Торговые марки

UNIX является торговой маркой Unix System Laboratories, Inc. Sun и SunOS являются торговыми марками Sun Microsystems, Inc. Netscape является торговой маркой Netscape Communications Corporation. Все остальные продукты, упомяниутые в настоящей книге принадлежат их уважаемым владельцам.

Содержание

1

Предисловие

Этот руководство посвящено самому популярному на сегодняшний день серверу WWW Apache версии 1.2 и в части справочного материала основано на разработках Apache Group. Автор искренне полагает, что остутствие подобной информации на отечетсвенном рынке самым неблагоприятным образом сказывается на распространении в России Internet/Intranet-технологий.

На сегодняшний день сервер Apache является наиболее популярным в России и по ряду оценок, некоторые из которых были опубликованы в майских номерах еженедельника PC Week/RE количество серверов Apache в России в 8-10 раз превышает кодичество **всех прочих установленных серверов**. Этот факт вызывает еще большее удивление о постоянном замалчивании самого факта существования полноценного, высокопроизводительного Web-сервера, не говоря уже о полном руководстве по его применению.

Сервер представляет собой результат интенсивных работ над проектом Apache, направленном на решение вопроса по созданию общедоступного (бесплатного) сервера HTTP. Конечная цель проекта состоит в разработке защищенного, производительного и расширяемого сервера, который осуществляет обработку HTTP-запросов в в соответствии с современным состоянием стандартов HTTP.

Сервер проекта Apache — httpd совместим с httpd 1.3 NCSA, и в то же время обладает несколькими важными особенностями:

- Поддержка баз данных DBM для решения задач аутентификации;

- Полностью программируемая реакция на ошибки и нештатные ситуации;
- Многократное использование директив **directoryindex** ;
- Неограниченное количество директив **Alias** и **Redirect**;
- Управление документом, основанное на его содержании;
- Создание и управление виртуальными серверами.

Распространение и использование сервера Apache в исходных текстах и скомпилированном виде допускается при выполнении следующих условий, установленных разработчиками сервера:

- Распространение исходного текста программы должно осуществляться только вместе с идентификатором авторских прав, данным списком условий и приведенным ниже сообщением об отказе от ответственности за последствия использования данной программы.
- Распространение программы в двоичном (скомпилированном) виде должно сопровождаться приведенным ниже идентификатором авторских прав и сообщением об отказе от ответственности, помещенным в сопроводительную документацию и/или иные материалы, включенные в состав дистрибутива.
- Все рекламные материалы, упоминающие возможности или использование данного программного продукта должны содержать сообщение следующего содержания:

  This product includes software developed by the Apache Group for use in the Apache HTTP server project (http://www.apache.org).
- Названия *Apache Server* и *Apache Group* не должны использоваться для рекламы или продвижения продуктов, созданных на основе данного сервера без *предварительного письменного разрешения*.

Ниже приведен оригинальный текст заявления о принадлежности авторских прав.

This software is provided by the APACHE GROUP “as is” and any expressed or implied warranties, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose are disclaimed. In no event shal the APACHE GROUP or its contributors be liable for any direct, indirect, incidental, special, extemplary or consequential damages (including, but not limited to, procurement of substitute goods or services; loss of the use, data or profits; or business interruption) however caused and on any theory of liability, whether in contract, strict liability, or tort (including negligence or otherwise) arising in any way ouut of the use of this software, even if advised of the possibility of such damage.

(Мы сочли возможным привести оригинальную версию отказа от всякой ответственности за любые неприятности, возникшие в результате использования сервера Apache в оригинальном виде, чтобы любознательнный читатель самостоятельно мог разобраться в буржуазной казуистике. А от себя добавим, что несмотря на подобное заявление, сервер Apache сегодня является самым популярным в сети Internet, и не только на платформе Intel — прим.авт.)

Данное программное обеспечение содержит набор дополнений и расширений, созданных многочисленными пользователями Apache и основано на общедоступном программном обеспечении, созданном в Национальном Центре Суперкомпьютерных Вычислений, Университете Иллинойс, Урбана-Шампейн. Для получения дополнительной информации относительно Apache Group и проекта создания сервера httpd обращайтесь по адресу: ***http://www.apache.org***.

Вся необходимая информация по вопросам обработки русскоязычных документов, представленных в различных кодировках, может быть найдена по адресу:

http://www.apache.lexa.ru,

а новая, более полная версия этой книги — по адресу:

http://www.geocities.com/SiliconValley/Pines/7895.
Все замечания и пожелания[1] просьба отправлять по адресу:
voldemarus@geocities.com.

[1]Критические замечания приветствуются!

Часть I

Руководство пользователя

Глава 2

Быстрый старт

В этой главе, добавленной при переводе настоящего руководства, мы рассмотрим, каким образом запустить сервер Apache, поставляемый в двоичном виде в большинстве дистрибутивов Linux **до** прочтения всего руководства. Сразу же обращаем внимание на тот факт, что эта глава не является полноценной заменой всего документа в целом, и направлена только на проверку работоспособности сервера *httpd* или на его установку в небольшой локальной сети.

