C). Я также сохранил средства C,

которые являются достаточно низкоуровневыми, чтобы справляться с самыми критическими системными задачами. *Язык C*, в свою очередь многим обязан своему предшественнику, BCPL; кстати, стиль комментариев // был взят в C++ из BCPL. Другим основным источником вдохновения был язык Simula67. Концепция классов (с производными классами и виртуальными функциями) была позаимствована из него. Средства перегрузки операторов и возможность помещения объявлений в любом месте, где может быть записана инструкция, напоминает Algol68. "

Название C++ выдумал Рик Масситти. Название указывает на эволюционную природу перехода к нему от C. "++" - это операция приращения в C. Чуть более короткое имя C+ является синтаксической ошибкой; кроме того, оно уже было использовано как имя совсем другого языка. Знатоки семантики C находят, что C+ + хуже, чем ++C. Названия D язык не получил, поскольку он является расширением C и в нем не делается попыток исцеляться от проблем путем выбрасывания различных особенностей...

Изначально *язык программирования C++* был разработан, чтобы автору и его друзьям не приходилось программировать на ассемблере, C или других современных языках высокого уровня. Основным его предназначением было сделать написание хороших программ более простым и приятным для отдельного программиста. Плана разработки C++ на бумаге никогда не было; проект, документация и реализация двигались одновременно. Разумеется, внешний интерфейс C++ был написан на C++. Никогда не существовало "Проекта C++" и "Комитета по разработке C++". Поэтому C++ развивался и продолжает развиваться во всех направлениях, чтобы справляться со сложностями, с которыми сталкиваются пользователи, а также в процессе дискуссий автора с его друзьями и коллегами.

В **языке C++** полностью поддерживаются принципы объектно-ориентированного программирования, включая три кита, на которых оно стоит: инкапсуляцию, наследование и полиморфизм. *Инкапсуляция в C++* поддерживается посредством создания нестандартных (пользовательских) типов данных, называемых классами. *Язык C++* поддерживает наследование. Это значит, что можно объявить новый тип данных (класс), который является расширением существующего.

Хотя **язык программирования C++** справедливо называют продолжением C и любая работоспособная программа на языке C будет поддерживаться *компилятором C++*, при переходе от C к C++ был сделан весьма существенный скачок. *Язык C++* выигрывал от своего родства с языком C в течение многих лет, поскольку многие программисты обнаружили, что для того, чтобы в полной мере воспользоваться преимуществами *языка C++*, им нужно отказаться от некоторых своих прежних знаний и приобрести новые, а именно: изучить новый способ концептуальности и решения проблем программирования. Перед тем как начинать осваивать C++, *Страуструп* и большинство других программистов, использующих C++ считают изучение языка C необязательным.

Язык программирования C++ в настоящее время считается господствующим языком, используемым для разработки коммерческих продуктов, 90% игр пишутся на C++ с применением DirectX.

Описание программного обеспечения wget

Wget— свободная неинтерактивная консольная программа для загрузки файлов по сети. Поддерживает протоколы **HTTP**, **FTP** и **HTTPS**, а также поддерживает работу через **HTTP прокси-сервер**. Программа включена почти во все дистрибутивы **GNU/Linux**.

Wget является неинтерактивной программой. Это означает, что после её запуска пользователь может повлиять на её работу только с помощью средств управления процессами операционной системы. Как правило, для этого используются сочетания клавиш **Ctrl+C** при необходимости прерывания работы программы и **Ctrl+Z** для помещения текущего задания в фон. Современные web-браузеры как правило имеют функцию закладки файлов, однако так как браузер рассчитан на интерактивный режим работы, то скачивание большого количества файлов вручную может быть утомительным. Браузеры, как правило, не предоставляют средств для автоматизации подобных задач. Wget же, например, поддерживает загрузку URL, указанных в файле. Таким образом можно составить список файлов, а в любое удобное время скачать их с помощью wget. Интерфейс командной строки позволяет управлять wget из других программ и скриптов, что используется при автоматизации загрузки файлов (регулярные обновления, мониторинг доступности сервера и т. д.).

