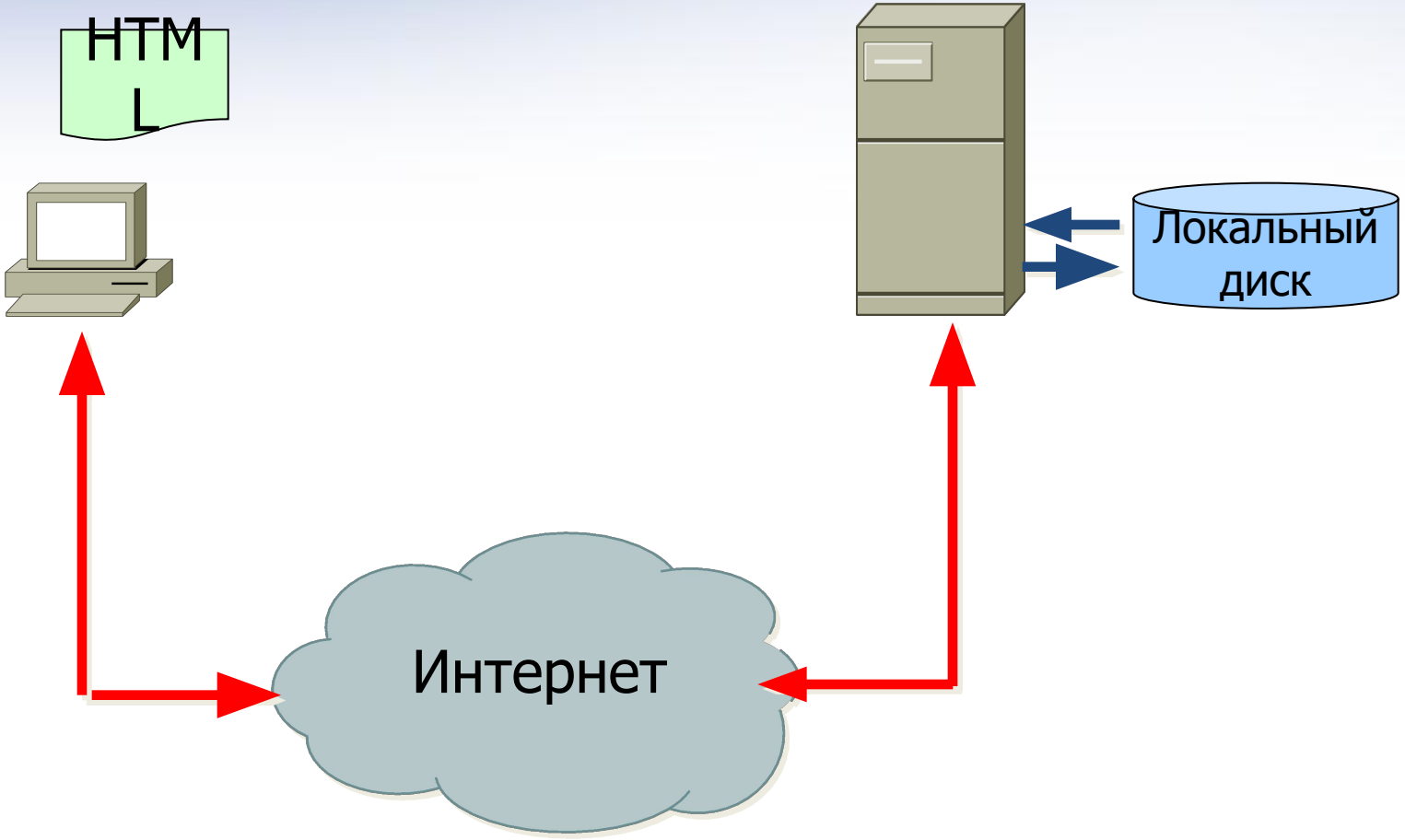