2.1 Поехали!

Прежде всего вы должны убедиться, что установили сервер httpd с дистрибутивных дисков. Для этого достаточно воспользоваться одной из программ просмотра установленных в системе пакетов. Конкретная версия программы зависит, как правило от версии операционной системы и дистрибутива, например, для Slackware таким средством является `pkgtool`.

Если сервер установлен, необходимо проверить его работоспособность. Сделать это можно двумя способами:

- использовать команду **ps -ax — grep httpd** , чтобы убедиться в наличии (отсутствии) сервера в памяти компьютера;
- просто запустить сервер с командной строки: **httpd**

В случае, если драйвер сконфигурирован и запущен, то вы сможете обнаружить несколько процессов httpd в памяти, а если вы запускаете сервер самостоятельно, то он аккуратно перейдет в фоновый режим без единого дополнительного сообщения.

Впрочем, скорее всего этого не произойдет. Связано это с тем, что параметры сервера при установке дистрибутива из архивного файла `не настраиваются`, а лишь копируются с дистрибутива. Поэтому содержимое этих параметров, по большому счету, никак не связано с реальным положением дел в вашем компьютере.

Наиболее простой способ настройки сервера Apache состоит в последовательном устранении всех препятствий, которые возникают на пути установки программы в памяти. Конечно, при этом мы сможем запустить лишь совершенно стандартный, незащищенный сервер, но ведь это только начало...

2.2 Настройка шаг за шагом

Итак, пойдем по пути наименьшего сопротивления. При первой же попытке запустить httpd вы получите следующее сообщение:

```
Syntax error on line 37 of /var/lib/httpd/conf/httpd.conf:
ServerRoot must be a valid directory
```

Вывод предельно прост. Необходимо открыть на редактирование соответствующий файл и установить корректное значение каталога `ServerRoot`. По умолчанию ServerRoot имеет значение */usr/lib/httpd*, но дело в том, что такого каталога при установке с дистрибутива просто не создается! Не совсем понятно, почему httpd при старте пытается обратиться к `/var/lib`..., ну да ладно, мы просто воспользуемся тем фактом, что нам известно, где сервер ищет свои конфигурационные файлы.

Взгляните на каталог `/var/lib/httpd`. Его хозяином является системный администратор (root) и все операции вам необходимо делать именно под этим идентификатором.

```
total 16
-rw-r--r--   1 root     root  4450 Aug  4  1995 CHANGES
-rw-r--r--   1 root     root  2605 Oct 13  1995 LICENSE
-rw-r--r--   1 root     root  2461 Nov 22  1995 README
drwxr-xr-x   2 root     root  1024 Feb 13 22:21 cgi-bin/
drwxr-xr-x   2 root     root  1024 Feb 13 22:21 conf/
drwxr-xr-x   2 root     root  1024 Feb 13 22:21 htdocs/
drwxr-xr-x   2 root     root  1024 Feb 13 22:21 icons/
drwxr-xr-x   2 root     root  1024 Feb 13 22:21 logs/
```

Его структура вполне соответствует ожиданиям сервера, а поэтому без лишних изысков мы корректируем строку в конфигурационным файле следующим образом:

```
ServerRoot /var/lib/httpd
```

Кстати, раз уж мы приступили к редактированию `httpd.conf`, то давайте сразу же изменим еще одну переменную — `ServerAdmin`, которая по умолчанию имеет совершенно абстрактное значение: *you@your.address*. Назначение этого адреса двоякое. Во-первых, по нему к вам будут обращаться за помощью пользователи, которым этот адрес сообщается при возникновении ошибок, отказа в доступе и так далее. А во-вторых, сам сервер будет посылать вам сообщения именно по этому адресу.

Самый очевидный путь — подставить свой собственный адрес имеет как хорошие, так и плохие стороны, причем, как показала практика, последние существенно перешивают. Дело в том, что пока с вами общается сервер вы в состоянии понять смысл письма: формализованное сообщение, адрес отправителя и т.д. Но как только вас начинают засыпать вопросами пользователи, то выясняется, что в большинстве случаев они не удосуживаются уточнить, что речь идет о проблемах с сервером, и разобрать смысл послания становится весьма и весьма затруднительно. Но и это еще не все. Предположим, на вашем компьютере функционирует *несколько* виртуальных серверов. От какого из серверов пришло письмо?

Поэтому не стоит жадничать – заведите на каждый из серверов отдельный пользовательский счет и адресуйте на него всю почту. Впрочем, пока мы пытаемся только запустить сервер, можно установить ServerAdmin на свой текущий счет.

Но давайте продолжим процесс установки. После внесения изменений в `httpd.conf` попытаемся еще раз запустить сервер. В результате мы получим следующее сообщение:

```
Syntax Error on line 15 of /var/lib/httpd/conf/srm.conf:
DocumentRoot must be a directory
```

Итак, мы продвинулись, хотя и незначительно. По крайней мере, сервер не обнаружил никаких противоречий в первом конфигурационном файле `httpd.conf` и перешел к обработке следующего. Возникшая теперь проблема ничем не отличается от предыдущей — сервер не может найти каталог, в котором должны храниться документы, предназначенные для пользователей сервера.