Wget позволяет загружать любые файлы во всемирной паутине (в том числе и (X)HTML-страницы) по протоколам http и https, а также файлы и списки директорий по протоколу ftp.

Файлы можно скачивать рекурсивно по ссылкам в HTML страницах, как с одного сайта с определённой глубиной следования по ссылкам, так и с нескольких. Помимо этого, при загрузке по ftp файлы можно скачивать «по маске» имени (то есть можно задавать с помощью «*» группу файлов).

Wget поддерживает докачку файла в случае обрыва соединения.

Основная часть

Обоснование выбора средств и методов для выполнения задания

Средством для выполнения задания была выбрана среда разработки Sublime Text 3 от SUBLIME HQ PTY LTD в связке с g++ 4.9.3. Приложение разработано под операционную систему Gentoo linux (gcc-4.9.3, glibc-2.20-r2, 4.1.1-gentoo-r1-apcels_ssd i686) (на других ОС/версиях библиотек приложение не тестировалось). Предметной областью задания является изучение работы с сокетами и протоколом HTTP/1.1

Для работы, в программе подключаются следующие заголовочные файлы (библиотеки):

<iostream>

Операции **ввода/вывода** выполняются с помощью классов `istream` (поточный ввод) и `ostream` (поточный вывод). Третий класс, `iostream`, является производным от них и поддерживает двунаправленный ввод/вывод. Для удобства в библиотеке определены три стандартных объекта-потока:

`cin` – объект класса `istream`, соответствующий стандартному вводу. В общем случае он позволяет читать данные с терминала пользователя;

`cout` – объект класса `ostream`, соответствующий стандартному выводу. В общем случае он позволяет выводить данные на терминал пользователя;

`cerr` – объект класса `ostream`, соответствующий стандартному выводу для ошибок. В этот поток мы направляем сообщения об ошибках программы.

Вывод осуществляется, как правило, с помощью перегруженного оператора сдвига влево (`<<`), а ввод – с помощью оператора сдвига вправо (`>>`)

Также в библиотеке реализовано определение пространства имён `std::`.

<netdb.h>

Содержит описание важных для работы с сетевыми базами данных методов, таких, как `getaddrinfo` (определение параметров, необходимых для подключения к удалённому хосту, по имени хоста и опциональным параметрам подключения, служащим для выделения нужного набора параметров подключения из возможных), и структур, таких, как `addrinfo` (структура, содержащая основную информацию для подключения)

Совместима со стандартом POSIX.

<sys/socket.h>

POSIX-совместимая библиотека для работы с сокетами. Содержит описание методов для работы с, как видно из названия, сокетами. Примеры функций:

```
int      accept(int, struct sockaddr *restrict, socklen_t *restrict);
int      bind(int, const struct sockaddr *, socklen_t);
int      connect(int, const struct sockaddr *, socklen_t);
int      getpeername(int, struct sockaddr *restrict, socklen_t *restrict);
int      getsockname(int, struct sockaddr *restrict, socklen_t *restrict);
int      getsockopt(int, int, int, void *restrict, socklen_t *restrict);
int      listen(int, int);
ssize_t  recv(int, void *, size_t, int);
ssize_t  recvfrom(int, void *restrict, size_t, int,
                  struct sockaddr *restrict, socklen_t *restrict);
ssize_t  recvmsg(int, struct msghdr *, int);
ssize_t  send(int, const void *, size_t, int);
ssize_t  sendmsg(int, const struct msghdr *, int);
ssize_t  sendto(int, const void *, size_t, int, const struct sockaddr *,
                socklen_t);
int      setsockopt(int, int, int, const void *, socklen_t);
int      shutdown(int, int);
int      socket(int, int, int);
int      socketatmark(int);
int      socketpair(int, int, int, int[2]);
```

Также данная библиотека содержит описание таких структур, как `sockaddr` (используется для хранения данных, необходимых для открытия сокета / приёма данных с сокета.)

