**Практическое задание 1.**

**Реализовать запросы к удаленному серверу по протоколу HTTP с использованием утилит telnet, curl к следующим ресурсам:**

[**http://www.mgpu.ru/**](http://www.mgpu.ru/)

[**https://bmstu.ru/**](https://bmstu.ru/)

[**https://cbr.ru/**](https://cbr.ru/)

**Развернуть HTTP-сервер nginx и провести хостирование статистических веб-ресурсов.**

**Место выполнения задания Виртуальная машина U20-01.**

**TELNET**

- установить telnet из официальных репозиториев, в Ubuntu:

**sudo apt install telnet**

**sudo telnet –d localhost 22**

Набрав в терминале telnet localhost 22, в ответ получим приветственное сообщение от SSH сервера. Режим "Отладка" (-d) данная опция применяется, если необходимо получить подробный отчет о каждом этапе утилиты.

**sudo telnet opennet.ru 80**

Утилита отправит запрос сайту opennet.ru на порт под номером 80.

**sudo telnet india.colorado.edu 13**

**CURL**

Установка Curl в Ubuntu:

**sudo apt install curl**

**curl -v** [**http://www.mgpu.ru/**](http://www.mgpu.ru/)

Если код состояния 301 надо посмотреть чуть ниже до строчки Location - там указан новый адрес ресурса. Копируем и повторяем операцию



**curl -v** [**https://www.mgpu.ru/**](https://www.mgpu.ru/)

После этого он вывалит полотно HTML кода, надо долистать наверх, где будет основная информация. Там уже рассказать показать что есть - какой протокол используется, на каком сервере хостится, какой код состояния (должен быть 200, значит все в порядке). Ниже теория на случай если надо будет подробнее про все это рассказать.

Аналогичным образом пробиваем эти:

**curl -v** [**https://bmstu.ru/**](https://bmstu.ru/)

**curl -v** [**https://cbr.ru/**](https://cbr.ru/)

ТЕОРИЯ

Просто `curl [http://mgpu.ru`](http://mgpu.ru%60/) выведет в stdout тело ответа, stderr будет пустым, а мы хотим посмотреть в содержимое запроса. Для этого можно указать опцию `- v`, тогда много дополнительной информации будет выведено в stderr

Тут мы видим 302 в ответе, это похоже на 301, но 302 говорит о том, что для данного запроса был найден новый путь, куда надо проследовать и возможно повторный запрос даст 302 на другую страницу (такое бывает). В ответе видно, что сервер решил, что мы англоязычный клиент и хотим читать английскую версию сайта вышки. Ну действительно, давайте сделаем запрос туда и получим свой долгожданный 200.

Некоторые коды состояния протокола:

* 200 OK — успешный запрос. Если клиентом были запрошены какие-либо данные, то они находятся в заголовке и/или теле сообщения.
* 201 Created — в результате успешного выполнения запроса был создан новый ресурс.
* 202 Accepted — запрос был принят на обработку, но она не завершена.
* 301 Moved Permanently — запрошенный документ был окончательно перенесен на новый URI, указанный в поле Location заголовка.
* 302 Found, 302 Moved Temporarily — запрошенный документ временно доступен по другому URI, указанному в заголовке в поле Location.
* 400 Bad Request — сервер обнаружил в запросе клиента синтаксическую ошибку.
* 401 Unauthorized — для доступа к запрашиваемому ресурсу требуется аутентификация.
* 402 Payment Required — предполагается использовать в будущем. В настоящий момент не используется. Этот код предусмотрен для платных пользовательских сервисов, а не для хостинговых компаний. Имеется в виду, что эта ошибка не будет выдана хостинговым провайдером в случае просроченной оплаты его услуг.
* 403 Forbidden — сервер понял запрос, но он отказывается его выполнять из-за ограничений в доступе для клиента к указанному ресурсу. Иными словами, клиент не уполномочен совершать операции с запрошенным ресурсом.
* 404 Not Found — самая распространённая ошибка при пользовании Интернетом, основная причина — ошибка в написании адреса Web-страницы.

HTTP 0.9

• The Original HTTP as dened in 1991

• Клиент-сервер, запрос-ответ

• Представление данных - ASCII

• Запрос - одна строка (GET ...)

• Ответ - гипертекстовый документ (HTML)

• Транспорт - TCP, соединение закрывается после каждого запроса

HTTP/1.0

• RFC 1945 (1996)

— Документирует best practices, не является формальной спецификацией

— Фокус на простоте реализации

• Методы GET, HEAD, POST

• Запрос и ответ содержат версию протокола

• Запрос и ответ могут содержать заголовки (дополнительные метаданные)

• Ответ включает статус обработки запроса

• Тело ответа может содержать не только гипертекст

• Content encoding, character set support, multi-part types, authorization, caching...