В самом деле, вот что вы обнаружите в файле `srm.conf`:

```
DocumentRoot /usr/lib/httpd/htdocs
```

Как и в прошлый раз нам достаточно заменить имя каталога, чтобы получить корректный отзыв сервера. В нашем случае вполне подойдет:

```
DocumentRoot /var/lib/httpd/htdocs
```

Теперь пытаемся запустить сервер еще раз. О чудо! Конфигурационные файлы загружены без проблем, сервер проинициализирован и запущен. Как нам в этом убедиться? Для этого нам необходимо взглянуть на каталог, в которм хранятся журналы работы сервера. Один из этих файлов `error_log` содержит нужную нам информацию. Вот его содержимое после запуска:

```
[...] Server configured -- resuming normal operations
```

Безусловно, рано говорить о том, чтобы запустить сервер для работы в Internet.

2.3 Тестовый набор данных

Часть II

Практическая работа с сервером

В этой части мы рассмотрим вопросы эффективного использования сервера Apache для решения практических задач предоставления WWW-доступа в сетях Internet и Intranet. Для этого нам потребуется осветить широкий круг вопросов, большая часть из которых, в той или иной степени связана с обеспечением безопасности работы сервера, контроля целостности данных, анализа эффективности использования вашего узла WWW и так далее.

Несколько особняком стоит проблема поддержки кириллицы при организации доступа к WWW. Как ни странно, но использование нескольких типов кодировок на одном сервере не представляет для Apache серьезной проблемы, при условии внесения в сервер некоторых исправлений.

Глава 3

Тотальная поддержка кириллицы

Прежде всего необходимо объяснить, почему в заголовок вынесено столь суровое определение — *тотальная поддержка*. Дело в том, что чаще всего пользователи и администраторы систем сталкиваются с наспех выполненными "заплатками", которые лишь частично решают проблемы эффективной работы с русскоязычными документами. При работе в Internet наряду с традиционными неприятностями — именами файлов, сортировкой и поиском, возникает еще одна неприятность, обусловленная так называемым "синдромом Вавилонской Башни", известным также как "война кодировок", обусловленная наличием нескольких способов представления кириллического алфавита в кодовом пространстве набора ASCII[1]. На сегодняшний день на территории России наибольшее распространение получили три типа кодировки:

- **Альтернативная** — известная также как CodePage 866, используется на персональных компьютерах, работающих под управлением операционной системы MS-DOS.

[1]Конечно, автора можно поставить на место, заявив, что в ASCII регламентируется только первые 127 символов, а все остальное — это самодеятельность. Но позвольте заметить, что поскольку **все** кодировки кириллицы представляют собой *расширение* кодировки ASCII до 256 символов, мы вполне можем "обозвать" и полный набор символов *кодировкой ASCII*.

- **Windows** — имеющая синоним CodePage 1251, разработана фирмой Microsoft для русифицированной версии своего программного продукта. Вопреки ожиданиям, оказалась *несовместимой* с предыдущей, что привело к огромному количеству забавных (и не только) казусов.

- **КОИ-8** — наиболее старая кодировка, использующаяся в настоящее время в мире UNIX-машин. Понятно, что большинство администраторов и программистов, работающих с Internet, отдают ей предпочтение, как более привычной.

Кроме того, разработано еще несколько видов кодовых таблиц, например, ISO-8859-5 или Apple, которые однако, особой популярностью и распространенностью в нашей стране не пользуются.

Обсуждение преимуществ и недостатков различных способов использования второй половины кодовой таблицы для кодирования кириллицы выходит за рамки данной книги, и здесь мы рассмотрим только пути реализации всех возможностей Apache с учетом современного состояния дел.

При этом в основу будут положены решения, предложенный Алексеем Тутубалиным, Михаилом Шойхером и Дмитрием Крюковым, которые вы всегда можете найти на сервере **apache.lexa.ru**.

3.1 Введение в проблему

Во-первых, отметим, что проблемы сортировки и русских имен файлов с точки зрения сервера WWW носят весьма условный характер. В самом деле, за поиск и отбор данных отвечают обычно либо связанные с сервером http системы управления базами данных[2], либо сценарии CGI или SSI. Поэтому для решения этих проблем используются либо локализационные таблицы СУБД, как это сделано, например в Progress, либо системные таблицы локализации самой системы[3]. При этом оба подхода реализуются за *пределами* сервера Apache и оказывают влияние на работу всей системы в

[2]Или серверы приложений, например, замечательная отечественная разработка — "Байконур"

[3]Известные в народе под названием "локаль"

целом, или, по крайней мере, приложений, которые используют сервер баз данных.

