Министерство образования Республики Беларусь

Учреждение образования

«Белорусский государственный университет
информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

Кафедра электронных вычислительных машин

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовой работе

на тему

HTTP - сервер

БГУИР КР 1-40 02 01 326 ПЗ

Студент: Черевко И. М.

Руководитель: Глоба А.А.

Минск 2022

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Несколько десятилетий назад интернет покорил мир, сделав доступ к информации как никогда легче и удобнее. С тех пор прошло немало времени, и сегодня миллиарды людей используют его для проведения досуга и работы. Мы можем пользоваться интернетом на огромном количестве устройств, от смарт-часов до наших персональных компьютеров. Для того,чтобы сделать всемирную паутину универсальной, был ввёден протокол HTTP[6].

HTTP — широко распространённый протокол передачи данных, изначально предназначенный для передачи гипертекстовых документов (то есть документов, которые могут содержать ссылки, позволяющие организовать переход к другим документам).

HTTP в настоящее время повсеместно используется во всемирной паутине для получения информации с веб-сайтов. В 2006 году в Северной Америке доля HTTP-трафика превысила долю P2P-сетей и составила 46 %, из которых почти половина — это передача потокового видео и звука.

HTTP используется также в качестве «транспорта» для других протоколов прикладного уровня, таких как SOAP, XML-RPC, WebDAV.

1 ОБЗОР ЛИТЕРАТУРЫ

Следует начать с определения сервера.

Сервер - программный компонент вычислительной системы, выполняющий сервисные (обслуживающие) функции по запросу клиента, предоставляя ему доступ к определённым ресурсам или услугам.

URI (Uniform Resource Identifier, унифицированный идентификатор ресурса) — путь до конкретного ресурса (например, документа), над которым необходимо осуществить операцию (например, в случае использования метода GET подразумевается получение ресурса). Некоторые запросы могут не относиться к какому-либо ресурсу, в этом случае вместо URI в стартовую строку может быть добавлена звёздочка (астериск, символ «*»). Например, это может быть запрос, который относится к самому веб-серверу, а не какому-либо конкретному ресурсу.

HTML (от англ. *HyperText Markup Language* — «язык гипертекстовой разметки») — стандартизированный язык разметки документов для просмотра веб-страниц в браузере. Веб-браузеры получают HTML документ от сервера по протоколам HTTP/HTTPS или открывают с локального диска , далее интерпретируют код в интерфейс, который будет отображаться на экране монитора.

Элементы разметки являются строительными блоками HTML страниц. С помощью HTML разные конструкции, изображения и другие объекты, такие как интерактивная веб-форма, могут быть встроены в отображаемую страницу. HTML предоставляет средства для создания заголовков, абзацев, списков, ссылок, цитат и других элементов. Элементы HTML выделяются тегами, записанными с использованием угловых скобок.

Бра́узер- прикладное программное обеспечение для просмотра страниц, содержания веб-документов, компьютерных файлов и их каталогов, управления веб-приложениями, а также для решения других задач. В глобальной сети браузеры используют для запроса, обработки, манипулирования и отображения содержания веб-сайтов. Многие современные браузеры также могут использоваться для обмена файлами с серверами FTP, а также для непосредственного просмотра содержания файлов многих графических форматов (gif, jpeg, png, svg), аудио- и видеоформатов (mp3, mpeg), текстовых форматов (pdf, djvu) и других файлов. Примерами таких браузеров являются всем известные Google Chrome,Mozilla Firefox,Opera и многие другие, причём данные браузеры доступны так же и на мобильных устройствах, что позволяет пользоваться ими в любой ситуации.

Протокол связи — набор определённых правил или соглашений интерфейса логического уровня, который определяет обмен данными между различными программами. Эти правила задают единообразный способ передачи сообщений и обработки ошибок.

Основой же HTTP протокола является технология «клиент-сервер», то есть предполагается существование:

1) Потребителей (клиентов), которые инициируют соединение и посылают запрос;

2) Поставщиков (серверов), которые ожидают соединения для получения запроса, производят необходимые действия и возвращают обратно сообщение с результатом.

Основным объектом манипуляции в HTTP является ресурс, на который указывает URI в запросе клиента. Обычно такими ресурсами являются хранящиеся на сервере файлы, но ими могут быть логические объекты или что-то абстрактное . Особенностью протокола HTTP является возможность указать в запросе и ответе способ представления одного и того же ресурса по различным параметрам: формату, кодировке, языку и т. д. (в частности, для этого используется HTTP-заголовок). Именно благодаря возможности указания способа кодирования сообщения клиент и сервер могут обмениваться двоичными данными, хотя данный протокол является текстовым.

HTTP — протокол прикладного уровня , аналогичными ему являются FTP и SMTP. Обмен сообщениями идёт по обыкновенной схеме «запрос-ответ». Для идентификации ресурсов HTTP использует глобальные URI. В отличие от многих других протоколов, HTTP не сохраняет своего состояния. Это означает отсутствие сохранения промежуточного состояния между парами «запрос-ответ». Компоненты, использующие HTTP, могут самостоятельно осуществлять сохранение информации о состоянии, связанной с последними запросами и ответами (например, «куки» на стороне клиента, «сессии» на стороне сервера). Браузер, посылающий запросы, может отслеживать задержки ответов. Сервер может хранить IP-адреса и заголовки запросов последних клиентов. Однако сам протокол не осведомлён о предыдущих запросах и ответах, в нём не предусмотрена внутренняя поддержка состояния, к нему не предъявляются такие требования.