<arpa/inet.h>

В данной реализации программы wget данная библиотека используется только для отладки (конвертации IP-адреса из подструктуры ai_addr структуры addrinfo, полученной в итоге работы функции getaddrinfo, в понятный и привычный человеческому глазу Ipv4-адрес вида «127.0.0.1»)

Также данная библиотека содержит практически альтернативные (гораздо более урезанные) описанным в <netdb.h> методы для работы с информацией об интернет-соединениях:

```
in_addr_t    inet_addr(const char *);
char         *inet_ntoa(struct in_addr);
const char   *inet_ntop(int, const void *restrict, char *restrict,
                        socklen_t);
int          inet_pton(int, const char *restrict, void *restrict);
```

Интерфейс командной строки.

Приложение неинтерактивное и, как следствие, не содержит меню. Интерфейс реализуется выводом справки и введением ключей (параметров) как аргументов при запуске программы:

```
apcel@apcel ~/git/apcel/Kursach_2Sem $ ./main.out
A simple wget implementation
Usage: ./main.out [-id][-o FILENAME][--ignore-all] ADDRESS
        -d
        enable debugging
        -i
        ignore some errors such as no '200 HTTP' message
        -o FILENAME
        manually select output filename
        --ignore-all
        ignore terrible errors such as opening output file descriptor error.
        This option is not recommended and was implemented only for testing purpose.
apcel@apcel ~/git/apcel/Kursach_2Sem $
```

Порядок следования опций командной строки не имеет значения за исключением случаев, когда опции дублируются. В таком случае используется последняя полученная опция:

```
apcel@apcel ~/git/apcel/Курсовая Прога 2семестр "wget" $ ./wget http://programmer5.ucoz.ru/_nw/0/14146845.jpg
-o 1 -o 2
The file '2' was written. Bye.
```

Аргументы командной строки описаны в краткой справке программы. Справка вызывается путём запуска программы без аргументов (см. скриншот в начале раздела).

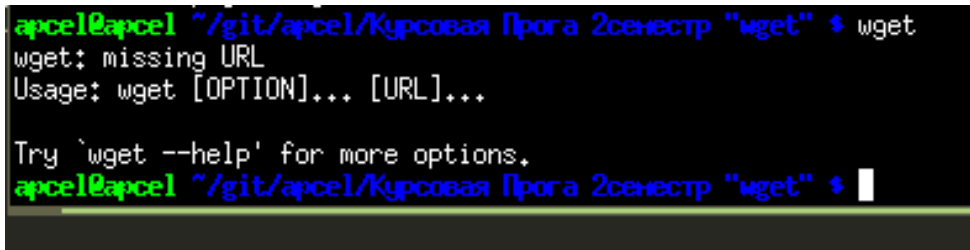
Не знакомые программе аргументы игнорируются.

В квадратных скобках указаны необязательные опции (для них имеются значения по умолчанию).

Оформление краткой справки сделано в соответствии с негласными правилами оформления кратких справок консольных программ под операционные системы семейства UNIX (имеются небольшие расхождения со среднестатистическими 'канонами' оформления, но не больше среднего расхождения оформления справки для ОС Gentoo Linux)

Аргумент ADDRESS – ссылка, содержащая хотя бы один символ «слэш» – '/'

На скриншоте ниже изображён вывод 'оригинальной' (скомпилированной из исходников, поставляемых командой разработчиков GNU/Linux, с комплектом системно-специфичных патчей для ОС Gentoo Linux) программы wget:



```
apcel@apcel ~/git/apcel/Курсовая Прога 2сенестр "wget" $ wget
wget: missing URL
Usage: wget [OPTION]... [URL]...

Try 'wget --help' for more options.
apcel@apcel ~/git/apcel/Курсовая Прога 2сенестр "wget" $
```

При вводе `wget —help` на экран выводится расширенная справка по использованию программы.

В выполненной в рамках курсовой работы реализации программы поддерживается малое число опций, что позволяет не делить справку на краткую и расширенную.