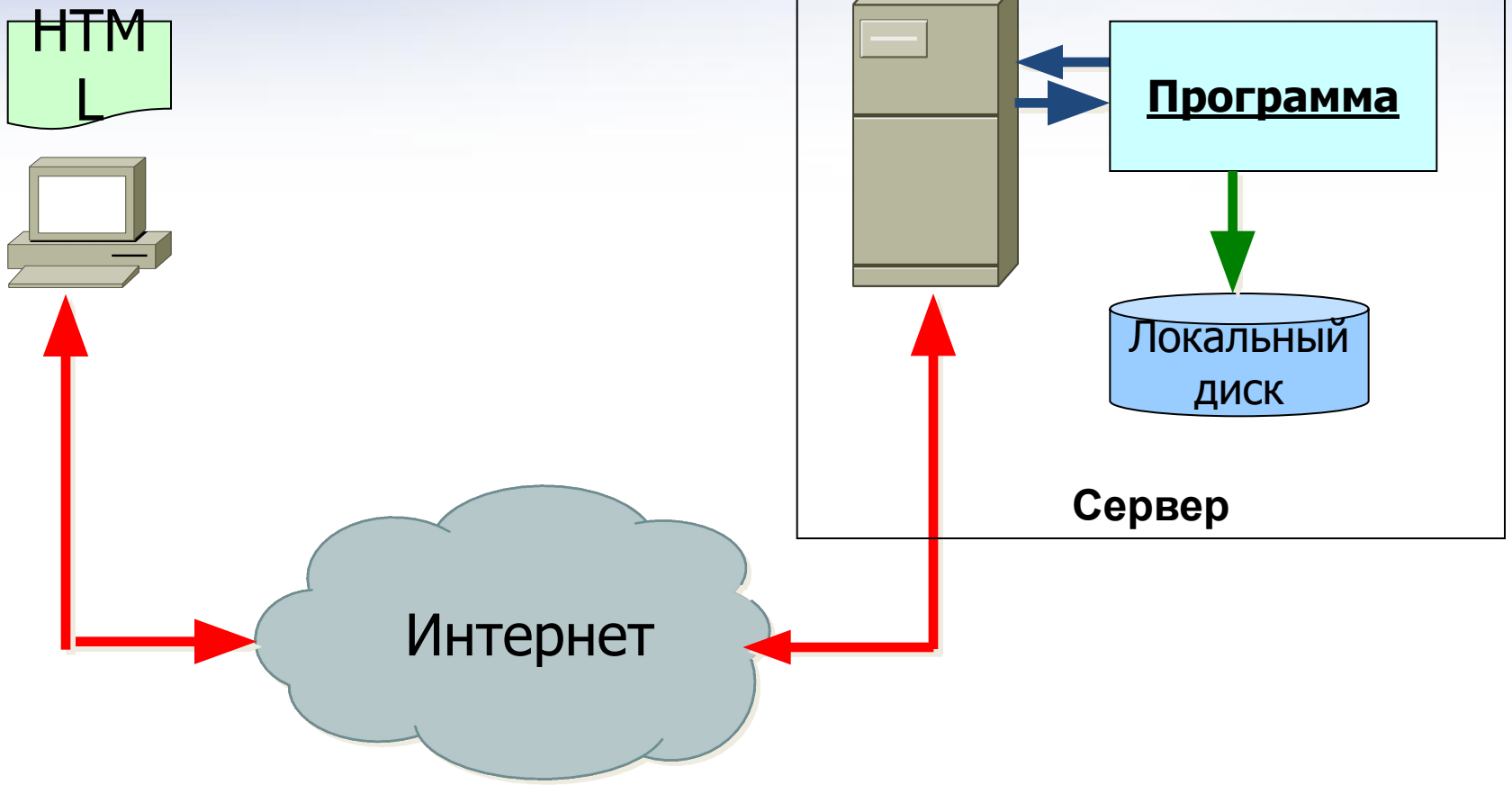
# *Common Gateway Interface (CGI)*