Большинство протоколов предусматривает установление TCP-сессии, в ходе которой один раз происходит авторизация, и дальнейшие действия выполняются в контексте этой авторизации. HTTP же устанавливает отдельную TCP-сессию на каждый запрос, в более поздних версиях HTTP было разрешено делать несколько запросов в ходе одной TCP-сессии, но браузеры обычно запрашивают только страницу и включённые в неё

объекты (картинки, каскадные стили и т. п.), а затем сразу разрывают TCP-сессию. Для поддержки авторизованного (неанонимного) доступа в HTTP используются cookies, причём такой способ авторизации позволяет сохранить сессию даже после перезагрузки клиента и сервера.

При доступе к данным по FTP[7] или по файловым протоколам тип файла (точнее, тип содержащихся в нём данных) определяется по расширению имени файла, что не всегда удобно. HTTP перед тем, как передать сами данные, передаёт заголовок «Content -Type: тип/подтип», позволяющий клиенту однозначно определить, каким образом обрабатывать присланные данные. Это особенно важно при работе с CGI-скриптами, когда расширение имени файла указывает не на тип присылаемых клиенту данных, а на необходимость запуска данного файла на сервере и отправки клиенту результатов работы программы, записанной в этом файле (при этом один и тот же файл в зависимости от аргументов запроса и своих собственных соображений может порождать ответы разных типов — в простейшем случае картинки в разных форматах).

Кроме того, HTTP позволяет клиенту прислать на сервер параметры, которые будут переданы запускаемому CGI-скрипту. Для этого же в HTML были введены формы.

Перечисленные особенности HTTP позволили создавать поисковые машины (первой из которых стала AltaVista, созданная фирмой DEC), форумы и Internet-магазины. Это коммерциализировало Интернет, появились компании, основным полем деятельности которых стало предоставление доступа в Интернет (провайдеры) ,создание сайтов.

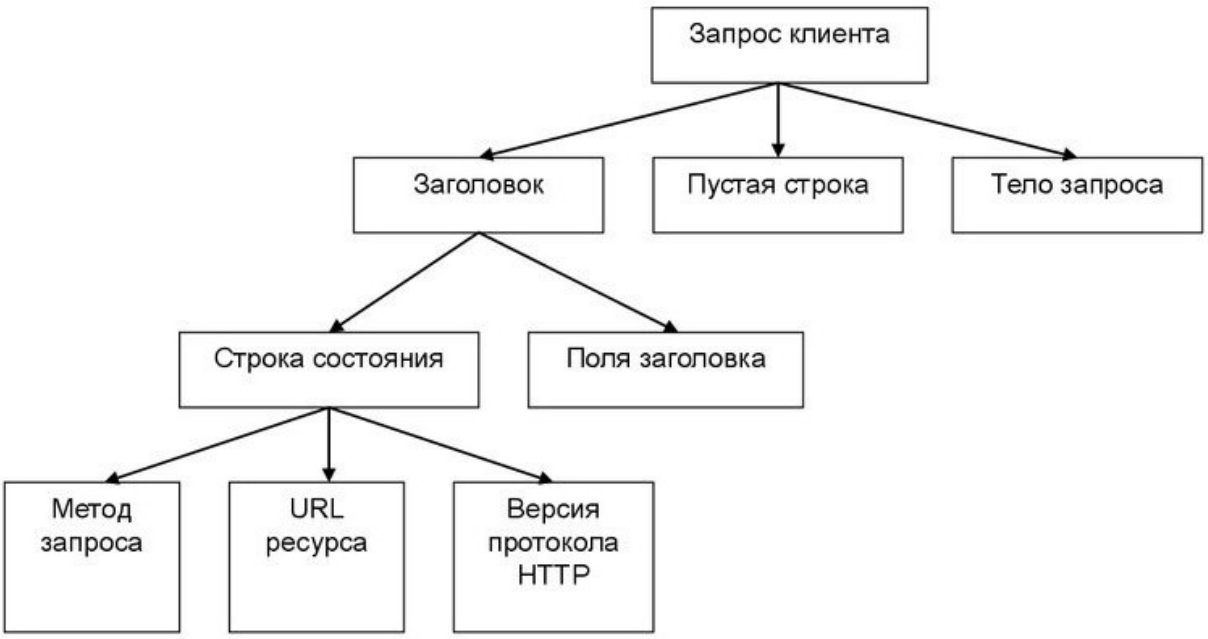

Рисунок 1.1 — Структура запроса клиента HTTP.

Понятия сервер, клиент и закреплённые за ними роли образуют программную концепцию «клиент-сервер».