Второй аспект русификации, которым так гордится фирма Microsoft — *русские имена файлов* для сервера http не представляется актуальным по нескольким причинам:

- Далеко не все броузеры[4] способны воспринимать русские имена файлов, что не позволяет использовать в HTML-страницах ссылки на такие документы;

- Вполне вероятна ситуация, при которой на сервере хранится несколько версий документов на различных языках, в том числе и на русском. Но для обеспечения нормальной работы сервера имена документов могут отличаться только расширением! Удастся ли, скажем, англоязычному пользователю запросить документ, если содержащий его файл "обозван" кириллицей, а на терминале отстутствует даже переключатель **"Рус/Лат"**?

- "Война кодировок" тоже оказывает на решение этого вопроса свое влияние. Рассмотренные ниже средства "прозрачного кодирования" русских текстов не оказывают влияния на механизмы разбора имен файлов и рассматривают весь текстовый документ как единое целое — с постоянной кодировкой символов от 20 до 127 и с несколькими типами кодировки кириллицы. Это означает, что при трансляции документа произойдет автоматическая перекодировка имен всех "русских" файлов, ссылки на которые размещены в HTML-документе. Конечно приятно, что пользователь сможет понять смысл заложенный в названии, но сервер-то при запросе вернет код ошибки *404: File not Found*[5]. А "кодопрозрачных" файловых систем, пока что, увы, не существует.

---

[4]Хочется напомнить, что на сегодняшний день количество разработанных броузеров http составляет несколько десятков и не ограничивается фаворитами мира PC — *Netscape Navigator* и *Internet Explorer*

[5]Дело в том, что компьютеру наплевать на смысл, заложенный в названии файла, для идентификации используется последовательность байтов, составляющая имя, а именно она и была изменена во время перекодировки.

Таким образом, на долю сервера остается решение одной, но достаточно важной задачи — обеспечение автоматической адаптации к программнымм средствам пользователя с целью выдачи документов в приемлемой кодировке. Эта задача может решаться несколькими способами, все из которых поддерживаются сервером Apache:

- Через заголовки клиента *Accept-Charset:* и/или *Accept: text/x-cyrillic...* Если сервер поддерживает работу с кодировкой, которую запросил клиент, то эти заголовки имеют высший приоритет для сервера, вне зависимости от его настройки на используемые по умолчанию значения.

- Через поиск в *имени сервера* названия одной из сконфигурировнных кодовых страниц (например: www-koi8-r.st.ru или www-windows-1251.st.ru)

- Через поиск в *префиксе запрошенного URI* названия одной из зарегистрированных кодовых страниц, например

  `http://www.st.net/ibm866/file.html.`

- Через конфигурацию кодовых страниц по умолчанию для различных типов клиентских программ в случае, когда сервер может опознать клиентскую программу[6].

- Через текущее дерево. В этом случае администратор может запустить несколько серверов на разных портах, или описать в конфигурации сервера несколько виртуальных серверов. В обоих вариантах для разных реальных или виртуальных серверов устанавливается своя используемая по умолчанию кодировка. После этого администратор может планировать дерево в разных кодировках. Естественно, само дерево документов в таком случае хранится в единственном экземпляре.

[6]Или среду, в которой работает клиентская программа, что характерно для Microsoft Windows

- В случае принятия сообществом единой транспортной кодировки, необходимо будет оставить только ее в сервере, независимо от физического способа представления данных документов.

Кроме того, средства сервера позволяют реализовать "многоязычный режим хранения" документов, при котором каждой кодировке соответствует свой отдельный файл. Но поскольку все версии документов должны соотноситься с типом "cyrillic", для разрешения возникающих конфликтоов необходимо использовать специальный модуль, разработанный Алексеем Тутубалиным — **MultiCharset Htmls Module**, который может быть найден по адресу:

`http://www.lexa.ru/lexa/mod_mc_htmls.html`

Подробное описание этого модуля находится в главе, посвященной модулям расширения функций сервера.

3.2 Установка русификатора

На момент написания книги русификатор Apache представлял собой набор заплаток к исходному тексту сервера Apache версии 1.1.3, которая хорошо зарекомендовала себя и пользуется заслуженной популярностью. Вполне возможно, что когда вы держите книгу в руках, на упомянутом выше сервере вы сможете найти и более позднюю версию "русского сервера".

3.2.1 Распаковка исходных текстов

Использование традиционного для UNIX способа доработки программного обеспечения с помощью механизма *заплаток*[7] привело к необходимости распространения либо полного исходного текста сервера либо только исходного текста исправлений. Поэтому, обратившись по адресу **apache.lexa.ru** вы сможете найти как полный

[7] Patch

исходный текст Apache 1.1.3 со всеми необходимыми исправлениями, так и только тексты заплаток в стандартном формате, воспринимаемом программой **patch**.

