

Владивостокский государственный университет  
экономики и сервиса  
Институт информатики инноваций и бизнес  
систем

**Предмет:**  
**«Технологии Интернет»**

Руководитель: Сачко Максим Анатольевич,  
старший преподаватель

# Тема 5

## Информационные сервисы

# Содержание:

- 1) Протокол FTP и его команды
- 2) Схемы работы.
- 3) Создание и администрирование анонимного FTP-сервера
- 4) Протокол HTTP
- 5) WWW-сервер Apache

---

**Цель:** изучить работу протокола FTP; научиться конфигурировать и администрировать анонимный FTP-сервер под ОС Unix.

**Необходимость:** FTP является сервисом доступа к файловым архивам: программному обеспечению, документации, и т.д. Необходимость предоставлять к таким данным публичный или корпоративный доступ часто появляется в работе организации.

---

**Основная концепция:** Сервис FTP обеспечивает авторизованный и анонимный доступ к удаленным файловым архивам через Интернет с помощью протокола FTP, находящегося на прикладном уровне стека TCP/IP.

Серверная часть обслуживается стандартным демоном **in.ftpd** (*Unix*) или специальными программами с расширенной функциональностью.

# Протокол FTP

---

**FTP** (*File Transfer Protocol* или "Протокол Передачи Файлов") - клиент-серверный протокол прикладного уровня, обеспечивающий навигацию клиента по файловой системе сервера, управление файлами и пересылку файлов между клиентом и сервером.

При работе с FTP между клиентом и сервером функционируют два соединения: одно для передачи текстовых управляющих команд клиента и откликов сервера, другое - для передачи собственно файлов.

