Лабораторная работа №6
Веб-сервер Nginx. Прокси сервер Squid.

1. Теоретическая часть

1.1. Основы языка HTML

HTML (от англ. HyperText Markup Language — «язык гипертекстовой разметки») — стандартизированный язык гипертекстовой разметки документов для просмотра веб-страниц в браузере.

HTML создавался как язык для обмена научной и технической документацией, пригодный для использования людьми, не являющимися специалистами в области вёрстки.

Текстовые документы, содержащие разметку на языке HTML (такие документы традиционно имеют расширение **.html** или **.htm**), обрабатываются специальными приложениями, которые отображают документ в его форматированном виде. Такие приложения, называемые «*браузерами*» или «*интернет-обозревателями*».

Любой документ на языке HTML представляет собой набор элементов, причём начало и конец каждого элемента обозначается специальными пометками — тегами.

Тег — это специальное служебное слово, заключенное в угловые скобки. Его ещё называют «элемент HTML». Если тег парный, то тегу <ТЕГ> соответствует </ТЕГ>.

Для создания простого HTML-документа достаточно следующих тегов:

Тег	Описание
<!DOCTYPE>	Объявляет тип документа и предоставляет основную информацию для браузера — его язык и версия.
<html>	Корневой элемент HTML-документа. Сообщает браузеру, что это HTML-документ. Является контейнером для всех остальных html-элементов.
<head>	Элемент-контейнер для метаданных HTML-документа, таких как **<title>**, **<meta>**, **<script>**, **<link>**, **<style>**.
<body>	Представляет тело документа (содержимое, не относящееся к метаданным документа).

Достаточно любого текстового редактора для создания HTML-страницы. Например, добавив в пустой документ следующие строки

```
<!DOCTYPE html>
<html>
  <head>
    <title>Моя первая страница</title>
  </head>
  <body>
    Всем привет!
  </body>
</html>
```

и назвав документ с расширением **.html**, его можно прочитать в отформатированном виде в любом браузере. В нём же можно посмотреть исходный код страницы на языке HTML. Впрочем, это относится ко всем WEB-страницам, получаемых из сети Интернет.

1.2. Протокол HTTP

HTTP (англ. HyperText Transfer Protocol — «протокол передачи гипертекста») — сетевой протокол прикладного уровня, предназначенный для передачи не только гипертекстовых документов в формате HTML (как было изначально), но и всех видов данных в сети Интернет и в изолированных веб-инфраструктурах.

Для передачи данных использует протокол транспортного уровня TCP и порт под номером 80.

Протокол не поддерживает шифрование и аутентификацию сервера. Поэтому при подключении к серверу по протоколу HTTP современный браузер может выдать предупреждение о том, что подключение является небезопасным. Для безопасного подключения используется протокол **HTTPS**.

Протокол работает в режиме «**Запрос — Ответ**». Веб-сервер, получив запрос, в случае, если запрашиваемый документ статический (изображение, или хранимый на диске сервера HTML-документ), считывает документ с диска и отправляет клиенту. В случае же, если документ – динамический, то веб-сервер запускает сценарий, или передает запрос следующему серверу, умеющему выполнять сценарий. Сценарий генерирует документ и веб-сервер отдает его клиенту. Пользователь в итоге получает документ (например, HTML), который отображается в браузере.

Документ может быть сгенерирован локально с помощью ***SPA (Single Page Application)***, написанном на **JavaScript**. В этом случае клиент запрашивает у сервера не документ, а данные, используя API (как правило, ***REST API***). Получив данные по API, SPA генерирует документ на стороне клиента.

Структура запроса имеет следующий вид:

```
МЕТОД РЕСУРС HTTP/1.1
```

Метод (англ. Method) — тип запроса по протоколу HTTP, состоящего из одного слова, написанного заглавными буквами.