В первом случае извлеките архивный файл с "русифицированным" исходным текстом, и распакуйте его стандартным образом с помощью tar и zcat:

```
zcat apache-1.1.3-rusPL14.tar.gz | tar xvf -
```

Если же вы уже имеете в своем распоряжении архивный файл с исходными текстами сервера, то вам достаточно получить файл с заплатками и применить их стандартным образом[8]:

```
zcat apache_1.1.3.tar.gz | tar xvf -
cd apache_1.1.3
zcat /path/to/patches_1.1.3rusPL15.gz | patch -p1
```

Обратите внимание, что при распаковке архивных файлов с исходным кодом сервера создается специальный подкаталог, в который помещается все дерево с исходными текстами программ, дистрибутивными конфигурационными файлами и так далее. Поэтому вам достаточно выполнять первые операции по распаковке, находясь в каталоге */usr/src*.

И еще один момент... Мне не очень нравятся сложные длинные имена каталогов, которые хотя и поясняют, что в них содержится, но оказываются слишком неудобными в практическом использовании. Поэтому вполне резонным может оказаться способ, используемый при установке ядра Linux:

```
cd /usr/src
ln -sf apache_1.1.3 /usr/src/apache
```

3.2.2 Настройка перед компиляцией

Теперь, прежде чем приступить к компиляции программы, необходимо настроить некоторые параметры сервера. Перейдите в каталог

[8] вам понадобится программа patch

/where/is/apache_1_1_3.../src

и отредактируйте файл Configuration, как это было рассказано в главе 1. Большинство проблем и сообщений "Russian Apache не работает"; связаны именно с пропуском этого шага. Если вы хотите "зашить"; в httpd путь до $WWW_ROOT отличный от /usr/local/etc/httpd, то для этого нужно внести необходимые изменения в файл httpd.h. Но это вовсе необязательно, поскольку переменную $WWW_ROOT можно указывать и в командной строке httpd.

3.2.3 Компиляция

Процесс компиляции ничем не отличается от рассмотренного нами ранее. Запустите ./Configure, а затем make.

Начальная конфигурация

Создайте директорию (если ее еще нет), в которой будут находиться конфигурационные файлы WWW-сервера. Сделайте там поддиректорию conf. Скопируйте в **conf/** конфигурационные файлы "Russian Apache", переименовав *.conf-dist в *.conf, отредактируйте *.conf в соответствии с вашей конфигурацией[9]. Подробное описание директив управления "русской" частью сервера Apache приведено в **********.

Пробный запуск

Попробуйте запустить собранную программу:

```
/path/to/httpd -d /your/www/root
```

Если сервер запустится без ошибок, то скорее всего все предшествующие шаги совершены успешно и сервер готов к работе.

[9]Реально нужно поправить пути и имена файлов

3.3 Практические приемы

3.3.1 Раздача слонов

Почему то большинство ошибок связано с предположением о том, что директива русифицированного сервера NativeCharset описывает кодировку, в которой сервер хранит документы на диске. Но это не так! NativeCharset всего лишь описывает кодировку, в которой документ будет показан клиенту в том случае, когда ни одним из стандартных способов[10] не удается. Для того, чтобы изменить представление информации, выдаваемой клиенту, необходимо переопределить описания таблиц кодировок. Эта задача может решаться как для сервера в целом, так и для отдельных виртуальных серверов. В качестве примера рассмотрим конфигурацию сервера с кодировкой документов на диске CP1251[11].

3.4 Модуль MultiCharset HTML

3.4.1 Использование возможностей mime

Входящий в состав дистрибутивного комплекта сервера Apache модуль **mime_module** поддерживает ряд директив, которые могут оказаться полезными при использовании совместно с модулем **MultiCharset Htmls Module**.

Подробное описание команд данного модуля приведено в .....

AddType ...

Эта директива используется для указания MIME-type файла с нестандартным расширением. Команда может быть использована как на server-wide уровне, так и для директорий, для которых разрешен режим Override FileInfo.

Пример:

```
AddType text/html .htm8 .htm1251 .htmlw
```

[10]Через заголовок Accept:. URI Prefix или User-Agent

[11]Автор примера — Mike Shoyher (msh@alina.ru)

Эта строка заставляет сервер воспринимать файлы с расширением .htm8, .htm1251, .htmlw как имеющие тип text/html (то есть так же как и *.html). Это нужно делать по двум причинам. Во-первых, от mime-type зависит интерпретация файлов броузером, во-вторых, обработчик по-умолчанию (при включенной опции **RecodeAllFiles** обрабатывает только файлы с типом text/html).

AddHandler server-recoded...

Эта директива указывает, что для файлов имеющих расширения *extension1, extension2* и так далее нужно использовать обработчик под именем **server-recoded**[12]. Это второй[13] способ указать серверу какие именно файлы являются предметом перекодировки.

Например:

```
RecodeAllFiles off
AddHandler server-recoded .htmlw
```

указывает серверу на необходимость перекодировать только файлы с расширением .htmlw. Как именно (по какой таблице) их перекодировать определяется директивами
UseFileCharset и *DefaultFileCarset*

3.5 Примеры работы с mod_mc_htmls

В этом параграфе приведено несколько примеров конфигурирования сервера в режиме с учетом возможностей, предоставляемых модулем MultiCharSet.