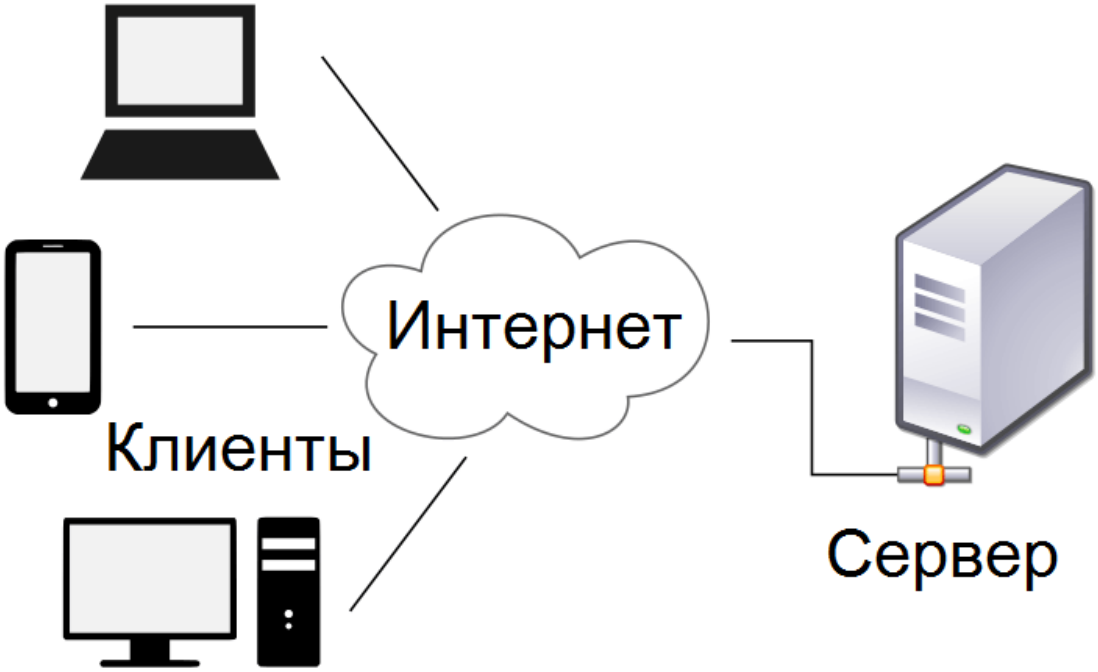

Рисунок 1.2 — Структура клиент-серверного приложения.

Характеристика «клиент-сервер» описывает отношения взаимодействующих программ в приложении. Серверный компонент предоставляет функцию или услугу одному или нескольким клиентам, которые инициируют запросы на такие услуги. Серверы классифицируются по предоставляемым ими услугам. Например, веб-сервер обслуживает веб-страницы, а файловый сервер обслуживает компьютерные файлы. Общий ресурс может быть любой из программного обеспечения и электронных компонентов компьютера — сервера, от программ и данных в процессорах и запоминающих устройств. Совместное использование ресурсов сервера представляет собой услугу.

Клиенты и серверы обмениваются сообщениями в шаблоне запрос-ответ. Клиент отправляет запрос, а сервер возвращает ответ. Этот обмен сообщениями является примером межпроцессного взаимодействия. Для взаимодействия компьютеры должны иметь общий язык, и они должны следовать правилам, чтобы и клиент, и сервер знали, чего ожидать. Язык и правила общения определены в протоколе связи. Все протоколы клиент-серверной модели работают на уровне приложений. Протокол прикладного уровня определяет основные шаблоны диалога. Чтобы ещё больше

формализовать обмен данными, сервер может реализовать интерфейс прикладного программирования (API). API[5] — это уровень абстракции для доступа к сервису. Ограничивая связь определённым форматом контента, он облегчает синтаксический анализ. Абстрагируя доступ, он облегчает межплатформенный обмен данными.

Сервер может получать запросы от множества различных клиентов за короткий период времени. Компьютер может выполнять только ограниченное количество задач в любой момент и полагается на систему планирования для определения приоритетов входящих запросов от клиентов для их удовлетворения. Чтобы предотвратить злоупотребления и максимизировать доступность серверное программное обеспечение может ограничивать доступность для клиентов. Атаки типа «отказ в обслуживании» используют обязанности сервера обрабатывать запросы, такие атаки действуют путем перегрузки сервера чрезмерной частотой запросов. Шифрование следует применять, если между клиентом и сервером должна передаваться конфиденциальная информация.

2 СИСТЕМНОЕ ПРОЕКТИРОВАНИЕ

2.1 Используемые инструменты проектирования

В качестве языка программирования выбран C, по причине высокой производительности, требуемой для обработки большого количества запросов и наличия опыта в использования данного языка.

В качестве компилятора будем использовать GNU GCC[8] как наиболее распространённый и удобный, так же автоматизируем сборку проекта с помощью make[9]

2.2 Моделирование программы

Архитектура программного средства является монолитной. Данный вид архитектуры легко развёртывается, простой в разработке. Отдельные части архитектуры вынесены в модули. Модули взаимодействуют через чётко определённые интерфейсы. Все части программного средства унифицированы. Основным критерием для выбора монолитной архитектуры была высокая производительность работы программного средства.