Используются следующие методы:

Метод	Описание
GET	Служит для получения ресурса от веб-сервера. В зависимости от реализации может запрашиваться веб-страница, документ или производиться обращение к **REST API** с целью запроса данных.
POST	Служит для отправки данных на сервер, в частности, для создания нового ресурса.
PUT	Служит для изменения ресурса на сервере.
DELETE	Служит для удаления ресурса.
HEAD	Похож на метод GET, но возвращает только заголовки без тела ответа. Это может быть полезно когда нужно получить только метаинформацию о странице.

Ресурсом может быть файл, документ и многое другое, лежащее на сервере с определённым IP-адресом либо доменным именем.

Получение доступа к ресурсам по HTTP-протоколу осуществляется с помощью указателя ***URL (Uniform Resource Locator)***. URL представляет собой строку, которая позволяет указать запрашиваемый ресурс и еще ряд параметров.

URL имеет следующую структуру:

```
схема:[//[имя[:пароль]@]хост[:порт]][/путь][?параметры] [#якорь]
```

- **схема** – как правило, протокол (http, https), но не обязательно (data:, blob: и т. д.);
- **имя:пароль** – часто применяются при HTTP-авторизации;
- **хост** – доменное имя или IP-адрес, идентифицирующее веб-сервер;
- **порт** – указывается, если нестандартный (8000, 8080);
- **путь** – путь к файлу, сценарию, или «виртуальная иерархия» ресурсов, разбираемая сценарием;
- **параметры** – как правило GET-параметры;
- **якорь** – позволяет идентифицировать часть документа.

Структура ответа имеет следующий вид:

HTTP/1.1 КОД_СОСТОЯНИЯ РАСШИФРОВКА

При запросе и ответе указывается версия протокола HTTP. Часто используется версия 1.1, иногда 1.0, хотя существуют и новые версии.

Код состояния позволяет определить, успешно ли обработался запрос. Они делятся на следующие группы:

- **1XX** – информационные;
- **2XX** – успешное завершение;
- **3XX** – перенаправление;
- **4XX** – ошибки клиента;
- **5XX** – ошибки сервера.

Пример кода группы 2XX: 200 Ok – успешный результат.

Пример кода группы 3XX: 301 Moved Permanently (в заголовке Location сообщается адрес перенаправления). Служит для переадресации с http на https, с десктопной версии на мобильную (если запрос пришел с мобильного устройства) и т.д.

Пример кода группы 4XX: 404 Not Found – ресурс не найден.

1.3. Утилита NetCat

netcat (англ. net сеть + cat) — утилита Unix, позволяющая устанавливать соединения TCP и UDP, принимать оттуда данные и передавать их.

Устанавливается следующей командой:

```
sudo apt install netcat
```

В простом случае **NetCat** вызывается как:

```
nc host port
```

Это приводит к созданию TCP-подключения с указанными реквизитами и замыканием стандартного ввода на сетевой вывод и наоборот, стандартного вывода на сетевой ввод. Такая функциональность напоминает команду **cat**, что обусловило выбор имени «netcat».

С помощью данной программы возможно получить HTML-страницу по протоколу HTTP в командном интерпретаторе.

В качестве примера сделаем запрос HTML-страницы, лежащего по адресу **lib.ru/POLITOLOG/MAKIAWELLI/gosudar.txt** и содержащего текст произведения Никколо Макиавелли «Государь».

Для этого в командную строку введём следующую команду:

```
echo -e “GET http://lib.ru/POLITOLOG/MAKIAWELLI/gosudar.txt HTTP/1.0\n\n” | nc lib.ru 80
```

Утилита **nc** принимает на вход строку с запросом, которую отправляет на сервер с доменным именем **lib.ru** в порт 80. У сервера запрашивается ресурс, лежащего по пути **POLITOLOG/MAKIAWELLI/gosudar.txt**.