• Соединение по-прежнему закрывается после каждого запроса

HTTP/1.1

• Поддержка виртуальных хостов, позволяющих серверам обслуживать несколько доменов на одном IP адресе

• Поддержка постоянных соединений, позволяющих браузерам делать несколько запросов в рамках одного TCP соединения

• Поддержка кеширования для экономии трафика и увеличения скорости

• Пересылка данных частями (chunked), когда не известен размер итоговой страницы

• Согласования, такие как: язык, кодировка или тип данных, позволяющих клиенту и серверу договориться об особенностях требуемого ответа

HTTP/2

• Под потоком понимается двунаправленная передача информации внутри установленного TCP соединения

• Передача осуществляется посредством одного TCP соединения с любым количеством параллельных потоков

• Такой протокол называется мультплексированным. Несколько параллельных запросов могут использовать одно соединение

• Это обеспечивает возможность для разной приоритезации передаваемых данных

• Также, это позволяет серверу самостоятельно инициировать передачу данных

• Вместо текстовых данных, протокол использует бинарный формат передачи данных, что позволяет увеличить производительность и безопасность

• Заголовки запросов и ответов сжимаются принудительно

Методы HTTP

• GET

— запрос представления ресурса с данным URI

— только чтение, не меняет состояние сервера

• POST

— создание нового ресурса (с новым URI!), отправка формы, запуск операции

— необходимые данные передаются в теле запроса

• PUT

— запись представления ресурса с данным URI

— в теле запроса передается представление ресурса

• DELETE

— удаление ресурса с данным URI

**NGNIX**

Сначала поставим `nginx` на вашу операционную систему.

**sudo apt update && sudo apt install -y nginx**

После установки должна появиться папка `/etc/nginx`, в которой мы и будем создавать конфигурации. В `/etc/nginx` есть папки `sites-available` и `sites-enabled`.

Это одни и те же конфигурации, только в `sites-available` находятся все доступные пользовательские конфигурации, а в `sites-enabled` добавляются ссылки на конфигурации, которые надо включить в данный момент у сервера. Там уже лежитконфигурация `default` и она включена:

**ls -l /etc/nginx/sites-enabled/**

*total 0*

*lrwxrwxrwx 1 root root 34 May 31 15:33 default -> /etc/nginx/sites-available/default*

**ls -l /etc/nginx/sites-available/**

*total 4*

*-rw-r--r-- 1 root root 2072 May 31 15:37 default*

(смотрим в браузере по адресу localhost установился ли ngnix)

Таким образом, мы будем писать конфигурации в `sites-available`, а потом

добавлять ссылки на них в `sites-enabled`.

Чтобы не конфликтовать с `default`, давайте сразу удалим его из `sitesenabled`:

**sudo rm /etc/nginx/sites-enabled/default**

Для начала создадим простую статику, которую можно будет раздать — html

файл и картинку. Это принято делать в `/var/www/your-website.com`:

**sudo mkdir -p /var/www/simple\_static**

**sudo chown НАЗВАНИЕ\_МАШИНЫ /var/www/simple\_static**

**cd /var/www/simple\_static**

**curl** [**https://www.google.com/images/branding/googlelogo/ HYPERLINK "https://www.google.com/images/branding/googlelogo/2x/googlelogo\_color\_272x92dp.png" HYPERLINK "https://www.google.com/images/branding/googlelogo/2x/googlelogo\_color\_272x92dp.png" HYPERLINK "https://www.google.com/images/branding/googlelogo/2x/googlelogo\_color\_272x92dp.png"2 HYPERLINK "https://www.google.com/images/branding/googlelogo/2x/googlelogo\_color\_272x92dp.png" HYPERLINK "https://www.google.com/images/branding/googlelogo/2x/googlelogo\_color\_272x92dp.png" HYPERLINK "https://www.google.com/images/branding/googlelogo/2x/googlelogo\_color\_272x92dp.png"x/googlelogo\_color\_ HYPERLINK "https://www.google.com/images/branding/googlelogo/2x/googlelogo\_color\_272x92dp.png" HYPERLINK "https://www.google.com/images/branding/googlelogo/2x/googlelogo\_color\_272x92dp.png" HYPERLINK "https://www.google.com/images/branding/googlelogo/2x/googlelogo\_color\_272x92dp.png"272 HYPERLINK "https://www.google.com/images/branding/googlelogo/2x/googlelogo\_color\_272x92dp.png" HYPERLINK "https://www.google.com/images/branding/googlelogo/2x/googlelogo\_color\_272x92dp.png" HYPERLINK "https://www.google.com/images/branding/googlelogo/2x/googlelogo\_color\_272x92dp.png"x HYPERLINK "https://www.google.com/images/branding/googlelogo/2x/googlelogo\_color\_272x92dp.png" HYPERLINK "https://www.google.com/images/branding/googlelogo/2x/googlelogo\_color\_272x92dp.png" HYPERLINK "https://www.google.com/images/branding/googlelogo/2x/googlelogo\_color\_272x92dp.png"92 HYPERLINK "https://www.google.com/images/branding/googlelogo/2x/googlelogo\_color\_272x92dp.png" HYPERLINK "https://www.google.com/images/branding/googlelogo/2x/googlelogo\_color\_272x92dp.png" HYPERLINK "https://www.google.com/images/branding/googlelogo/2x/googlelogo\_color\_272x92dp.png"dp.png**](https://www.google.com/images/branding/googlelogo/2x/googlelogo_color_272x92dp.png)**> google.png**

**printf "<body>This is our first html file</body>\n" > index.html**

Теперь в папке `/var/www/simple\_static` есть два файла:

**ls**

*google.png index.html*

**sudo service nginx stop**

**sudo nano /etc/nginx/sites-available/simple\_static.conf**

Пишем туда и сохраняем:

server {

listen 80 default\_server;

server\_name \_;

root /var/www/simple\_static;

}

**cd /etc/nginx/**

**sudo ln sites-available/simple\_static.conf sites-enabled/simple\_static.conf**

**sudo nginx -t**

**sudo service nginx start**

**curl -v** [**http://localhost: HYPERLINK "http://localhost/" HYPERLINK "http://localhost/" HYPERLINK "http://localhost/"80**](http://localhost/)

(потом смотрим в браузере по адресу localhost, там должно быть This is our first html file)