Архитектура программного средства состоит из 6 основных модулей:

- модуль установки адреса сервера;
- модуль отправки HTTP запроса;
- модуль обработки HTTP запроса;
- модуль ожидания запроса;
- модуль завершения соединения;
- модуль установки соединения.

Каждый модуль отвечает за определенную задачу. Такое разделение позволяет легко поддерживать, обновлять и расширять программное средство как вертикально, так и горизонтально.

2.2.1 Модуль установки адреса сервера

Данный модуль будет представлять собой простую установку адреса сервера, с котором в дальнейшем установится соединение . Данный модуль будет использоваться лишь один раз во время запуска сервера.

2.2.2 Модуль отправки HTTP запроса

Данный модуль будет представлять собой отправку запроса посредством возврата структуры данных с такими полями,как имя метода,путь,версию протокола ,состояние и индекс. Сразу после создания

этой структуры она будет передана на дальнейшую обработку другим модулем.

2.2.3 Модуль обработки HTTP запроса

Модуль будет реализовывать собой обработку запроса в зависимости от выбранного метода GET/POST,производить парсинг тела запроса и отправку результата обработки пользователю с записью .

2.2.4 Модуль ожидания запроса

Данный блок кода просто будет представлять собой бесконечный цикл с проверкой на установленное соединение и готовность принять запрос, и если оба эти условия удовлетворены и запрос отправлен то всё остальное управление будет отдано в модуль отправки запроса

2.2.5 Модуль завершения соединения

Данный модуль будет вызываться один раз в конце работы сервера и будет очищать все данные и так же закреплённый за сервером адрес, отключать соединение сервера с клиентом и завершать работу самого сервера.

2.2.6 Модуль установки соединения

Модуль представляет собой подключение к порту и сокету,в результате наше устанавливается соединение между клиентом и сервером по адресу,указанному в модуле установки адреса сервера.

Все вышеперечисленные модули позволяют обеспечить полноту выполняемых действий, соответственно необходимы для выполнения курсовой работы.

3 ФУНКЦИОНАЛЬНОЕ ПРОЕКТИРОВАНИЕ

Данная глава будет представлять собой ключевой раздел, дающий понимание работы сервера , с точки зрения описания отдельных функций и модулей, обработки данных, с приведением листинга и структурной диаграммой, вынесенной в приложение «А».

1. Файл http.c содержит основные функции для работы с запросами,

такие как:

- new_request(void)
- listen_http(HTTP *http)
- parsehtml_http(int conn, char* filename)
- _parse_request(HTTPreq *request, char *buffer, size_t size)
- handle_http(HTTP *http,char* path, void(*)(int,HTTPreq*))
- new_http(char *address)

Стоит рассмотреть функцию new_http(char* address),так как она неразрывно связана с установкой соединения,и будет сохранять наши данные о соединении. Для хранения этих самых данных используется структура HTTP, которая имеет следующий вид:

```
struct HTTP{
char* host;
int32_t len;
int32_t cap;
HastTab *tab;
}HTTP;
```

Для отправки запроса будет использоваться структура HTTPreq, содержащая такие поля, как метод запроса, путь к ресурсу, версию протокола, состояние и индекс. Создание нового запроса с помощью функции new_request() представляет собой возврат новой структуры HTTPreq,в которую дальше будут заноситься параметры запроса .

2. Файл net.c содержит функции,связанные с установкой и закрытием соединения. В нем содержатся такие функции, как:

- _parse_address(char* address,char* ipv4, char *port)
- connect_net(char *address), close_net(int conn)

Для установки соединения мы используя функцию connect_net(char *address) подключимся к сокету и откроем доступ к серверу по установленному заранее адресу.

Функция _parse_address(char* address,char* ipv4,char* port) используется для обработки переданного адреса и разбивки его на сам IP-адресс и порт,используещийся сервером.

3. Файл main.c содержит функции для обработки запроса /index и /about и в фунции main() вызываются фунции, описанные выше, по сути запуская сервер.

4. Файл hashtab.c содержит такие функции,как

- _get_hash(HashTab *hashtab,void* key)
- _strhash(char* s,size_t size)
- new_hashtab(size_t size,vtype_t key,vtype_t value)
- del_hashtab(HashTab *hashtab,void* key)
- in_hashtab(HashTab *hashtab,void* key)
- get_hashtab(HashTab *hashtab,void* key)
- set_hashtab(HashTab *hashtab,void* key)
- eq_hashtab(HashTab *hashtab,void* key)
- size_hashtab(HashTab *hashtab)
- free_hashtab(HashTab *hashtab)
- print_hashtab(HashTab *hashtab)

5. Файл tree.c содержит следующие функции

- _new_node(vtype_t key,vtype_t value)
- _set_tree(Tree* tree)
- set_key(vtype_t key)
- print_tree()
- del_tree(Tree* tree)

Функции, находящиеся в файлах tree.c,hashtab.c,type.c являются утилитарными и используются в основных модулях.