# Обычный доступ



# Обычный доступ … почти обычный



# *Common Gateway Interface*

***Common Gateway Interface*** **(«общий интерфейс шлюза»)** — стандарт **интерфейса**, используемого для связи внешней **программы** с **веб-сервером**.

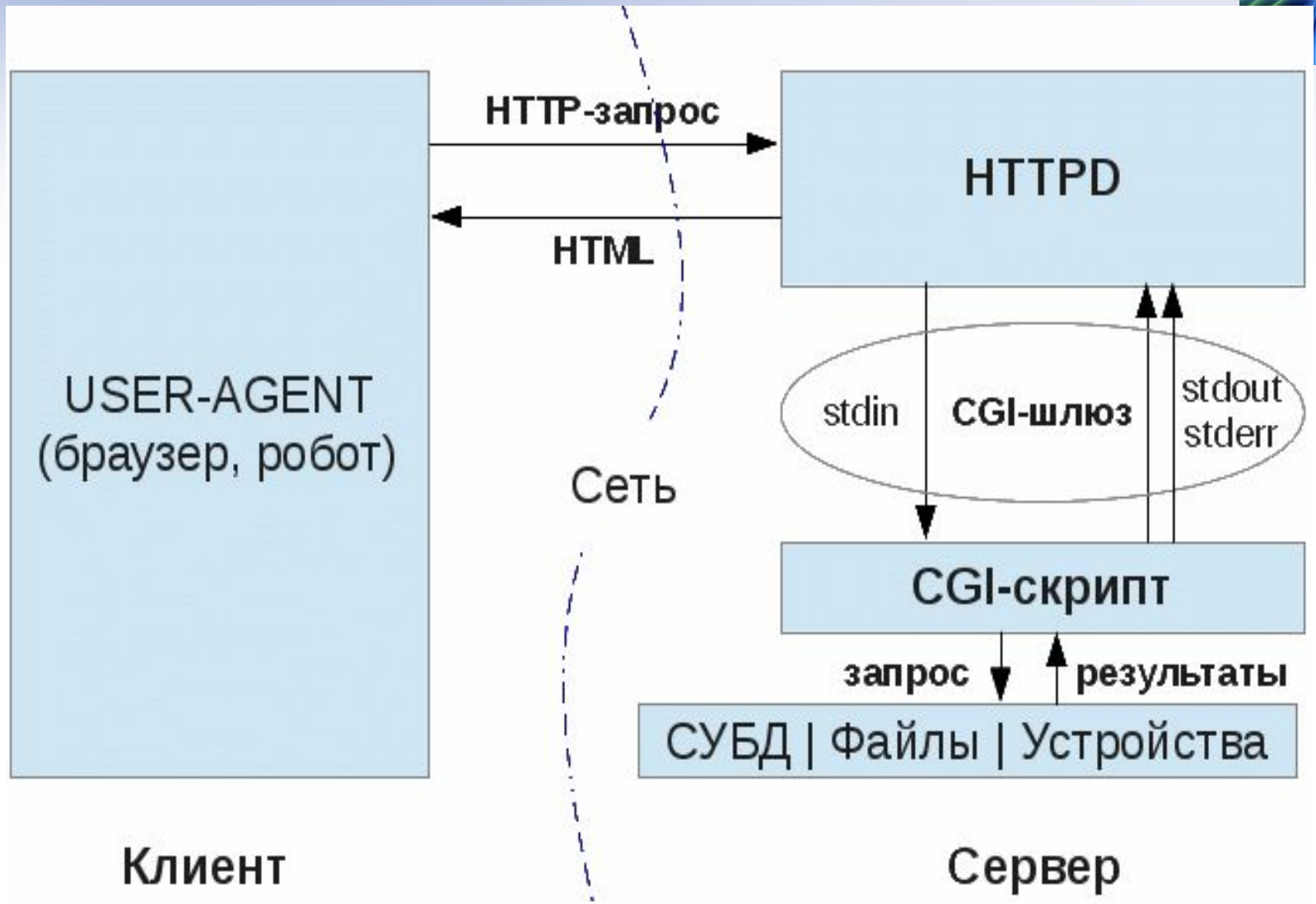
**Программу**, которая работает по такому интерфейсу совместно с **веб-сервером**, принято называть **шлюзом**, хотя многие предпочитают названия «**скрипт**» (сценарий) или «**CGI-программа**».

Сам **интерфейс** разработан таким образом, чтобы можно было использовать любой язык программирования, который может работать со стандартными устройствами ввода/вывода.

Все **скрипты**, как правило, помещают в каталог cgi (или cgi-bin) сервера, но это необязательно: скрипт может располагаться где угодно.

CGI является одним из наиболее распространённых средств создания **динамических веб-страниц**.