Реализация связи по протоколу HTTP/1.1 на примере реализации программы wget

Реализация программы состоит из следующих пунктов:

1. Инициализация, проверка входных данных.
2. В случае отсутствия входных аргументов – вывод справки и выход из программы
3. Обработка поданных на вход программы аргументов: установка требуемых флагов и значений нужных переменных.
4. Разделение поданной на вход ссылки на составные части: протокол, имя хоста, имя файла
5. Определение параметров для открытия соединения
6. Открытие соединения: создание сокета, привязка к требуемым параметрам
7. Формирование GET-запроса с последующей его отправкой интернет-хосту (в дальнейшем – серверу)
8. Принятие ответного сообщения с сервера
9. Обработка заголовка сообщения
10. Открытие файла для принятия данных
11. Принятие нужной последовательности байт с последующей записью их в файл

На каждом пункте выполняется проверка правильности выполнения шага. При некорректном его выполнении, если не указано иное аргументами «-i» или «--ignore-all» (или, в некоторых случаях, определением макроса DEBUG на стадии компиляции), выполняется вывод краткого сообщения об ошибке и выход из программы.

Отдельно хотелось бы рассмотреть реализацию обработки заголовка полученного с сервера сообщения.

Так как стандарт HTTP/1.1 допускает несколько вариантов формирования заголовков сообщений, в обработке имеющих существенные различия, в программе для некритичных частей ответа реализован не самый надёжный метод разбора, основанный на поиске рекомендованного стандартом варианта ответа (поиск «200 OK» на ~225 строке кода).

Для критически важных частей реализован более точный поиск, в теории учитывающий все варианты использования стандарта (также такая критическая часть ответа сервера, как размер файла, более однозначно определена в стандарте, что упростило реализацию алгоритма для его вычленения из заголовка)

Также после приёма с сервера информация сохраняется в сетевой буфер, выделяемый при открытии сокета операционной системой. Буфер занимает память, из-за чего стоит как можно быстрее «достать» оттуда информацию для возможности приёма остальных частей ответа сервера: на обработку ответа сервера затрачивается как можно меньше времени за счёт того самого упрощения алгоритма поиска некритичных частей ответа сервера.

Использование цикла для копирования ответа сервера можно оправдать, так как:

- Данные уже находятся в памяти компьютера, т. е., не приходится, как можно подумать, `contentLength` раз принимать пакеты, из которых вычленять один байт, и по байту писать их на диск.
- Это позволяет наиболее среднестатистически быстро освободить буфер для сохранения содержимого следующего пакета (если из буфера «взять» данных на байт меньше размера пакета и спокойно их обрабатывать, следующий пакет данных в буфер уже не поместится, что может создать дополнительную нагрузку на участвующие в обмене данными вычислительные устройства, вплоть до потери данных => повторной их передачи)
- Это позволяет сохранить содержимое передаваемого файла с точностью до байта и избежать коллизий, т. к. `sizeof(char)` по определению – байт.

Заключение

В процессе выполнения курсовой работы проводилась разработка алгоритмов, тестирование и отладка, как самого приложения, так и отдельных его модулей.

Как видно из описания используемых библиотек, они все входят в стандарт POSIX => код можно скомпилировать и запустить на любой POSIX-совместимой операционной системе (Linux, *BSD, Solaris..), существует возможность запуска программы на ОС семейства windows с помощью спец. утилит (например, Cygwin)

В ходе разработки выяснилось, что с сокетами в C/C++ работать проще, чем об этом сообщается или не сообщается, и что в ОС Linux есть достаточное количество встроенной (доступной без выхода в сеть Интернет) документации для написания большей части программ, в том числе программ для работы с сетью Интернет.

В целом, этот проект помог мне закрепить и улучшить как общие знания по программированию на языке C/C++, так и по работе с сокетами и выделением из последовательности символов нужных частей (некое подобие регулярных выражений), которые использовались в данном проекте.

Также приобретён дополнительный опыт по поиску информации на заданные темы в разнообразных англоязычных источниках: официальной документации RFC, секциях справки man, поисковом сайте google.com

В процессе отладки программы был вспомнен старый сайт класса, картинки с которого и брались для проверки корректной работы программы.

Список литературы

1. Бьерн Страуструп "Язык программирования C++. 3-е издание.
2. Дейтел Х. Как программировать на C++/ Х.Дейтел, П.Дейтел. – М. Бином-Пресс, 2014 – 1200 с.
3. Крупник А. Изучаем C++/ СПб.: Питер, 2003 – 256 с.
4. manpages for linux, секция «man 3: Library calls (functions within program libraries)», разделы по используемым библиотекам.