Схема отношений файлов представляет из себя список файлов и их функций и приведена в приложении. Структурная схема приведена в приложении.

Реализовав данный функционал мы можем увидеть, что у нас формируется структура проекта, которая содержит сам сервер, клиент для отправки запроса(браузер), и логгирование информации через терминал.

Более подробно про реализацию и внутреннее устройство некоторых функций будет указано в следующем разделе.

4 РАЗРАБОТКА ПРОГРАММНЫХ МОДУЛЕЙ

В данном разделе будет рассматриваться алгоритм работы программы.

Основной алгоритм работы программного обеспечения таков:

1) Установление соединения по указанному IP-адресу
2) Ожидание принятия запроса
3) Отправка запроса пользователем
4) Обработка запроса(определения GET/POST,передача тела запроса)
5) Ожидание нового запроса/Завершение работы сервера(пункт 6)
6) Закрытие соединения,завершение работы программы.

Модули, описанные в главе Системное проектирование обеспечивают полноту действий пользователя, и позволяют этому алгоритму работать. Реализация этих методов будет приведена в листинге кода.

Стоит отдельно рассмотреть некоторые основные функции, и привести их краткое описание, для более детального понимания работы программного обеспечения.

Функция new_request() используется для создания нового запроса,код этой функции будет иметь следующий вид:

```
HTTPreq new_request(){
        return (HTTPreq){
                .method = {0},
                .path={0},
                .proto={0},
                .state=0,
                .index=0,
        };
}
```

Функционально представляет собой создание нового экземпляра структуры HTTPreq , которая имеет вид:

```
struct HTTPreq{
        char method[16];
        char path[2048];
```

```
    char proto[16];
    uint8_t state;
    size_t index;
}HTTPreq;
```

Далее эта структура заполняется переданными пользователем данными,и проходит последующую обработку

Функция parse_request() представляет собой обработку запроса, то есть заполнение структуры HTTPreq данными, предоставленными пользователями.

Функция pasrehtml_http() производит обработку html документов и подготавливает их к отправке пользователю в ответ на запрос. Так, если мы нажмём кнопку about на сайте, то в функцию parsehtml_http() на обработку поступит файл about.html из корневой директории проекта.

Так же присутствует функция page404_http() которая вызывается при обработке исключительных ситуаций, таких как неправильно ввёденный URL. По сути своей является типичной ошибкой 404,информация о которой выводится пользователю.

Функции index_page() и about_page() реализуются практически идентично, за исключением парсинга страниц с помощью функции parsehtml_http(),так как для каждой функции используется свой HTML файл.

Функция listen_http() подразумевает собой бесконечный цикл с «прослушкой соединения», по сути своей являющийся ожиданием принятия запроса.

Функция free_http() которая очищает структуру HTTP последнего запроса, является чисто утилитарной функцией,которая нужна для очищения памяти.Остальные функции,такие как:

- switch_http()
- null_request()
- handle_http()
- parse_address()
- listen_net()

Стоит отметить функцию parse_address(), которая обрабатывает адрес, указанный при запуске сервера, производя парсинг данного адреса и разбивая его на IP-адрес и номер порта,закрепленного за сервером.

Некоторые другие функции являются утилитарными, и используются вышеописанными функциями.

Все реализованные модули позволяют реализовать поставленные задачи и обеспечить дальнейшую расширяемость кода. Подход к реализации и испльзование монолитной архитектуры проекта позволяет увеличивать функционал без особых проблем.

5 ПРОГРАММА И МЕТОДИКА ИСПЫТАНИЙ

Тестирование — это наблюдение за работой приложения в разных искусственно созданных ситуациях. Данные, полученные в ходе тестирования, важны при планировании последующей стратегии развития приложения. Это своего рода диагностика, которая влияет на многие дальнейшие действия.

Для тестирования разработанного приложения были сымитированы ошибки, чтобы проверить, как приложение на них реагирует.

5.1 Тестирование корректности обращения к серверу

В любом веб-приложении существует определённый набор запросов,которые доступны пользователю. Например, к популярному мессенджеру VK можно получить доступ используя сдедующий URL:vk.com.

Тем самым мы указываем сам ресурс,к которому идёт обращение. Дальше мы можем вызвать новостную ленту,написав в адресной строке браузера vk.com/feed. Так и у нашего сервера есть запрос /about, который выводит html страницу. Но если пользователь ввёдет несуществующий URL, он получит об этом сообщение, так называемую ошибку 404.

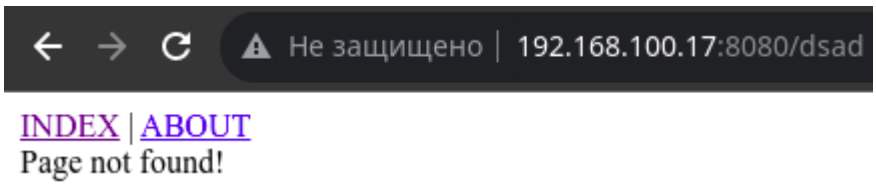

Рисунок 5.1 — Ошибка при вводе запроса

6 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ

Алгоритм работы программы таков:

1) Запуск программы используя терминал

2) Обращение к серверу используя любой браузер по установленному адресу

3) Отправка POST/GET запроса используя поле для ввода или открытие страницы используя кнопки index и about

Запуск сервера приведен на рисунке 6.1.