# Common Gateway Interface



# CGI-программа (Скрипт)

**CGI-программы** часто написаны на каком-нибудь интерпретируемом языке (PHP, Perl, Bash, Pyton и др.)
Также можно написать программу на компилируемом языке, например **C**, который и будет обрабатывать данные пользователя. Приведем пример CGI-программы на C:

```
#include <time.h> // Нужна для инициализации функции rand()
#include <stdio.h> // Включаем поддержку функций ввода/вывода
#include <stdlib.h> // А это — для поддержки функции rand()
// Главная функция. Именно она и запускается при старте сценария.
void main(void) {
// инициализируем генератор случайных чисел
int Num; time_t t; srand(time(&t));
// в Num записывается случайное число от 0 до 9
Num = rand()%10;
// далее выводим заголовки ответа. Тип — html-документ
printf("Content-type: text/html\n");
// запрет кэширования
printf("Pragma: no-cache\n");
// пустой заголовок
printf("\n");
// выводим текст документа — его мы увидим в браузере
printf("<html><body>");
printf("<h1>Здравствуйте!</h1>");
printf("Случайное число в диапазоне 0-9: %d",Num);
printf("</body></html>");
}
```

# *Веб-сервер (web-server)*

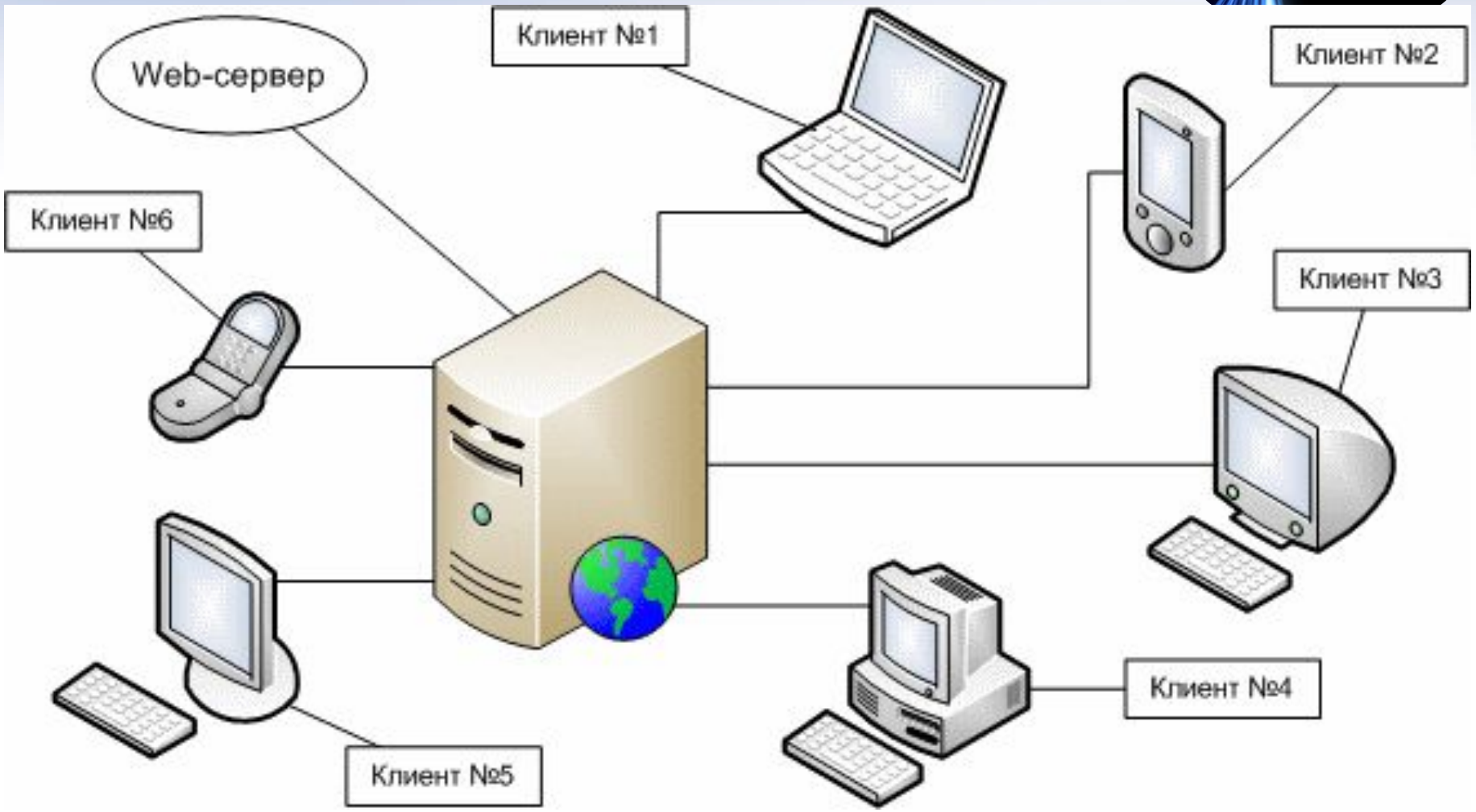
***Веб-сервер*** - это сервер, принимающий **HTTP-запросы** от клиентов, обычно **веб-браузеров**, и выдающий им **HTTP-ответы**, как правило, вместе с **HTML-страницей**, изображением, файлом, медиа-потоком или другими данными.