3.5.1 Простейший пример

Вначале рассмотрим самый простой случай, при котором нам потребуется перекодировать только файлы с расширением **.htmlw**, которые содержат документы в кодировке Windows в кодировку

[12]Название обработчика зарезервировано для модуля mod_mc_htmls.
[13]Кроме директивы **RecodeAllFiles**.

КОИ-8. Предполагается, что остальные файлы в перекодировке не нуждаются.

```
AddType text/html .htmlw
FileCharset cp1251 conf/win-koi.tab
UseFileCharset  cp1251 .htmlw
AddHandler server-recoded .htmlw
```

Как видно из приведенного выше фрагмента, вначале производится регистрация расширения имени файлов **.htmlw** как документов с mime-типом *text/html*. Затем осуществляется загрузка таблицы перекодиовки и устанавливается ее соответствие введенному расширению имени файла. Теперь остается только указать, что перекодировка этих документов будет осуществляться с помощью обработчика, встроенного в модуль.

3.5.2 Два типа перекодируемых документов

В этом примере мы заставим сервер перекодировать файлы, имеющие mime-тип text/html, но по-разному: файлы .htm — по таблице win-koi, а все остальные — по таблице koi-koi. Смысл последнего преобразования состоит в "выкусывании" тегов **META HTTP-EQUIV**.

```
FileCharset koi8 conf/koi-koi.tab
FileCharset cp1251 conf/win-koi.tab
UseFileCharset cp1251 .htm
DefaultFileCharset koi8
RecodeAllFiles on
StripHttpEquivs on
```

В приведенном фрагменте определяется две таблицы перекодировки: koi8 и cp1251. Затем

Часть III

Справочное руководство

Глава 4

Компиляция и установка Apache

4.1 Выгрузка Apache из сети Internet

Информация о последней версии Apache может быть найдена по адресу:

http://www.apache.org.

На этом узле вы сможете найти последнюю выпущенную версию сервера, следующую версию, находящуюся на этапе тестирования, а также необходимую справочную информацию относительно зеркальных узлов (mirror sites) и ftp-серверов.

4.2 Компиляция Apache

Данная версия Apache реализует концепцию "дополнительных модулей", однако для ее использования сервер должен знать, какие модули скомпилированы вместе с ним. Поэтому при установке сервера генерируется небольшой файл `modules.c`, который упрощает последующую работу с модулями.

Если вас вполне устраивает стандартный набор модулей, и вы не планируете устанавливать какие-либо дополнительные функции, вам достаточно внести незначительные изменения в `Makefile` и собрать продукт стандартным образом. Если же вы планируете подключить дополнительные модули, вам придется запустить конффигурирующий скрипт-файл.

Для этого:

Внесите необходимые изменения в файл `Configuration`, который содержит специфичные для вашего компьютера настройки для `Makefile`. Кроме того в этом файле (в самом конце) имеется специальная секция, которая перечисляет компилируемые в составе сервера модули и файлы, в которых эти модули содержатся. Поэтому вам потребуется сделать следующие операции:

Выберите компилятор и опции этапа трансляции, специфичные для вашего компьютера;

Уберите символы комментария с дополнительных модулей, которые вы хотели бы включить в состав сервера (это строки в конце файла, имеющие идентификатор `Module`) или добавьте новые строки, которые соответствуют модулям, написанным вами.

Обратите внимание, что если вам необходимо установить DBM-аутентификацию, то соответствующий модуль должен быть явно включен в состав сервера; для этого вам достаточно просто снять символ комментария с соответствующей строки.

Запустите скрипт-файл `Configure`:

```
% Configure
Using 'Configuration' as config file
\%
```

В результате будет созданы новые версии файлов **Makefile** и **modules.c**. Если же вам необходимо создать и поддерживать несколько различных конфигураций сервера, вы можете использовать, например, следующую конструкцию:

```
% Configure -file Configuration.ai
Using alternate config file Configuration.ai
%
```

Введите команду **make**.

Модули, которые мы поместили в дистрибутив Apache, относятся к полностью протестированным и регулярно используемым различными членами группы разработчиков проекта Apache. Дополнительные модули, созданные третьими лицами для решения каких-либо специальных задач могут быть найдены по адресу:

```
<URL:http//www.apache.org/dist/contrib/modules>
```

На этой же странице расположены и все инструкции, необходимые для включения этих модулей в ядро Apache.

4.3 Установка Apache

Компиляция завершается формированием исполняемого файла httpd, который будет помещен в каталоге src/. При использовании бинарного дистрибутива Apache поставляется именно этот файл.

Следующим шагом в настройке сервера является редактирование конфигурационных файлов сервера.