В случае успеха будет получен заголовок, имеющий вид

```
HTTP/1.1 200 OK
Server: nginx/1.21.5
Date: Sun, 13 Oct 2024 18:48:42 GMT
Content-Type: text/html; charset=koi8-r
Connection: close
```

```
Last-Modified: Sat, 19 Dec 2009 20:06:19 GMT
```

Далее идёт сам запрашиваемый HTML-документ. Однако стоит иметь в виду, что протокол HTTP передаёт текст в кодировке, которая может неправильно отображать кириллицу.

Чтобы мы могли прочитать произведение итальянского мыслителя, необходимо выполнить конвертацию кодировки текста.

```
echo -e "GET http://lib.ru/POLITOLOG/MAKIAWELLI/gosudar.txt HTTP/1.0\n\n" | nc lib.ru 80 | iconv -f koi8-r -t UTF-8
```

Современные операционные системы поддерживают кодировку UTF-8, которая правильно отображает кириллицу. Чтобы узнать, из какой кодировки необходимо провести конвертацию, можно посмотреть в заголовке ответа в переменной **charset** или с помощью утилиты **uchardet**.

1.4. Веб-сервер

Всемирная паутина (англ. World Wide Web) — распределённая система, предоставляющая доступ к связанным между собой документам, расположенным на различных компьютерах, подключённых к сети Интернет. Для обозначения Всемирной паутины также используют слово **веб (англ. web «паутина»)** и аббревиатуру **WWW**.

Веб-сервер — сервер, принимающий HTTP-запросы от клиентов, обычно веб-браузеров, и выдающий им HTTP-ответы, как правило, вместе с HTML-страницей, изображением, файлом, медиа-потоком или другими данными.

Наиболее популярным в мире веб-сервером является **Apache HTTP Server**, также известный как **Apache**. Это веб-сервер с открытым исходным кодом, разработанный Apache Software Foundation, при этом является исторически первым. Он имеет модульную структуру, а на каждое соединение выделяется отдельный процесс из-за чего его так легко повесить, если клиенты очень медленные.

Веб-сервер **NGINX** разработан для устранения ограничений производительности Apache. Он так же с открытым исходным кодом, при этом выполняет функции обратного прокси, балансировщика нагрузки, почтового прокси и HTTP-кэша. Это популярный выбор для сайтов с высокой посещаемостью.

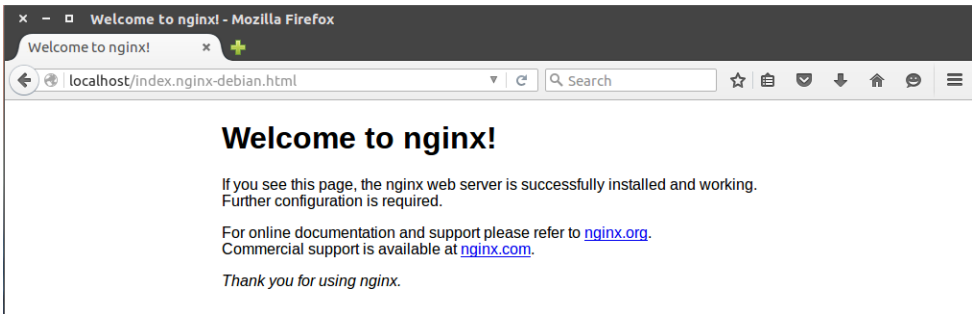
Для его установки применяется команда

```
sudo apt install nginx
```

После того как установка сервера Nginx будет завершена добавим программу в автозагрузку, чтобы она запускалась автоматически:

```
sudo systemctl enable nginx
```

Уже после установки можно проверить работу веб-сервера. Если в браузере ввести в адресной строке **localhost** (локальный IP-адрес сервера), то должна показываться приветственная страница сервера.



У веб-сервера главный конфигурационный файл лежит по пути **/etc/nginx/nginx.conf**.