**Веб-сервером** называют как **программное обеспечение**, выполняющее функции веб-сервера, так и непосредственно компьютер, на котором это программное обеспечение работает.

**Клиент**, которым обычно является **веб-браузер**, передаёт веб-серверу запросы на получение ресурсов, обозначенных **URL-адресами**.

Ресурсы — это **HTML-страницы**, изображения, файлы, медиа-потоки или другие данные, которые необходимы клиенту. В ответ **веб-сервер** передаёт клиенту запрошенные данные. Этот обмен происходит по протоколу **HTTP**.

# Веб-сервер (web-server)



# Веб-сервер (web-server)

## Дополнительные функции

- автоматизация работы веб страниц;
- ведение журнала обращений пользователей к ресурсам;
- аутентификация и авторизация пользователей;
- поддержка динамически генерируемых страниц;
- поддержка HTTPS для защищённых соединений с клиентами.

## Программное обеспечение

- Apache — свободный веб-сервер, наиболее часто используемый в UNIX-подобных операционных системах;
- IIS от компании Microsoft, распространяемый с серверными ОС семейства Windows
- nginx — свободный веб-сервер, разрабатываемый Игорем Сысоевым с 2002 года и пользующийся большой популярностью на крупных сайтах
- lighttpd — свободный веб-сервер.
- Google Web Server — веб-сервер, основанный на Apache и доработанный компанией Google.
- Resin — свободный веб-сервер приложений.

# *Динамическая веб-страница*

**Динамическая веб-страница** — веб-страница, сгенерированная программно, в отличие от статичной страницы, которая является просто файлом, лежащим на сервере.

Сервер генерирует **HTML** код динамической страницы для обработки браузером или другим агентом пользователя.

Динамические страницы обычно обрабатывают и выводят информацию из базы данных.

Наиболее популярные на данный момент технологии для генерации динамических страниц:

- PHP — Для серверов Apache и других под управлением GNU/Linux, других UNIX-подобных, и прочих ОС.
- JSP и Java Servlet — Для серверов Apache, JBoss, Tomcat под управлением различных ОС.
- ASP.NET — Для Microsoft Windows серверов под управлением IIS.

# FastCGI

Интерфейс **FastCGI** — клиент-серверный протокол взаимодействия веб-сервера и приложения, дальнейшее развитие технологии **CGI**.
По сравнению с CGI является более производительным и безопасным.

**FastCGI** снимает множество ограничений **CGI-программ**.
**Недостаток CGI-программ** в том, что они должны быть перезапущены веб-сервером при каждом запросе, что приводит к понижению производительности. **FastCGI**, вместо того чтобы создавать новые процессы для каждого нового запроса, использует постоянно запущенные процессы для обработки множества запросов. Это позволяет экономить время.

В то время как **CGI-программы** взаимодействуют с сервером через STDIN и STDOUT запущенного **CGI-процесса**, FastCGI-процессы используют Unix Domain Sockets или TCP/IP для связи с сервером.
Это даёт следующее преимущество над обычными CGI-программами: FastCGI-программы могут быть запущены не только на этом же сервере, но и где угодно в сети. Также возможна обработка запросов несколькими FastCGI-процессами, работающими параллельно.

***FastCGI** может быть использован в любом языке, поддерживающем сокеты.*

# Веб-серверы, поддерживающие FastCGI

- Abyss Web Server
- Apache HTTP-сервер *(частично)*
- Cherokee HTTP Server
- Hiawatha webserver
- Lighttpd
- LiteSpeed Web Server
- Microsoft IIS
- MyServer
- nginx
- Open Market Web Server
- Roxen webserver
- Sun Java System Web Server (и предшественники)
- WebSTAR
- Yaws
- Zeus

# *HyperText Transfer Protocol (HTTP)*

# Протокол HTTP

**HTTP** (англ. *HyperText Transfer Protocol* — «протокол передачи гипертекста») — протокол прикладного уровня передачи данных (изначально — в виде гипертекстовых документов).

