1. **Электронная почта. Принципы организации и функционирования. Протоколы SMTP, POP3 и IMAP.**

**Электро́нная по́чта** — технология и служба по пересылке и получению электронных сообщений между пользователями компьютерной сети.

**Протокол передачи почты** **SMTP** (англ. Simple Mail Transfer Protocol):

Порт сервера 25.

Обеспечивает доставку электронных писем, не рассчитан на работу с входящей почтой. Почтовые клиенты используют SMTP для отправки писем (современный порт сервера для отправки – 587).

Промежуточные сервера используют SMTP для пересылки отправленных писем адресату, на нужный почтовый сервер.

Данные в SMTP представлены в 7-разрядных символах ASCII. Каждый символ передаётся как 8 битов с установленным в 0 старшим разрядом. Расширения стандарта SMTP могут разрешить передачу полноценных 8-разрядных байтов данных.

**Протокол доступа к почтовому ящику POP3** (англ. Post Office Protocol v3):

Порт сервера 110.

Служит для изъятия части или всей почты из почтового ящика.

Ориентирован на малый размер хранилища почтовых ящиков на сервере. Ориентирован на работу с почтой в автономном режиме.

Состояния сеанса: AUTHORIZATION (требуется вход) – состояние после подключения клиента к серверу; необходимо авторизоваться для получения доступа к ящику. TRANSACTION (накопление изменений) – состоояние после успешной авторизации; сервер заблокировал ящик, и клиент работает с почтой; все изменения накапливаются, но не применяются сразу. UPDATE (применение изменений) – состояние после выхода клиента: сервер применяет накопленные изменения.

После авторизации клиент может: просматривать состояние ящика, просматривать весь список и содержимое отдельных писем, помечать их к удалению, отменять все текущие изменения.

Положительный ответ сервера: +ОК …, отрицательный: -ERROR …

**Недостатки протокола POP3:**

Отсутствие возможности управления письмами на сервере – перемещение по папкам, назначение ярлыков.

Необходимость «забирать» письма с почтового сервера для полноценной работы с почтой.

Отсутствие возможности работы несколькими клиентами с одним почтовым ящиком. Клиент A «забрал» письма, и клиент B их «не увидит».

**Протокол доступа и управления сообщениями IMAP** (англ. Internet Message Access Protocol):

Порт сервера 143.

Ориентирован на интерактивный режим работы с почтовым ящиком через интернет. Возможен одновременный доступ нескольких клиентов к одному ящику.

Возможность работы с несколькими ящиками (папками: «Входящие», «Исходящие», «Помеченные» и др.).

Гибкое управление почтой на сервере: ярлыки, пометки «важное», перемещение писем между ящиками.

Возможность отслеживать состояние письма (например, «прочитано», «отправлен ответ», «отложено», «удалено»).

Состояния сеанса: Not Authenticated (клиент не аутентифицировался) Authenticated (клиент аутентифицировался) Selected (клиент выбрал ящик) Logout (клиент выходит)

1. **Прикладной протокол FTP. Принципы организации и функционирования. Команды и их формат.**

FTP (File Transfer Protocol) – это протокол передачи файлов.

Использует раздельные соединения для управления и передачи файлов.