Файл разделён на секции, которые имеют вид:

```
глобальные опции
events {}
http{
server {
location{}
}
server {}
}
mail {}
```

Секция	Описание
глобальные опции	Отвечают за работу всей программы.
events	Содержит настройки для работы с сетью.
http	Содержит настройки веб-сервера. Должна содержать секцию **server** для тонкой настройки каждого сайта.
server	Содержит настройку каждого размещённого на веб-сервере сайта. Для каждого сайта необходимо создавать отдельную секцию.
location	Может находиться только внутри секции **server** и содержит настройки только для определённого запроса.
mail	Содержит настройки почтового прокси.

По умолчанию конфигурационный файл выглядит следующим образом:

```
user www-data;
worker_processes auto;
pid /run/nginx.pid;
include /etc/nginx/modules-enabled/*.conf;
events {
	worker_connections 1024;
}
http {
	sendfile on;
	tcp_nopush on;
	tcp_nodelay on;
	keepalive_timeout 65;
	types_hash_max_size 2048;
	include /etc/nginx/mime.types;
	default_type application/octet-stream;
	ssl_protocols TLSv1 TLSv1.1 TLSv1.2 TLSv1.3;
	ssl_prefer_server_ciphers on;
	access_log /var/log/nginx/access.log;
	error_log /var/log/nginx/error.log;
	gzip on;
	include /etc/nginx/conf.d/*.conf;
	include /etc/nginx/sites-enabled/*;
```

```
}
```

Здесь использованы следующие директивы:

user – пользователь, от имени которого работает nginx, здесь это www-data;

worker_processes – количество процессов сервера, значение выставляется равным количеству ядер процессора, auto – сервер определит автоматически;

pid – файл, внутри которого хранится идентификатор запущенного главного процесса (PID);

include – подключаемый файл или файлы конфигурации;

events – блок директив, определяющих работу с сетевыми соединениями;

worker_connections – максимальное количество одновременных соединений;

http – блок директив http сервера;

sendfile – метод отправки данных, включаем значением on;

tcp_nopush и **tcp_nodelay** – параметры, положительно влияющие на производительность, оставляем значение on;

keepalive_timeout – время ожидания keepalive соединения до его разрыва со стороны сервера;

types_hash_max_size – регламентирует максимальный размер хэш таблиц типов;

default_type – указывает тип MIME ответа по умолчанию;

ssl_protocols – включает указанные протоколы;

ssl_prefer_server_ciphers – указывает, что серверное шифрование; предпочтительнее клиентского, при использовании SSLv3 и TLS протоколов;

access_log – задает путь к файлу лога доступа, при выставлении значения в off, запись в журнал доступа будет отключена;

error_log – путь к журналу регистрации ошибок;

gzip – при помощи этой директивы можно включать или отключать сжатие.

Конфигурации сайтов хранятся в папке **/etc/nginx/sites-enabled**. С помощью директивы **include /etc/nginx/sites-enabled/*** конфигурация загружается в главный конфигурационный файл. Поэтому для начала все настройки можно проводить в одном файле.

Попробуем разместить собственную HTML-страницу на веб-сервер, чтобы по запросу клиента его выдавать.

Сделаем backup конфигурационного файла:

```
sudo mv /etc/nginx/nginx.conf /etc/nginx/nginx.conf.orig
```

Затем в пустой файл **/etc/nginx/nginx.conf** добавим следующие строки:

```
user www-data;
worker_processes auto;
pid /run/nginx.pid;

events {
        worker_connections 1024;
        use epoll;
        multi_accept on;
}
http {
        sendfile on;
        tcp_nopush on;
        tcp_nodelay on;
        keepalive_timeout 60;

        access_log /var/www/site/access.log;
        error_log /var/www/site/error.log;

server {
        listen 80 default_server;
        root /var/www/site;
        index site.html;
        server_name site.gad.miet.stu;
}
}
```

listen 80 - указывает, что нужно ожидать подключения на порту 80, может также содержать опцию default-server, которая означает, что этот домен будет открывается если домен не был задан в запросе.

root /var/www/site - директория, в которой находятся файлы сайта.

index site.html - страница, которая будет открываться по умолчанию.

server_name - доменное имя сайта.