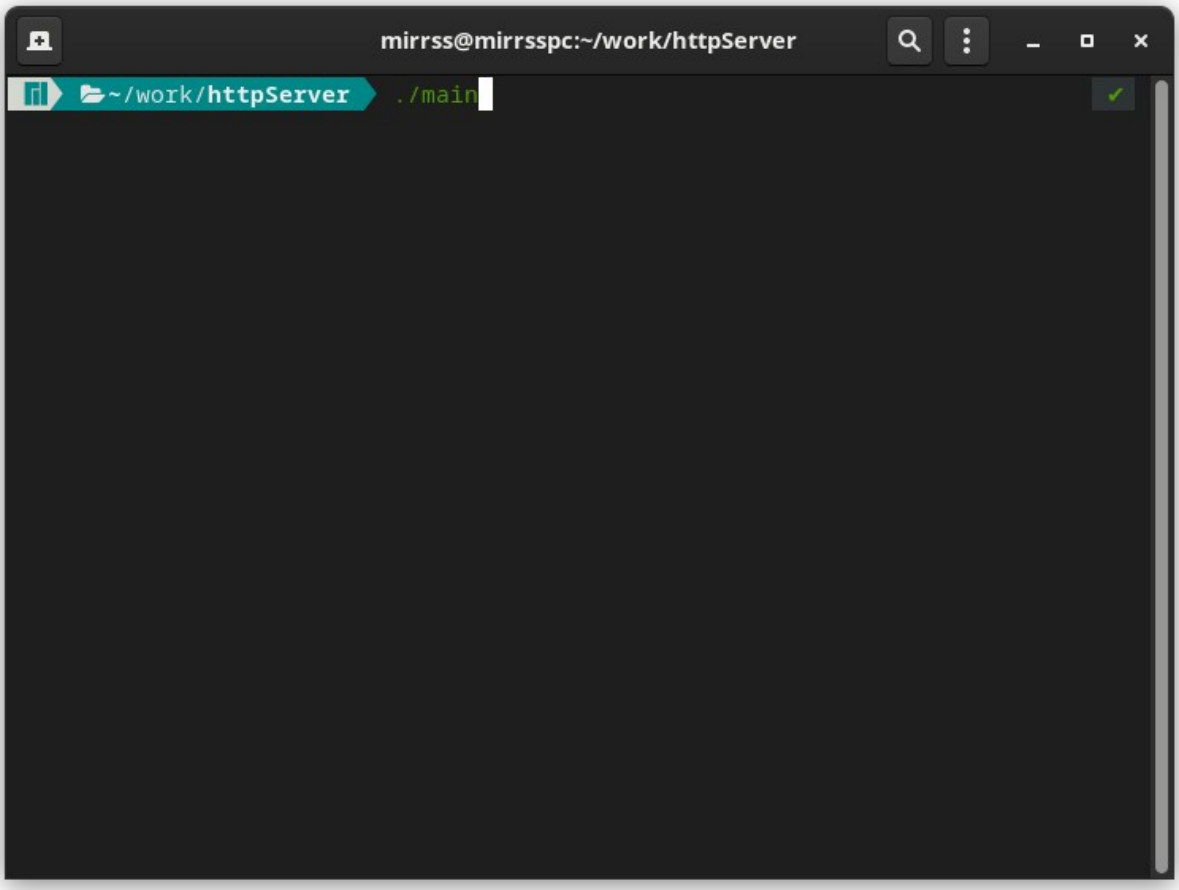

Рисунок 6.1 — Окно программы при запуске.

Обращение к серверу используя браузер(рисунок 6.2)

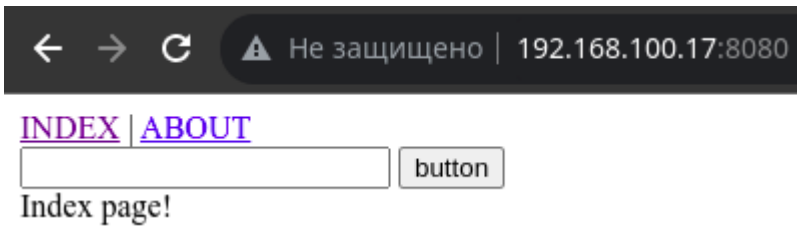

Рисунок 6.2 — Главная страница сайта.