Приложение

```
#include <unistd.h>
#include <string.h>
#include <netdb.h>
#include <iostream>
#include <sys/socket.h>

bool debug_enabled = false;
//#define DEBUG
#ifdef DEBUG
    #define DEBUG_FUCKN
    #define DEBUG_GETHOST

    #include <arpa/inet.h>
#endif

struct linkStruct
{
    std::string protocol = "";
    std::string hostname = "";
    std::string relative = "";
    std::string filename = "";
};

struct requestSkeleton
{
    std::string method = "GET";
    char SP = char(32);
    std::string httpVer = "HTTP/1.1";
    std::string requestHeader = "\r\nAccept: */*\r\nAccept-Charset: utf-8\r\nConnection:
Keep-Alive\r\n";
    std::string CRLF = "\r\n\r\n";
    int flags = 0;
};

void show_help(char* cmdname);
int parceHost(char* address, linkStruct *result);
void log(std::string message);
void log(FILE * fd, std::string message);

int main (int argc, char* argv[])
{
    #   ifdef DEBUG
        debug_enabled = true;
    #   endif
```

```

    if (argc < 2) {

        log("Sry, no arguments: " + std::to_string(argc) + " : " + std::string(argv[argc
- 1]));

        show_help(argv[0]);
    #    ifdef DEBUG
        argv[1] = (char
*)"https://pp.vk.me/c623120/v623120157/465ee/c5fCboWQibg.jpg";
        argc += 1;
    #    else
        return -1;
    #    endif
    }

    int temporaryInteger = 1;

    linkStruct addr;
    bool ignoreErrors = false;
    bool ignoreAllErrors = false;

    for (int i = 1; i < argc; i++) {
        std::string s = argv[i];
        if(s == "-o") {
            log("Output filename will be " + std::string(argv[i + 1]));
            addr.filename = argv[i + 1];
            i += 1;
        } else if (s == "-i") {
            ignoreErrors = true;
            log("Some errors will be ignored");
        } else if(s == "--ignore-all") {
            ignoreAllErrors = true;
            log("Ignoring everything. :C");
        } else if (s == "-d") {
            debug_enabled = true;
            log("Debugging enabled");
        } else if (s == "-h" || s == "--help") {
            show_help(argv[0]);
            return 0;
        } else if (s.find("/") != s.npos) {
            log("found link-like argument" + std::string(argv[i]));
            temporaryInteger = i;
        } else {
            log(stderr, "Option \' + std::string(argv[i]) + \'\' was not recognized,
sorry. Ignoring.");
        }
    }
}

```

```

log("found link-like " + std::string(argv[temporaryInteger]));

int linkNum = temporaryInteger;
temporaryInteger = parceHost(argv[temporaryInteger], &addr);

if(temporaryInteger != 0) {
    log(stderr, "Parsing function returned non-zero value :(");
    log(stderr, "Link seemed to be " + std::string(argv[linkNum]));
    if((!ignoreErrors) && (!ignoreAllErrors))
        return temporaryInteger;
}

struct addrinfo *he = new struct addrinfo;
memset(he, 0, sizeof *he);
he->ai_socktype = SOCK_STREAM;
he->ai_family = AF_INET;
//he->ai_flags = AF_UNSPEC;

temporaryInteger = getaddrinfo (addr.hostname.c_str(), "80", he, &he);

log ("getaddrinfo done: " + std::to_string(temporaryInteger) + ": " +
gai_strerror(temporaryInteger));

if(temporaryInteger != 0) {
    temporaryInteger = getaddrinfo(addr.hostname.c_str(), "80", NULL, &he);
    log(stderr, "getaddrinfo done[+1]: " + std::to_string(temporaryInteger) + ": " +
gai_strerror(temporaryInteger));
    log(stderr, "This may mean that server doesn't use http (forced https, for
example)\n");
    #    ifndef DEBUG
    if((!ignoreErrors) && (!ignoreAllErrors)) {
        log(stderr, "Consider using -i if you know what you're doing");
        return 25;
    }
    #    endif
}

log("SOCK_DGRAM = " + std::to_string(SOCK_DGRAM));
log("SOCK_STREAM = " + std::to_string(SOCK_STREAM));

#    ifdef DEBUG_FUCKN
    struct addrinfo * temporaryPointer = he;