Для проверки на ошибки используется команда

```
nginx -t
```

Если изменялись незначительные параметры можно использовать команду **systemctl reload**, тогда будет просто обновлена конфигурация без перезагрузки. Если же изменяли глобальные опции, нужно перезагрузить программу полностью с помощью **restart**.

Если всё настроено верно, то в веб-браузере возможно получить свою страницу, введя в адресную строку локальный IP-адрес сервера.

Страницу можно получить по доменному имени, добавив соответствующую запись в базу данных DNS-сервера.

Настройка службы PHP-FPM

Для запуска программ, написанных на PHP необходимо наличие службы PHP-FPM. FPM расшифровывается как Fastcgi Process Manager, менеджер процессов FastCGI. PHP-FPM запускается как отдельный процесс и взаимодействует с веб-сервером через порт 9000 или сокетный файл. Является альтернативной реализацией PHP FastCGI с несколькими дополнительными возможностями, обычно используемыми для высоконагруженных сайтов.

Для её установки на сервере выполним команду:

```
sudo apt install php8.1-fpm
```

Для подключения службы к веб-странице необходимо внутри секции server указать секцию location с адресом расположения файла сокета для общения с php-fpm

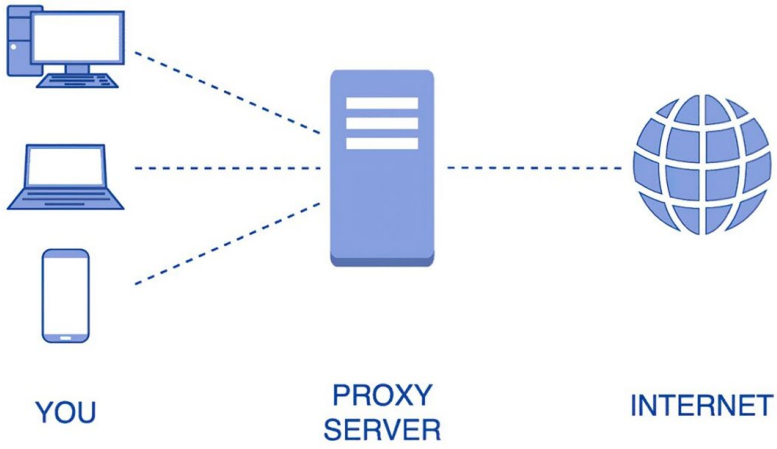
```
location ~ \.php$ {
        fastcgi_pass unix:/run/php/php8.1-fpm-example-com.sock;
        include fastcgi.conf;
    }
```

1.1. Прокси-сервер

Прокси-сервер (от англ. proxy — представитель, уполномоченный; часто просто прокси, сервер-посредник) — это программные и аппаратные средства в компьютерных сетях, выполняющие роль посредника между пользователем и целевым сервером.

Прокси-сервер принимает и пересылает запросы на подключение, а затем возвращает данные для этих запросов. Он использует анонимный сетевой идентификатор вместо фактического IP-адреса клиента (то есть скрывает IP-адрес клиента), так что фактический IP-адрес клиента не может быть раскрыт.

Сначала клиент подключается к прокси-серверу и запрашивает какой-либо ресурс (например e-mail), расположенный на другом сервере. Затем прокси-сервер либо подключается к указанному серверу, используя свой IP-адрес, и получает ресурс у него, либо возвращает ресурс из собственного кэша (если прокси-сервер является кеширующим).



При этом стоит иметь в виду, что прокси-сервер анализирует пакеты на прикладном уровне модели OSI, в то время как Firewall — на уровнях ниже. Поэтому для большей эффективности применяют связку Proxy-Firewall, которые в полной мере предоставляют возможности для контроля трафика.