Отправка GET запроса:

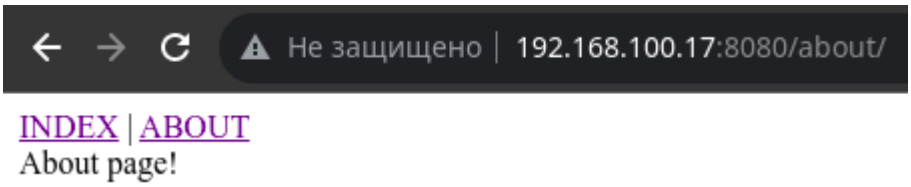

Рисунок 6.3.1 — Отправка GET запроса(вывод браузера)

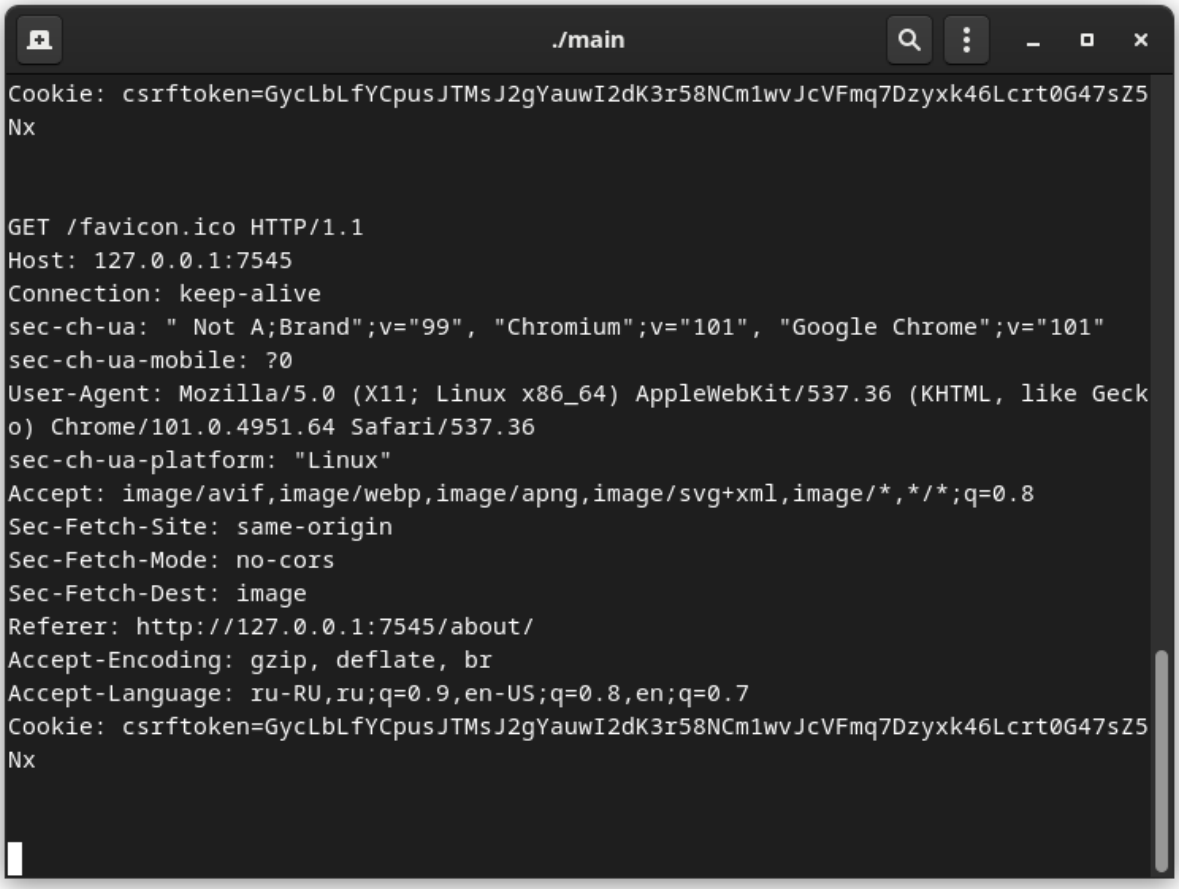

Рисунок 6.3.2 — Отправка GET запроса(вывод терминала).

Отправка POST запроса:

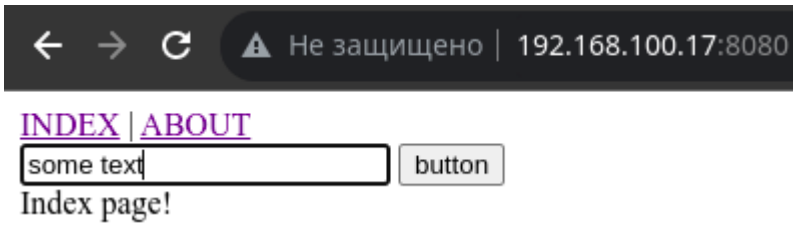

Рисунок 6.4.1 — Отправка POST запроса(вывод браузера).