```



```

log("he->ai_socktype = " + std::to_string(he->ai_socktype));

struct sockaddr_in *tempinadr = (struct sockaddr_in *)he->ai_addr;

log("struct addrinfo {\n
    \nint      ai_flags;          " + std::to_string(he->ai_flags) + \
    \nint      ai_family;        " + std::to_string(he->ai_family) + \
    \nint      ai_socktype;      " + std::to_string(he->ai_socktype) + \
    \nint      ai_protocol;      " + std::to_string(he->ai_protocol) + \
    \nsize_t    ai_addrlen;      " + std::to_string(he->ai_addrlen) + \
    \nstruct    sockaddr *ai_addr; " + std::to_string(he->ai_addr->sa_family) +
" " + inet_ntoa(tempinadr->sin_addr));

log("char      *ai_canonname;    /* canonical name */" \
    "\nstruct  addrinfo *ai_next; /* this struct can form a linked list */" /*+
std::to_string(he->ai_next) +*/\
    "\n}");
delete temporaryPointer;
# endif

log("he->ai_socktype = " + std::to_string(he->ai_socktype));

////////////////////////////////////
//So we have host adress from link.
int socketFd;
socketFd = socket(he->ai_family, he->ai_socktype, he->ai_protocol);

log("socketFd = " + std::to_string(socketFd));
if(socketFd < 0)
{
    log(stderr, "Error opening socketFd: " + std::to_string(socketFd));
    if(!ignoreAllErrors)
        return 2;
}

if (addr.protocol != "http") {
    log(stderr, "You're trying to use non-http protocol. I don't know what it will
end.");

    log(stdout, "Use -i option to ignore errors.");
    if((!ignoreErrors) && (!ignoreAllErrors))

```

```

        return 15;
    }

    log("Trying to connect..");
    temporaryInteger = connect(socketFd, he->ai_addr, he->ai_addrlen);
    log("connected; return value: " + std::to_string(temporaryInteger));
    if (temporaryInteger != 0) {
        log(stderr, "connection error");
        if(!ignoreAllErrors)
            return 50;
    }

    //////////////////////////////////////
    //Connection established. It's time to send data! ^.^
    std::string message = "";
    requestSkeleton requestSkeleton;
    message = requestSkeleton.method + requestSkeleton.SP + addr.relative +
requestSkeleton.SP + requestSkeleton.httpVer
        + requestSkeleton.SP + requestSkeleton.requestHeader + "Host:" + addr.hostname +
requestSkeleton.CRLF;
    log(message);

    temporaryInteger = send(socketFd, message.c_str(), message.size(), 0);
    log("Sent data: " + std::to_string(temporaryInteger));

    log("Trying to open file: " + std::string("./" + addr.filename) + "\n...");
    FILE * localFd = fopen(std::string("./" + addr.filename).c_str(), "wb+");
    if (localFd == NULL) {
        log(stderr, "Error opening localFd");
        if(!ignoreAllErrors)
            return 100;
    }
    log("Success.");

    //////////////////////////////////////
    log("starting recv in a loop");
    std::string server_reply;
    char * server_reply_buf;

    while(server_reply.find("\r\n\r\n") == server_reply.npos) {
        recv(socketFd, server_reply_buf, sizeof(char), 0);
        server_reply.push_back(*server_reply_buf);
    }

    log(server_reply);

```

```

if(server_reply.find('\0') != server_reply.npos) {
    log(stderr, "Ughm, could parse response header illegally, sry.");
    debug_enabled = true;
    log("Turning debugging on.");
}

```

```

server_reply.push_back('\0');
for(int i = 0; server_reply[i] != '\0'; ++i)
    server_reply[i] = std::toupper(server_reply[i]);
//log(server_reply);

```

```

int temp = 0;
int temp2 = 0;

```

```

temp = server_reply.find("200 OK");
if(temp == server_reply.npos) {
    log(stderr, "Could not find '\200 OK\' in the header.\n");
    if(!ignoreErrors) && (!ignoreAllErrors)
        return 2;
}