В качестве функции прокси-сервера указана подмена локального IP-адреса своим собственным. Это похоже на технологию NAT, которая так же сопоставляет множеству локальных адресов одному IP-адресу для выхода в сеть. Однако эта технология, которая поддерживается маршрутизаторами, работает только на сетевом уровне модели OSI и не занимается анализом трафика.

В качестве правил, которыми руководствуется прокси-сервер, могут выступать условия пакетной фильтрации. Правила могут быть достаточно сложными, например в рабочие часы блокируется доступ к тем или иным узлам и/или прилопсжениям, а доступ к другим узлам разрешается только определенным пользователям, причем для FTP-серверов пользователям разрешается делать лишь загрузку, а выгрузка запрещается. Прокси-серверы могут также фильтровать почтовые сообщения по типу пересылаемого файла (например, запретить получение сообщений формата MP3) и по их контенту.

К разным пользователям могут применяться разные правила фильтрации, поэтому часто на прокси-серверы возлагается задача аутентификации пользователей.

Наиболее популярным прокси-сервером является ***Squid (англ. squid — «кальмар»)*** — программный пакет, реализующий функцию кэширующего прокси-сервера для протоколов HTTP, FTP, Gopher и (в случае соответствующих настроек) HTTPS.

Для его установки применяется команда

```
sudo apt install squid
```

Конфигурационный файл лежит по пути **/etc/squid/squid.conf**. Проанализируем его структуру.

В файле настраиваются правила доступа клиентов к прокси-серверов. Это делается с помощью списков управления доступом **ACL (Access Control List)**.

Общий формат списка управления доступом:

```
acl имя_списка параметр содержимое
```

Параметром может быть одно из следующих:

- **src addr1-addr2/mask** — диапазон IP-адресов источников запросов;
- **dstdomain [-n] .foo.com ...** — домен из URL в запросе;
- **dstdom_regex [-n] [-i] \.foo\.com ...** — регулярное выражение для домена из URL запроса;
- **url_regex [-i] ^http:// ...** — регулярное выражение для URL запроса:
- **time [day-abbrevs: MTWHFAS] [h1:m1-h2:m2]** — установка времени.

Для настроенного списка с определённым названием отдельно устанавливается правило, имеющего вид:

```
http_access инструкция имя_списка
```

В качестве инструкции применяются **allow** (разрешить) или **deny** (запретить).

Конфигурация обрабатывается построчно как при работе правил Firewall. Если поставить запрещающую строку выше разрешающей, то работа будет неправильной.

Приведём пример конфигурационного файла:

```
acl eth port 80                          #Добавить в список номер порта
acl localnet src 172.102.0.100           #Добавить в список адрес
acl localnet src 172.1.0.1-172.1.0.255   #Добавить в список диапазон адресов
acl localnet src 172.16.0.0/12           #Добавить в список диапазон адресов
acl localnet srcdomain .mpsu.stu         #Добавить в список адрес домена от которого хотим подключиться

acl guestnet dst 172.16.0.11             #Добавить в список адрес машины к которой пытаемся осуществить доступ
acl guestnet dstdomain .temp.ru          #Добавить в список адрес домена к которому хотим подключиться

http_access allow localnet     #Разрешить доступ к списку с именем localnet
http_access deny all           #Разрешить доступ из остальной сети
```

Проверка синтаксиса конфигурационного файла производится командой

```
squid -k check
```

После внесения изменений необходимо перезагрузить службу squid силами **systemctl**

```
sudo systemctl restart squid
```

Просмотр конфигураций, используемой прокси:

```
squid -k parse
```

В качестве примера запретим доступ к поисковой системе **ya.ru**.