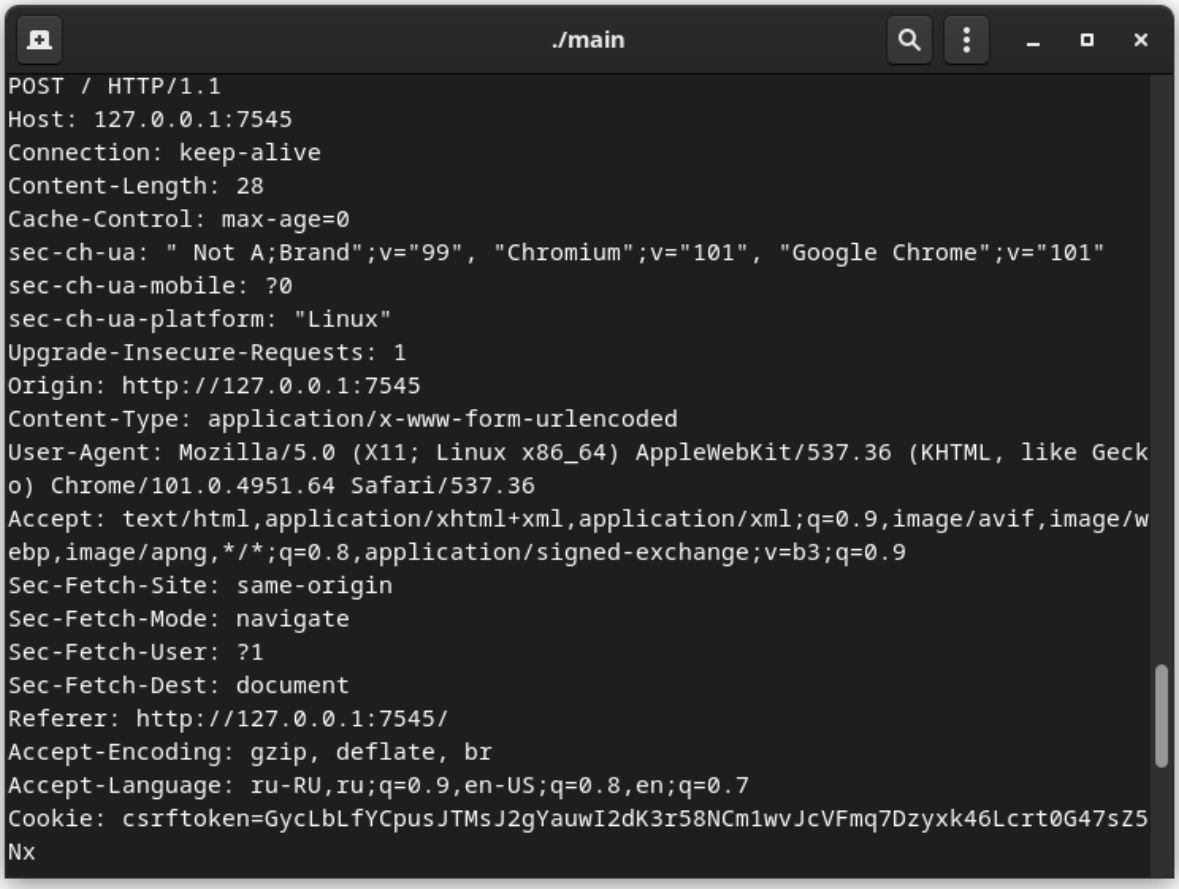

Рисунок 6.4.2 — Отправка POST запроса(вывод терминала).

ЗАКЛЮЧЕНИЕ

В данном разделе будут проведены итоги разработки программного обеспечения.

Плюсы данного приложения:

· Простой консольный интерфейс для отслеживания хода работы сервера;

· Возможность отправки как GET так и POST запроса;

· Достаточно большой потенциал для расширения функциональности.

· Стабильная работа из-за надежного протокола HTTP.

В результате выполнения курсового проекта были изучены темы проектирования серверного приложения, основы протоколов передачи данных и были углублены знания языка программирования C.

Работа была разделена на такие этапы, как анализ литературных источников, постановка требований к проектируемому программному продукту, системное и функциональное проектирование, конструирование программного продукта , разработка программных модулей и тестирование проекта. После последовательного выполнения вышеперечисленных этапов разработки было получено исправно работающее приложение, которое может быть усовершенствовано без особых затрат.

ЛИТЕРАТУРА

[1]. Лав Р. Системное программирование на Linux/ 2-е издание 2014.

[2]. Керниган Б. Язык программирования С/ 4-е издание М.:Питер, 2004.

[3]. Столяров А. В. Язык Си и начальное обучение программированию/ВМК.МГУ, 2010.

[4]. Гукин Д. Язык программирования Си для «чайников»/М.: Диалектика,2006

[5]. Что такое API? [Электронный ресурс]. Режим доступа: https://aws.amazon.com/ru/what-is/api/

[6]. Протокол HTTP [Электронный ресурс]. Режим доступа: https://developer.mozilla.org/ru/docs/Web/HTTP

[7]. Протокол FTP [Электронный ресурс]. Режим доступа: https://experience.dropbox.com/ru-ru/resources/what-is-ftp

[8]. GNU GCC — free compiler [Электронный ресурс]. Режим доступа: https://gcc.gnu.org/wiki

[9]. Make — free program collector [Электронный ресурс]. Режим доступа: https://www.gnu.org/software/make/

ПРИЛОЖЕНИЕ А

(обязательное)

ПРИЛОЖЕНИЕ Б

(обязательное)

ПРИЛОЖЕНИЕ В

(обязательное)

ПРИЛОЖЕНИЕ Г

(обязательное)