```

```

temp = server_reply.find("CONTENT-LENGTH:");
if(temp == server_reply.npos) {
    log(stderr, "Could not find content-length option in http header, sry.");
    if(!ignoreErrors) && (!ignoreAllErrors)
        return 116;
}

```

```

temp = server_reply.find(":", temp);
temp += 2;
temp2 = server_reply.find_first_not_of("0123456789", temp);
if(temp2 == server_reply.npos) {
    log(stderr, "WTF? Could not find content-length value");
    if(!ignoreErrors) && (!ignoreAllErrors)
        return 115;
}

```

```

int contentLength = stoi(server_reply.substr(temp, abs(temp2-temp)), NULL, 10);
log(std::to_string(contentLength) + " bytes");

```

```

contentLength = (contentLength / sizeof(char) ); // okay, lets divide by 1.
if((contentLength % sizeof(char)) != 0) {

```

```

    log(stderr, "Something is wrong: contentLength % sizeof(char) returned non-zero
value");

    if(!ignoreAllErrors)
        return 250;

```

```

    }
    log(std::to_string(contentLength) + " chars");

    int i = 0;
    int recvd = 1;
    while((++i < contentLength) && (recvd != 0)) {
        recvd = recv(socketFd, server_reply_buf, sizeof(char), 0);
        fprintf(localFd, "%c" , *server_reply_buf);
    }
    log(stdout, "The file \'' + addr.filename + '\'' was written. Bye.");

    fflush(localFd);
    fclose(localFd);
    close(socketFd);
    return 0;
}

void show_help(char* cmdname) {
    log(stdout, "A simple wget implementation");
    log(stdout, "Usage:\t" + std::string(cmdname) + " [-id][-o FILENAME][--ignore-all]
ADDRESS" );
    log(stdout, "\t" + std::string(cmdname) + " [-h][--help] \n");

    log(stdout, "\t\t-h --help\n\t show this help message");
    log(stdout, "\t\t-d\n\t enable debugging");
    log(stdout, "\t\t-i\n\t ignore some errors such as no '200 HTTP' message in server
response");
    log(stdout, "\t\t-o FILENAME\n\t manually select output filename");
    log(stdout, "\t\t--ignore-all\n\t ignore terrible errors such as opening output file
descriptor error.");
    log(stdout, "\tThis option is not recommended and was implemented only for testing
purposes.");
}

int parceHost(char* address, linkStruct * result) {
    int begin = 0;
    std::string s = address;
    int end = s.size() / sizeof(char);
    int temp = 0;
    temp = s.find("//", 1);
    if (temp != s.npos)
    {
        begin = temp + 2;
        result->protocol = s.substr(0, temp - 1);
    } else {
        temp = 0;
        result->protocol = "http";
    }
}

```

```

temp = s.find("/", begin + 1);
if (temp != s.npos) {
    end = temp;
    result->hostname = s.substr(begin, abs(end - begin));
    result->relative = s.substr(end);
}

if(result->filename == "") {
    temp = s.rfind("/");
    end = s.find("?", temp + 1) - 1;
    if(temp != s.npos) {
        if(end == s.npos - 1)
            result->filename = s.substr(temp + 1);

        //if (end < temp) // We're trying to cut string from TEMP pos to left. This
is not normal as TEMP pos is the beginning of filename.
        //    return -3;

        result->filename = s.substr(temp + 1, end - temp);
    } else {
        return -2;
    }
}

#   ifdef DEBUG_GETHOST
    log("parceHost returns:");
    log("begin = " + std::to_string(begin));
    log("end   = " + std::to_string(end));
    //log("ukhm, host may be the \'' + s + '\'");
    log("result->protocol = " + result->protocol);
    log("result->hostname = " + result->hostname);
    log("result->relative = " + result->relative);
    log("result->filename = " + result->filename);
#   endif
    return 0;
}

void log(std::string message) {
    if (debug_enabled)
        log(stdout, message);
}

void log( FILE * fd, std::string message) {
    fprintf(fd, "%s\n", message.c_str());
}

```