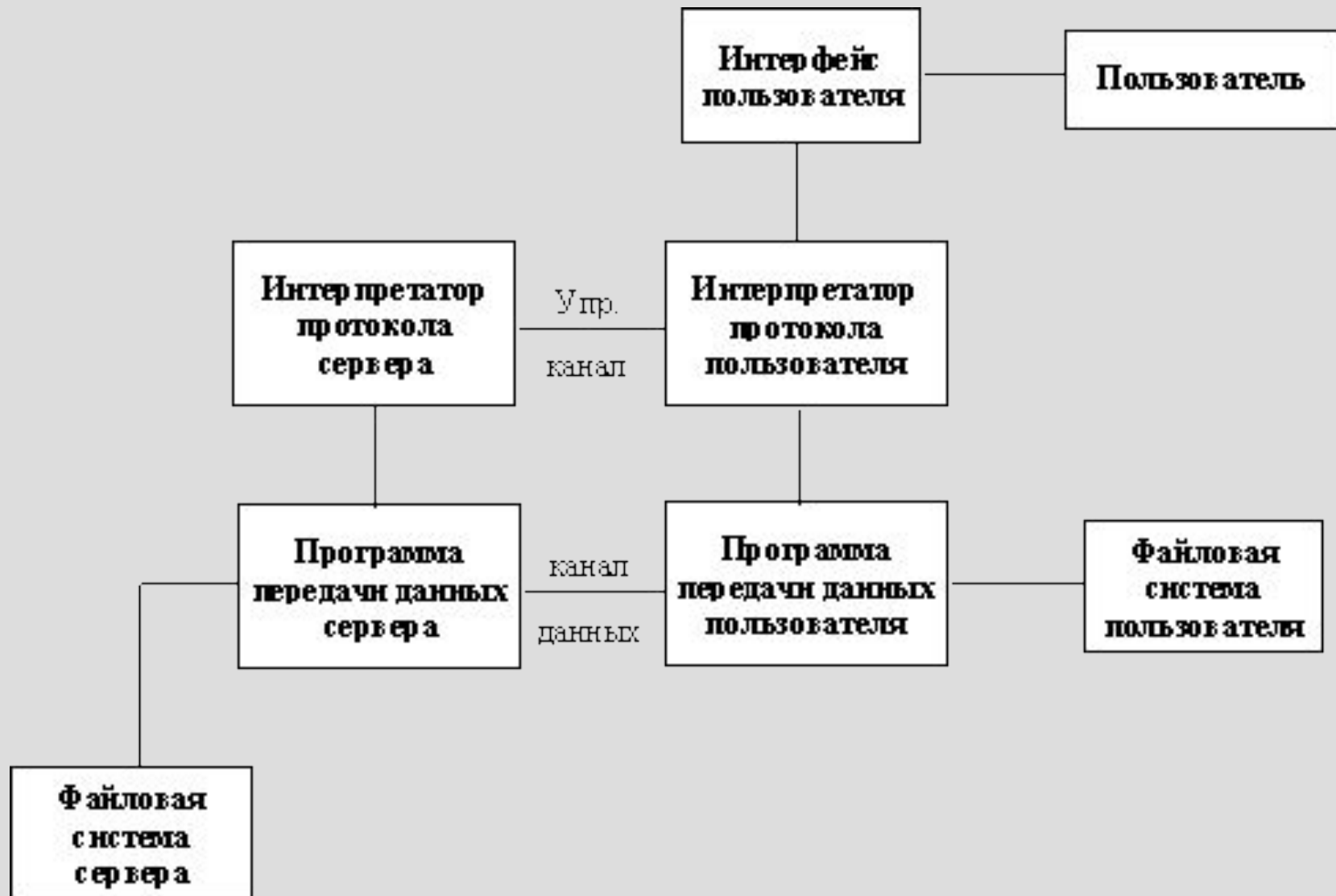
# Ключевые понятия

---

**Управляющий канал ftp** - соединение со стандартным портом TCP/21, по которому (в текстовом виде) передаются команды протокола FTP и отклики на них. Управляющий канал открывается при подсоединении клиента к серверу FTP.

**Канал данных ftp** - соединение со стандартным портом TCP/20, по которому передаются данные - содержимое файлов. Канал данных открывается автоматически при поступлении соответствующей команды (get, put, dir и др.) по каналу управления.

# Модель протокола FTP



# Соединение с двумя разными серверами и передача данных между ними



# Сервис World Wide Web

---

**WWW** является наиболее популярным из сервисов Интернет - настолько, что часто Интернет и WWW используются как синонимы. На технологиях WWW базируется электронная коммерция и другие Интернет-бизнесы. Эффективность организации веб-сервера на предприятии может иметь решающее значение для успеха деятельности предприятия.

# Основная концепция

---

**WWW** - клиент-серверная технология, базирующаяся на прикладном протоколе HTTP, в котором имеются два типа сообщений: запросы от клиента (броузера) к серверу и ответы сервера клиенту. Запросы и ответы содержат заголовки со служебной информацией и данные (в запросе данными может быть содержимое заполненной пользователем формы).

# Протокол HTTP

---

**Протокол HTTP** (*Hyper Text Transfer Protocol*) находится на прикладном уровне стека TCP/IP и состоит из запросов, посылаемых клиентом, и ответов, возвращаемых сервером. Запрос состоит из строки запроса, заголовков и данных (если они есть; например, если пользователь заполнил HTML-форму). В заголовках запроса могут быть указаны дополнительные параметры, уточняющие запрос или определяющие представление запрашиваемого документа.

# Тема

HTTP - прикладной клиент-серверный протокол типа "запрос-ответ" работающий поверх TCP (стандартный порт 80).

Клиентом является браузер; сервером - WWW-сервер.

В первой версии протокола (1.0) на одно TCP-соединение приходился один запрос и ответ, после чего соединение закрывалось. В настоящее время протокол позволяет выполнить несколько итераций в рамках одного соединения.

# HTTP-запрос

---

|                   |
|-------------------|
| Строка запроса    |
| Заголовки запроса |
| Заголовки данных* |
| Пустая строка     |
| Данные*           |

Поля, помеченные звездочкой (\*), присутствуют только, если в запросе использован метод POST или PUT

# Пример

---

GET / HTTP/1.1

POST /cgi-bin/sript?X=1&Y=a%20b HTTP/1.1

HEAD http://www.vvsu.ru/index.html HTTP/1.1

# Основные методы

---

**GET** - запрос документа, идентифицируемого указанным URI.

**HEAD** - запрос заголовков документа, идентифицируемого указанным URI (аналогично GET, но само содержимое документа не возвращается).

**POST** - пересылка данных на сервер (используется при заполнении форм). URI в этом случае обычно идентифицирует CGI-скрипт, который будет обрабатывать присланные данные, и результат этой обработки будет возвращен в HTTP-ответе.

# HTTP-ответ

---

|                   |
|-------------------|
| Статусная строка  |
| Заголовки ответа  |
| Заголовки данных* |
| Пустая строка     |
| Данные*           |

Поля, помеченные звездочкой (\*), могут отсутствовать - это зависит от типа запроса.

# Пример

---

HTTP/1.1 200 OK

HTTP/1.1 304 Not Modified

# Код ответа

---

- 1\*\*** - Промежуточные информационные сообщения (практически не используются).
- 2\*\*** - Успешная обработка запроса.
- 3\*\*** - Для получения документа требуются дополнительные действия со стороны клиента.
- 4\*\*** - Ошибка клиента.
- 5\*\*** - Ошибка сервера.