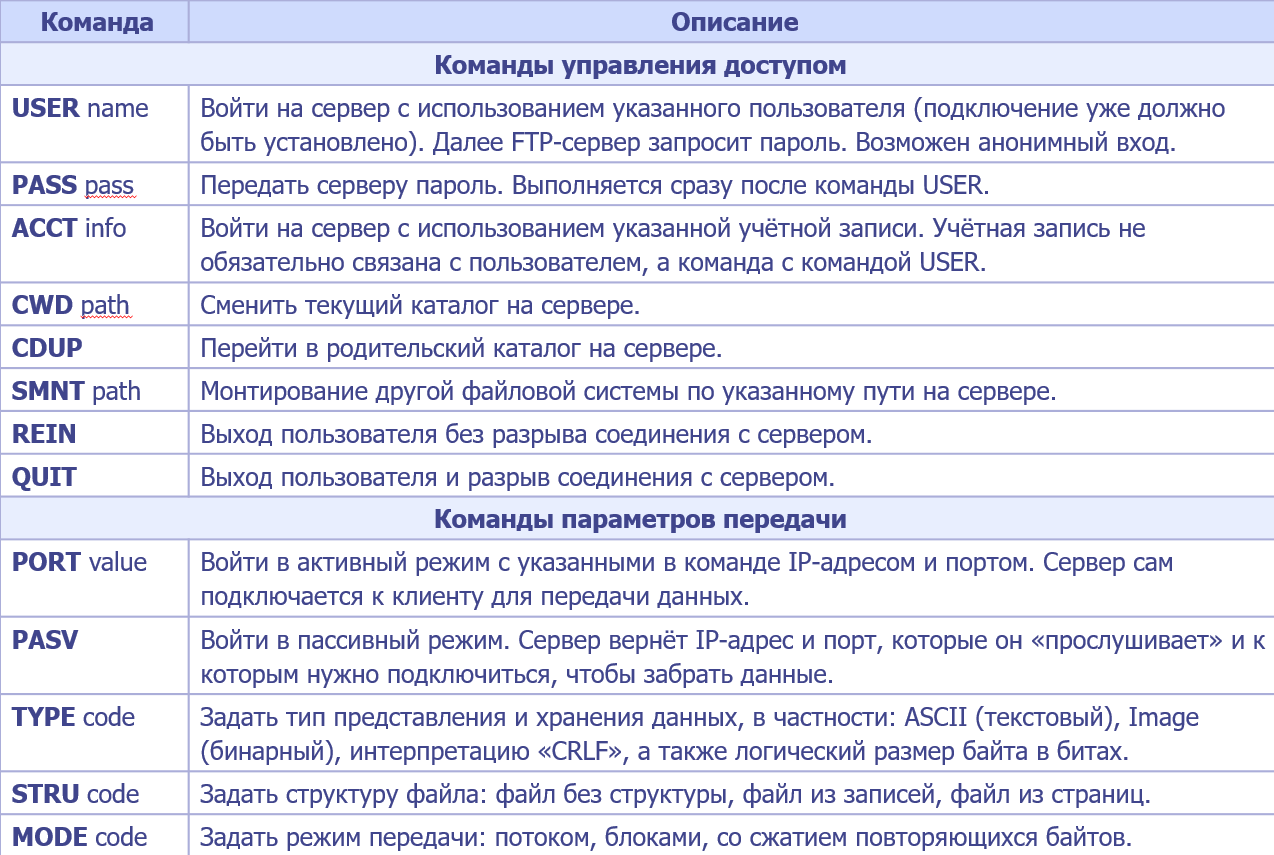
Соединение для управления устанавливается на порт сервера 21. Через это соединение передаются текстовые команды и ответы на них.