Создадим файл **/etc/squid/block.txt** и запишем в него строку

```
.ya.ru
```

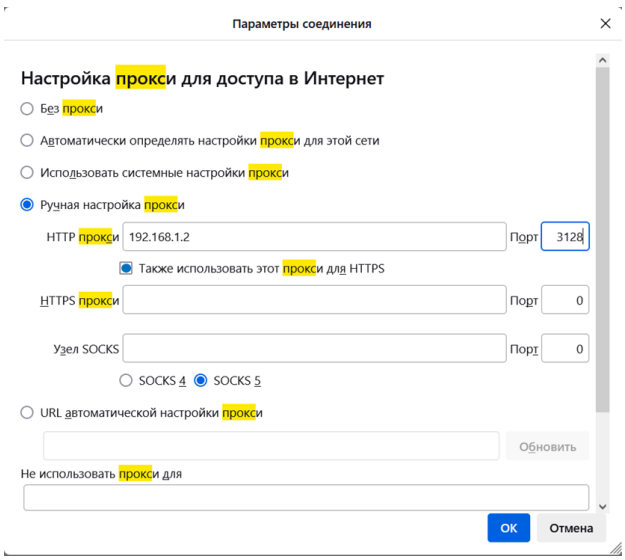
Затем в конфигурационный файл запишем строки:

```
http_port 3128

acl block-site dstdomain ”/etc/squid/block.txt”
http_access deny block-site
```

Примечание: ***http_port 3128*** *— порт, используемый клиентами для запросов к прокси.*

После перезагрузки Squid проверим работу прокси. Для этого зайдём в браузер и в его настройках укажем данные для подключения к прокси-серверу. На картинке показана настройка для Mozilla FireFox.



2. Практическая часть

2.1. Задание 1

2.1.1. Выполните в терминале HTTP-запрос страницы с информацией про сеть TCP/IP, лежащего по адресу **lib.ru/unixhelp/network.txt**. HTTP-ответ сохраните в домашней директории под именем **network.html**.

2.1.2. Откройте с помощью браузера файл **network.html** и убедитесь, что текст правильно отформатирован.

2.2. Задание 2

2.2.1. Создайте в директории **/var/www/site/** HTML-документ.

2.2.2. Установите веб-сервер **Nginx** и настройте его работу так, чтобы он при HTTP-запросе выдавал созданный вами HTML-документ.

2.2.3. Добавьте созданную страницу в список записей DNS и откройте её с машины клиента, используя доменное имя **site.«Ваши инициалы».miet.stu**

2.3. Задание 3

2.3.1. Установите на сервер утилиту PHP-FMP.

2.3.2. Создайте еще один сайт, добавив новую секцию server. Установите параметры, необходимые для обработки PHP скриптов. Установите название сайта **site1.«Ваши инициалы».miet.stu**

2.3.3. Добавьте веб-страницу со следующим содержимым:

```
<!DOCTYPE html>
<html>
   <body>
      <p>
         Hello MIET from HTML
      </p>
      <p>
         <?php
            echo "Hello MIET from PHP";
         ?>
      </p>
      <p>
         <script>
            let div = document.createElement('div');
```

```
                    div.className = "alert";
                    div.innerHTML = "Hello MIET from JS";
                    document.body.append(div);
                </script>
            </p>
        </body>
    </html>
```

Перегрузите сервер и убедитесь, что обе созданные вами страницы отображаются корректно. Используя утилиту *curl*, запросите страницу site1 из командной строки. Сравните с выводом в браузер и объясните результат.

2.3. Задание 4

2.4.1. Установите **Squid** на сервер.

2.4.2. Настройте прокси-сервер таким образом, чтобы был ограничен доступ к сайту **vk.com** в рабочее время, а остальные сайты запускались свободно.

2.4.3. Убедитесь, что из-под клиента имеется доступ к созданному вами сайту. Запретите клиенту подключение к сайту с помощью прокси-сервера.

Контрольные вопросы

1. Что такое HTML? Какие теги HTML вам известны?

2. Что такое HTTP? Каким образом браузер получает WEB-страницу?

3. Для чего нужен межсетевой экран?

4. Зачем нужен прокси-сервер?

5. В чём разница между работой NAT и прокси-сервером?