Основой **HTTP** является технология «клиент-сервер», то есть предполагается существование **потребителей** (**клиентов**), которые инициируют соединение и посылают запрос, и **поставщиков** (**серверов**), которые ожидают соединения для получения запроса, производят необходимые действия и возвращают обратно сообщение с результатом.

Протокол **HTTP** является протоколом, не запоминающим состояния, т.е. каждое подключение закрывается сразу же после того, как сервер отправит ответ клиенту. Таким образом, Web-сервер не запоминает никаких сведений о предыдущих запросах.

Основным объектом манипуляции в **HTTP** является ресурс, на который указывает **URI** (англ. *Uniform Resource Identifier*) в запросе клиента.

Обычно такими ресурсами являются хранящиеся на сервере файлы, но ими могут быть логические объекты или что-то абстрактное.

# Спецификация MIME

Протоколы приложений могут обмениваться только текстовой информацией. Для обеспечения возможности передачи *двоичных файлов* по протоколу **HTTP** используется спецификация **MIME (Multipurpose Internet Mail Extension)**.

Согласно спецификации **MIME**, формат данных описывается следующим образом:

enctype=<тип>/<подтип>

Атрибут enctype определяет тип кодирования данных в теле сообщения и разбиение сообщения на части.

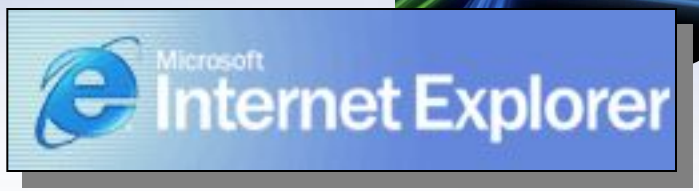
**Тип** определяет, какого рода информация содержится в двоичном файле (текст, приложение, изображение, видеозапись и т.п.), а **подтип** – формат файла.

Например: text/plain, где text- указывает на наличие текстового содержимого, а plain –уточняет его как простой текст.

Другие виды: text/html, text/plain, image/gif, image/jpeg, audio/basic, video/mpeg, multipart/mixed, application/octet-stream

# Приложения, использующие HTTP

- Клиентские (браузеры):
  - MS Internet Explorer
  - Opera
  - Apple Safari
  - Mozilla FireFox
  - Chrome






- Серверные (Web-серверы):
  - Apache (public domain)
  - MS Internet Information Server (IIS)
  - …


# Структура протокола HTTP

Каждое **HTTP-сообщение** состоит из трёх частей, которые передаются в указанном порядке:

- Стартовая строка (англ. *Starting line*) — определяет тип сообщения;

  *Метод URI HTTP/Версия*

  *GET /path/resource?param1=value1&param2=value2 HTTP/1.1*

- Заголовки (англ. *Headers*) — характеризуют тело сообщения, параметры передачи и прочие сведения;

  Server: Apache/2.2.11 (Win32) PHP/5.3.0
  Last-Modified: Sat, 16 Jan 2010 21:16:42 GMT
  Content-Type: text/plain; charset=windows-1251
  Content-Language: ru

- Тело сообщения (англ. *Message Body*) — непосредственно данные сообщения. Обязательно должно отделяться от заголовков пустой строкой.

Заголовки и тело сообщения могут отсутствовать, но стартовая строка является обязательным элементом, так как указывает на тип запроса/ответа.

# Методы HTTP

**Метод HTTP** (англ. *HTTP Method*) — последовательность из любых символов, кроме управляющих и разделителей, указывающая на основную операцию над ресурсом.