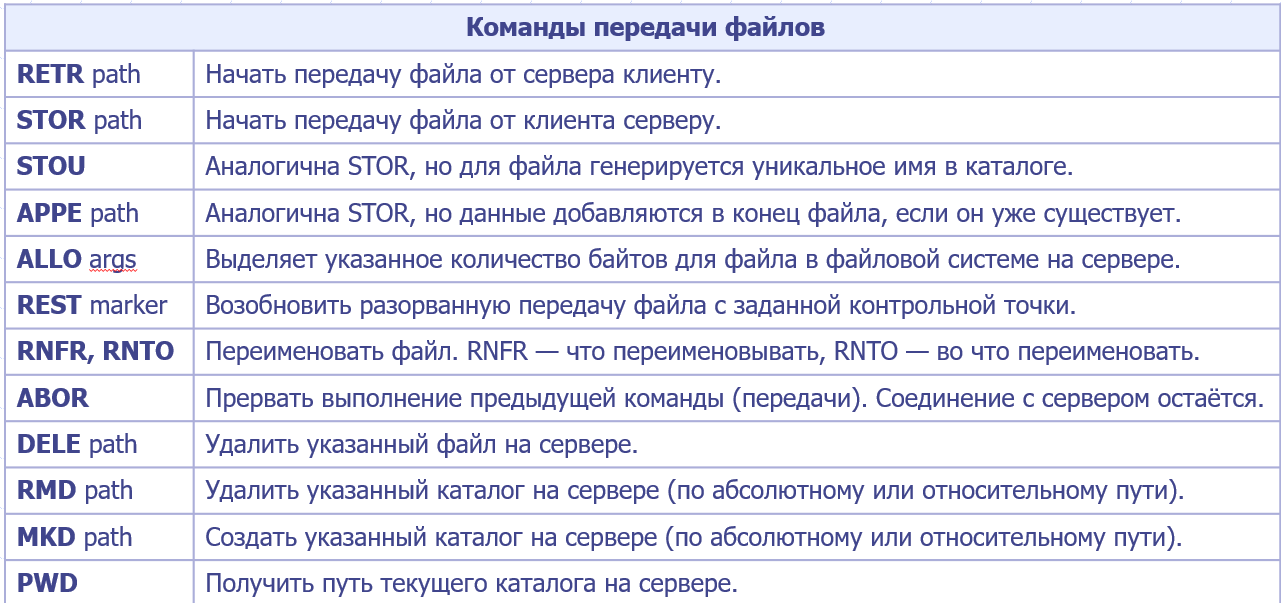
Соединение для передачи файлов открываются по мере необходимости. Инициатором соединения может выступать как сервер, так и клиент.

Режимы работы:

* Активный – инициатором соединения является сервер. Не пригоден, если клиент находится за устройством PAT или сетевым экраном.
* Пассивный – инициатором соединения является клиент. Позволяет клиенту использовать технологию PAT и сетевой экран.

После установления TCP-соединения клиент выполняет аутентификацию и авторизацию. Имя пользователя и пароль передаются в открытом виде.

После авторизации клиент с помощью соответствующих команд может: просматривать содержимое каталогов, менять текущий каталог, управлять файлами и каталогами на сервере, отправлять файлы на сервер и получать файлы с сервера. 



1. **Протокол HTTP. Принципы организации и функционирования. Формат URL-адресов, структура запроса и ответа.**

**HTTP(Hyper-Text Transfer Protocol)** — широко распространённый протокол передачи данных, изначально предназначенный для передачи гипертекстовых документов.

В соответствии со спецификацией OSI, HTTP является протоколом прикладного уровня.

Протокол HTTP предполагает использование клиент-серверной структуры передачи данных. Клиентское приложение формирует запрос и отправляет его на сервер, после чего серверное программное обеспечение обрабатывает данный запрос, формирует ответ и передаёт его обратно клиенту.

Для того, чтобы сформировать HTTP-запрос, необходимо составить стартовую строку, а также задать по крайней мере один заголовок — это заголовок Host, который является обязательным, и должен присутствовать в каждом запросе.

Коды ответов HTTP

100-199 – Информационные

200-299 – Успех

300-399 – Переадресация

400-499 – Ошибка со стороны клиента

500-599 – Ошибка со стороны сервера

Uniform Resource Identifier



Структура запросов и ответов в 1.1

Запросы и ответы – это текст, состоящий из заголовков и содержимого.

Заголовок – это строка, оканчивающаяся символами «CRLF».

Блок заголовков отделяется от содержимого пустой строкой (двумя парами символов «CRLF»).

Первый заголовок запроса содержит тип, путь и версию протокола.

Первый заголовок ответа содержит версию протокола и код ответа.