В каталоге *conf* помещены дистрибутивные версии этих файлов (представлены слева), которые необходимо скопировать для получениярабочих файлов (справа):

```
srm.conf-dist        ---->    srm.conf
access.conf-dist     ---->    access.conf
httpd.conf-dist      ---->    httpd.conf
```

Вначале необходимо отредактировать **httpd.conf**. В этом файле устанавливаются основные параметры сервера: номер порта, пользователи, которые могут пользоваться услугами сервера и так далее. Затем необходимо внести правки в файл **srm.conf**, который определяет корневой каталог дерева документов, специальных функций, таких как генерация сервером HTML-документов, внутренний разбор битовых карт и так далее. И в заключение, вам потребуется отредактировать файл **access.conf** чтобы установить типовые схемы доступа к данным.

После этого вы должны вызвать **httpd**, указав полный путь к конфигурационному файлу *httpd.conf*. В общем случае:

```
/usr/local/etc/apache/src/httpd \
  -f /usr/local/etc/apache/conf/httpd.conf
```

Если все конфигурационные файлы были настроены правильно, сервер запустится и перейдет в фоновый режим.

По умолчанию, файлы **srm.conf** и **access.conf** располагаются в том же каталоге, что и основной конфигурационный файл. Поэтому, в том случае, когда вы хотите использовать иные файлы или пути доступа, вы должны поместить в **httpd.conf** специальные директивы `AccessConfig` и `ResourceConfig`.

Глава 5

Запуск Apache

5.1 Вызов Apache

Программа httpd вызывается либо суперсервером **inetd** каждый раз при поступлении запроса на соединение со службой HTTP, либо она может быть запущена как демон, который выполняется непрерывно, обслуживая запросы пользователей. Какой бы метод вы не выбрали, необходимо установить параметр директивы `ServerType`, которая определит соответствующий режим работы.

5.2 Опции командной строки

При запуске программа **httpd** распознает следующие опции:

-d *serverroot* Устанавливает значение переменной *ServerRoot* в *serverroot*. Это значение может быть затем переустановлено командой **ServerRoot** в конфигурационном файле. По умолчанию используется значение `/usr/local/etc/httpd`.

-f *config* При запуске программы выполняет последовательность команд, содержащуюся в файле *config*. Если имя *config* не начинается с символа /, то используется относительный путь доступа, начинающийся с **ServerRoot**. По умолчанию используется значение `conf/httpd.conf`.

-X Запуск сервера в режиме одного процесса, используется только при отладке; демон не отрывается от консольного терминала и не порождает потомков. ***Не используйте этот режим для штатного обслуживания HTTP-запросов!***

-v Печатает на терминале версию httpd и завершает работу программы.

-? Выводит на стандартный вывод список опций httpd и завершает работу.

5.3 Конфигурационные файлы

Сервер при инициализации считывает три конфигурационных файла. В любом из этих файлов может использоваться любая из перечисленных ниже директив. Имена файлов формируются с учетом пути доступа, установленного директивой **ServerRoot** или опцией командной строки **-d**. В большинстве случаев в качестве конфигурационных файлов используются:

conf/httpd.conf Содержит директивы, которые управляют работой демона сервера. Это имя может быть переопределено с помощью опции **-f**.

conf/srm.conf Содержит директивы, управляющие спецификацией документов, которые сервер может предоставлять пользователям. Имя этого файла может быть переопределено директивой *ResourceConfig*.

conf/access.conf Содержит директивы, управляющие режимом доступа к документам. Имя файла устанавливается директивой *AccessConfig*.

Как уже указывалось, строго следовать этим соглашениям вовсе не обязательно.

Сервер считывает также при загрузке файл, содержащий типы документов класса mime; имя этого конфигурационного файла устанавливается директивой *TypesConfig* и по умолчанию равно `conf/mime.types`.

5.4 Файлы журналов

5.4.1 PID-файл

Демон помещает идентификатор родительского процесса httpd в **logs/httpd.pid**. Это имя файла может быть изменено с помощью директивы *PidFile*. Идентификатор процесса используется администратором при перезапуске или остановке сервера; сигнал `HUP` заставляет демон повторно считать свои конфигурационные файлы, а сигнал *TERM* инициализирует процесс корректной выгрузки демона из памяти.

Если же программа будет завершена аварийно, необходимо самостоятельно уничтожить все дочерние httpd-процессы.

5.4.2 Журнал сообщений об ошибках

Сервер осуществляет регистрацию сообщений об ошибках в файл **logs/error-log**. Имя файла может быть изменено с помощью директивы *ErrorLog*. Кроме того, для различных виртуальных узлов WWW могут быть назначены различные журналы.

5.4.3 Журнал учета передачи данных

Сервер обычно регистрирует все запросы на передачу файлов пользователям в специальном журнале — **logs/access-log**. Имя журнала может быть назначено с помощью директивы *TransferLog*; каждому из виртуальных узлов WWW может быть установлен отдельный файл журнала учета.

Глава 6

Ядро сервера Apache

В данной главе рассматриваются средства настройки ядра сервера httpd, которые доступны в любой конфигурации программы.