- **OPTIONS** Используется для определения возможностей веб-сервера или параметров соединения для конкретного ресурса.
- **GET** Используется для запроса содержимого указанного ресурса. С помощью метода GET можно также начать какой-либо процесс.
- **HEAD** Используется для запроса содержимого указанного ресурса. С помощью метода GET можно также начать какой-либо процесс.
- **POST** Применяется для передачи пользовательских данных заданному ресурсу.
- **PUT** Применяется для загрузки содержимого запроса на указанный в запросе URI
- **PATCH** Аналогично PUT, но применяется только к фрагменту ресурса.
- **DELETE** Удаляет указанный ресурс.
- **TRACE** Возвращает полученный запрос так, что клиент может увидеть, какую информацию промежуточные серверы добавляют или изменяют в запросе.
- **LINK** Устанавливает связь указанного ресурса с другими.
- **UNLINK** Убирает связь указанного ресурса с другими.
- **CONNECT** Преобразует соединение запроса в прозрачный TCP/IP туннель, обычно чтобы содействовать установлению защищенного SSL соединения через нешифрованный прокси.

# *Примеры диалогов HTTP*

**Запрос клиента:**

GET /wiki/*страница* HTTP/1.1
Host: ru.wikipedia.org
User-Agent: Mozilla/5.0 (X11; U; Linux i686; ru; rv:1.9b5) Gecko/2008050509 Firefox/3.0b5
Accept: text/html
Connection: close *(пустая строка)*

**Ответ сервера:**

HTTP/1.1 200 OK
Date: Wed, 11 Feb 2009 11:20:59 GMT
Server: Apache
X-Powered-By: PHP/5.2.4-2ubuntu5wm1
Last-Modified: Wed, 11 Feb 2009 11:20:59 GMT
Content-Language: ru
Content-Type: text/html; charset=utf-8
Content-Length: 1234
Connection: close *(пустая строка)*
*(далее следует запрошенная страница в HTML)*

# Метод GET

По умолчанию при запросе используется метод **GET**.

Для CGI-программиста очень важно понимать, что при запросе методом **GET** данные формы передаются серверу вместе с **URL**.

Web-серверы, поддерживающие CGI, копируют эти данные в переменную окружения с именем **QUERY_STRING**. После этого забота о получении данных из переменной окружения и их обработке возлагается на CGI-программу.

URL со строкой запроса выглядит так:

http://www.domen-name.com/login.php?nick=maks&psw=parol

Знак ? отделяет строку запроса от собственно URL ресурса; nick и psw - переменные передаваемые серверу, maks и parol - их значения соответственно.

# Метод POST

**Метод POST** используется тогда, когда это явно указано в атрибуте формы **METHOD**.

- В отличии от метода GET, **POST** помещает данные не в URL, а в тело запроса.
- Тело запроса при использовании метода POST передается программе как стандартный поток ввода.

URL со строкой запроса выглядит так:

http://www.domen-name.com/login.php

# Выбор между GET и POST

- Если вы хотите, чтобы ваша программа вызывалась с помощью ссылки, предпочтение следует отдать методу **GET**.

- Если вы **не** хотите, чтобы аргументы, передаваемые вашей программе, записывались в файл отчета сервера, используйте метод **POST**. *Например*, если форма требует указать имя пользователя и пароль, вы вряд ли захотите, чтобы имена и пароли сохранялись в файле отчета. Кроме того, не разумно передавать пароль как часть URL.

- Если ваша форма имеет значительные размеры, например в ней есть текстовые окна с заметками и комментариями, следует использовать метод **POST**. Вообще говоря, можно и в этом случае применять метод **GET**, но тогда вы сможете столкнуться с ограничениями на размер URL, разными для разных операционных систем и браузеров (ограничение обусловлено размером переменных окружения). Проще воспользоваться методом **POST**.

- Если ваша форма содержит файловое поле, используйте метод **POST**. Кроме того, в этом случае нужно установить значение атрибута **ENCTYPE** в **multipart/form-data**.

# Взаимодействие с пользователем

## Простой запуск скрипта – через URL

***php-файл (test.php)***

```
<?php
    echo 'Hello, world!';
?>
```

## Запуск скрипта через html-форму

***html-файл (fr.html)***

```
<form action = "http://mathsite/pm3/test.php" method="get">
    <input type="submit"/>
</form>
```

# Пасхальные яйца

- Отображение логотипов
  - script_name.php?=PHPE9568F36-D428-11d2-A769-00AA001ACF42
  - script_name.php?=PHPE9568F34-D428-11d2-A769-00AA001ACF42
  - script_name.php?=PHPE9568F35-D428-11d2-A769-00AA001ACF42
- Отображение авторов
  - script_name.php?=PHPB8B5F2A0-3C92-11d3-A3A9-4C7B08C10000