Все заголовки делятся на: заголовки запроса; заголовки ответа; заголовки запросов и ответов.

1. **Типы HTTP-запросов и их классификация. Основные HTTP-заголовки. Понятие "куки".**

GET – получить текущее представление указанного ресурса.

HEAD – получить информацию о текущем представлении ресурса (аналогичен GET, но в ответе только заголовки, без содержимого).

PUT – заменить текущее представление указанного ресурса или создать новый ресурс.

POST – отправить данные на сервер, добавив к указанному ресурсу.

DELETE – удалить указанный ресурс.

PATCH – обновить, частично изменить указанный ресурс

(добавлен в 2010 году).

OPTIONS – получить параметры соединения с указанным ресурсом: доступные методы, форматы.

TRACE – проверить прохождение запроса к указанному ресурсу.

+ CONNECT – установить соединение (при использовании прокси-сервера).

Классификация:

* Безопасные (не изменяют ресурсы на сервере). Например, запросы на чтение (GET, HEAD, OPTIONS, TRACE)
* Небезопасные (могут изменять ресурсы). Например, запросы на изменение (PUT, POST, DELETE, PATCH)
* Идемпотентные (результаты повторных вызовов идентичны результату одиночного вызова). Например, GET, HEAD, OPTIONS, TRACE, PUT, DELETE.
* Не идемпотентные (повторный вызов приводит к новому изменению). Например, POST, PATCH.

**Основные HTTP-заголовки**

Заголовки управления соединением: Connection, Keep-Alive

Заголовки согласования типа содержимого: Accept, Accept-Encoding

Заголовки свойств содержимого: Content-Type, Content-Encoding, Content-Length

Заголовки условий: Last-Modified, ETag , If-Match

Заголовки «куки»: Set-Cookie, Cookie

Заголовок переадресации: Location

Заголовки кэширования: Cache-Control, Expires

Заголовки контекста запроса: Host, User-Agent

Заголовки контекста ответа: Allow, Server

HTTP cookie – это небольшой фрагмент данных, хранящий состояние для протокола HTTP. Отправляется сервером клиенту, который может сохранить эти данные и отсылать обратно с новыми запросами к данному серверу.

Применение:

* Управление сеансом – логины, корзины для виртуальных покупок;
* Персонализация – пользовательские предпочтения, оформление;
* Мониторинг – отслеживания поведения пользователя, таргетированная реклама.

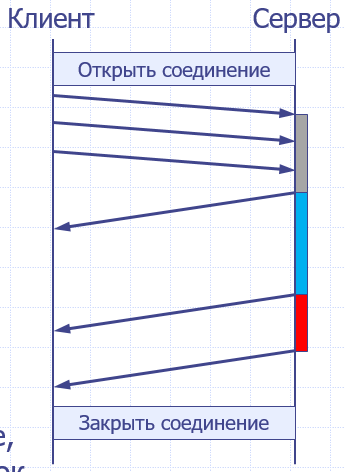
«Куки» бывают:

* временные (сессионные) или постоянные (указан атрибут Expires или Max-Age);
* «безопасные», требующие протокол HTTPS (указан атрибут Secure);
* недоступные из JavaScript (указан атрибут HttpOnly).

1. **Виды HTTP-соединений и их особенности. Организация конвейера запросов-ответов. Составной HTTP-ответ, размер которого не известен в момент отправки.**

HTTP-соединение:

* Единовременное (close) – по одному соединению передается только один запрос ответ.
* Постоянное(keep-alive) – по одному соединению может передаваться несколько запросов и ответов последовательно. В ответе обязательно указываются Content-Length и Content-Type.

**Конвейера запросов-ответов**

Отсылка новых запросов по одному соединению без ожидания ответов на предыдущие запросы (HTTP/1.1).