# WWW-сервер Apache

---

Веб-сервер **Apache** имеет открытую модульную архитектуру и открытый API, что позволяет создавать различные модули, расширяющие функциональность сервера.

Администрирование сервера производится путем редактирования текстового файла с директивами

# Общая схема работы Apache

---

1. Запуск сервера - этот процесс запускается с большими привилегиями (например, root - для доступа к стандартному HTTP-порту 80). Непосредственно запросы клиентов этим процессом не обслуживаются; он управляет работой сервера. Запустившись, сервер производит разбор директив в конфигурационном файле (файлах), открывает лог-файлы, перенаправляет вывод стандартной ошибки в файл, указанный директивой ErrorLog. Этими же лог-файлами будут пользоваться все порожденные процессы.

# Общая схема работы Apache

---

2. Инициализация модулей.

3. Запуск дочерних процессов - главный процесс ветвится с помощью вызова `fork`, порождая несколько дочерних процессов, которые собственно и будут обслуживать запросы клиентов. В целях безопасности порожденные процессы запускаются с низким уровнем привилегий (по умолчанию - как пользователь `nobody`). Число порождаемых процессов определяется директивой `StartServers`.

# Общая схема работы Apache

---

4. Запрос, поступающий от клиента, отправляется на обработку одному из дочерних процессов. Цикл обработки запроса является основным смыслом деятельности сервера, он разобран отдельно в следующем пункте. После завершения обработки запроса процесс ожидает поступления следующего запроса. При достижении максимального числа обработанных запросов (директива `MaxRequestsPerChild`) дочерний процесс прекращает свою работу.

# Общая схема работы Apache

---

5. Главный процесс осуществляет мониторинг дочерних процессов. Если число процессов падает ниже установленного директивой `MinSpareServers`, то запускаются дополнительные дочерние процессы. При одновременном поступлении большого числа запросов главный процесс также запускает дополнительные процессы, но после прохождения пика лишние процессы будут удалены так, чтобы общее число ожидающих запроса процессов не превышало установленного директивой `MaxSpareServers`.

Максимальное количество порожденных процессов определяется директивой `MaxClients`, что и является ограничением на число одновременно обслуживаемых сервером запросов.

# Общая схема работы Apache

---

6. При получении команды "stop" (сигнал TERM) главный процесс останавливает дочерние процессы, а потом выходит сам. При получении команды "restart" (сигнал HUP) главный процесс останавливает дочерние процессы, но сам не выходит; заново производится разбор конфигурационных директив и инициализация модулей, потом запускаются дочерние процессы. При получении команды "graceful" (сигнал USR1) действия сервера аналогичны команде "restart", однако дочерние процессы не прерываются принудительно, а им дается возможность обслужить текущий запрос (если таковой имеется) и только после ЭТОГО ВЫЙТИ.

# Вопросы для самопроверки:

1. Какие способы контроля и ограничения имеются в анонимном FTP?
2. Опишите алгоритм аутентификации клиента в протоколе HTTP.
3. Опишите алгоритм определения кодировки посылаемого клиенту документа, которым пользуется Russian Apache.
4. Укажите приоритет контекстов конфигурации в Apache.
5. Какие коды HTTP-ответа вы знаете?

# Рекомендуемая литература:

1. Мамаев М., Петренко С. Технологии защиты информации в Интернете. Специальный справочник. – СПб: "Питер", 2005.
2. К. Хант. Персональные компьютеры в сетях TCP/IP: Руководство администратора сети/ Пер. с англ. – СПб.: ЗАО "ЭлектроникаБизнесИнформатика", Киев: "BNV", 2003.
3. UNIX для системных администраторов: Энциклопедия пользователя/ Пер.с англ. – Киев: ДияСофт, 2008.

- **Использование материалов презентации**

- Использование данной презентации, может осуществляться только при условии соблюдения требований законов РФ об авторском праве и интеллектуальной собственности, а также с учетом требований настоящего Заявления.
- Презентация является собственностью авторов. Разрешается распечатывать копию любой части презентации для личного некоммерческого использования, однако не допускается распечатывать какую-либо часть презентации с любой иной целью или по каким-либо причинам вносить изменения в любую часть презентации. Использование любой части презентации в другом произведении, как в печатной, электронной, так и иной форме, а также использование любой части презентации в другой презентации посредством ссылки или иным образом допускается только после получения письменного согласия авторов.