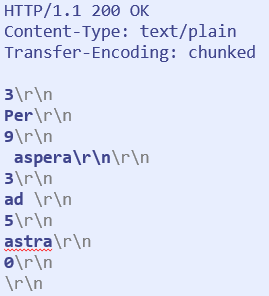
Поддерживает только идемпотентные запросы (GET, HEAD, PUT, DELETE), повторное выполнение которых безопасно.

Можно повысить производительность, упаковав несколько HTTP запросов в один TCP-пакет.

В HTTP/1.1 отсутствуют средства сопоставления запросов и ответов, поэтому ответы обязаны приходить в том же порядке, что и отправленные запросы. Этот недостаток не позволяет полноценно использовать конвейерный режим запросов-ответов.

**Составной HTTP-ответ, размер которого не известен в момент отправки**

Указывается служебный заголовок Transfer-Encoding: chunked.

Ответ отправляется по частям (англ. chunks) с указанием размера каждой части. Конец ответа обозначается пустым блоком нулевой длины.

1. **Использование заголовков условий HTTP при кэшировании содержимого ответов и совместного редактирования данных.**

Современные веб-браузеры поддерживают кэширование редко меняющихся ресурсов.

Может быть кэширование не только страниц но и ресурсов страниц, например, изображений, таблиц стилей, библиотек JS.

Кэширование используется для ускорения загрузки страниц, но при этом расходует память на локальном диске компьютера для кэша.

Протокол HTTP поддерживает работу хэша

Чтобы браузер понимал, когда нужно запрашивать новую страницу, а когда брать из кэша используется заголовок Expires, который указывает до какого вр емени можно хранить ресурс в кэше. Но веб-сервера не всегда устанавливают этот заголовок. (Пример: Expires: Sun, 12 Jun 2016 10:35:18 GMT).

Если Expires не установлен то браузер может использовать поле Last-Modified, в котором указано время последнего изменения и если ресурс давно не менялся то может использовать ресурс из кэша. Но в данном случае возрастает шанс ошибки.

**Запрос Get с условием(Conditional get):**

Это запрос веб-серверу передать ресурс в том случае, если он был изменен.

При первом запросе веб-серверу:

* Ответ сервера содержит заголовок Last-Modified.
* Ресурс записывается в кэш.

При повторном обращении

* Запрос будет содержать заголовок If-Modified-Since: {здесь будет Last-Modified который мы получили при первом обращении}

**В ответ на If-Modified-Since может прийти два ответа:**

* 304 Not Modified. В этом случае могут быть переданы дополнительные заголовки: Cache-Control, Expires, Last-Modified
* 200 OK. Полная передача обновленной в ерсии ресурса.

**ETag(entity tag) начиная с HTTP 1.1**

Код который генерируется на основе содержимого ресурса.

Этот заголовок прикрепляется к ответу.(ETag: “123-8df”)

Запрос GET с условием:

If-None-Match: 123-8df

**Заголовок Cache-Control начиная с HTTP 1.1**

С помощью этого заголовка можно более гибко управлять кэшированием.

Cache-Control: private, max-age = 10

Возможные значения: no-store - не сохранять в кэш, no-cache - можно сохранять, но при использовании выполнить conditional get, public - информация доступна всем и ее можно кэшировать, private - может быть сохранена только в частном кэше браузера, max-age - время хранения в кэше в секундах.

**ПРЕЗЫ. При кэшировании:**

Клиент запрашивает данные.

Сервер возвращает данные с посчитанным хэш-кодом и датой последнего изменения.

Клиент кеширует данные.

При повторном запросе данных заголовки условий позволяют серверу узнать версию данных у клиента и избежать повторной передачи содержимого.

**Совместное редактирование:**

Клиент A получает данные, изменяет их локально и отправляет серверу.

Клиент B получает данные одновременно с клиентом А, изменяет их локально и отправляет серверу свои изменения позже, чем клиент А.

Изменения клиента A применяются, а изменения клиента B отбрасываются с ошибкой.