6.1 Директива AccessConfig

Синтаксис:	AccessConfig *имя файла*
Значение по умолчанию:	**conf/access.conf**
Контекст:	конфигурация сервера
Режим:	ядро

Сервер считывает файл, указанный этой директивой после загрузки основного конфигурационного файла *ResourceConfig*. *Имя файла* приводится относительно значения `ServerRoot`. При необходимости обработка файла конфигурации доступа может быть отключена путем указания:

AccessConfig /dev/null

По историческим причинам этот файл содержит обычно только секцию ***«Directory»***; хотя в него может быть помещена любая директива настройки сервера, разрешенная в контексте *конфигурация сервера*.

6.2 Директива AccessFileName

Синтаксис:	AccessFileName *имя файла*
Значение по умолчанию:	**.htaccess**
Контекст:	конфигурация сервера
Режим:	ядро

При отправке документа, запрошенного пользователем узла сервер проверяет содержимое файла контроля доступа. Если управление доступом включено для каталога, в котором расположен запрашиваемый файл, сервер будет просматривать все дерево каталогов на предмет инструкций, разрешающих или запрещающих передачу данного файла.

Например, строка:

AccessFileName .acl

определяет, что прежде, чем отправить файл */usr/local/web/index.html* сервер считывает файлы: **.acl, /usr/.acl, /usr/local/.acl а также /usr/local/web/.acl** и проанализирует их содержание. Этот режим может быть отключен с помощью следующей конструкции:

```
<Directory>
AllowOverride None
</Directory>
```

6.3 Директива AllowOverride

Синтаксис:	AllowOverride *override...*
Значение по умолчанию:	**AllowOverride all**
Контекст:	каталог
Режим:	ядро

Если сервер обнаруживает файл контроля доступа .htaccess (или иной, определяемый директивой AccessFileName), необходимо определить, какие из директив, определенных в этом файле могут изменить ранее установленные правила доступа к данным.

Override может принять значение `None`, и в этом случае сервер просто не будет считывать этот файл, или `All`, что заставит сервер следовать всем директивам, записанным в каждом каталоге дерева доступа. Но это крайние случаи, а для более гибкого реагирования сервера на запросы можно использовать еще несколько вариантов значений директивы *AllowOverride*:

AuthConfig

Разрешает использование директив авторизации доступа:

```
AuthDBMCroupFile,    AuthDBMUserFile,
AuthGroupFile,       AuthName,
AuthType,            AuthUserFile.
```

FileInfo

Позволяет использовать директивы, управляющие содержимым документов (*AddEncoding, AddLanguage, AddType, DefaultType и LanguagePriority*).

Indexes

Позволяет использовать директивы, управляющие индексацией каталогов (*AddDescriptio, AddIcon, AddIconByEncoding, AddIconByType, DefaultIcon, DirectoryIndex, FancyIndexing, HeaderName, IndexIgnore, IndexOptions и ReadmeName*).

Limit

Позволяет использовать директивы, регламентирующие доступ к серверу со стороны отдельных хостов (*allow, deny и order*).

Options

Позволяет использовать директивы, управляющие особенностями доступа к отдельным каталогам (*Options и XBitHack*).

6.4 Директива AuthName

Синтаксис:	AuthName *auth_domain*
Контекст:	каталог, .htaccess
Переопределяет:	Authconfig
Режим:	ядро

Эта директива устанавливает параметры авторизации для каталога. Эти параметры определяют, какое имя пользователя и пароль должны отправляться для получения доступа к данным. Данная директива должна использоваться совместно с *AuthType* и директивами require, а также директивами *AuthUserFile* и *AuthGroupFile*.

6.5 Директива AuthType

Синтаксис:	AuthType *type*
Контекст:	каталог, .htaccess
Переопределяет:	AuthConfig
Режим:	ядро

Эта директива выбирает тип аутентификации пользователя для конкретного каталога. В настоящее время реализован только один тип доступа – `Basic`. Директива может использоваться совместно с AuthName и директивами require, а также директивами AuthUserFile и AuthGroupFile.

6.6 Директива BindAddress

Синтаксис:	BindAddress *saddr*
Значение по умолчанию:	**BindAddress ***
Контекст:	сервер, конфигурация
Режим:	ядро

Сервер http в UNIX-системах может работать в двух режимах приема входящих соединений: либо анализировать запросы, поступающие от всех IP-адресов, назначенных машине, либо принимать запросы только по выделенному специально для сервера IP-адресу. При этом *Saddr* может принимать следующие значения:

- *
- IP-адрес
- полное доменное имя сервера http

Если значением *Saddr* является *, осуществляется прием сообщений по любым IP-адресам, используемым компьютером, во всех остальных случаях сервер реагирует только на определенный адрес (или имя).

Эта опция может использоваться как альтернативный метод поддержки виртуальных хостов вместо секции «*VirtualHost*».

6.7 Директива DefaultType

Синтаксис:	DefaultType *mime-type*
Значение по умолчанию:	`DefaultType text/html`
Контекст:	сервер, виртуальный хост, каталоги, .htaccess
Переопределяет:	Fileinfo
Режим:	ядро

Возможны ситуации, когда сервер должен предоставить пользователю документ, тип которого не может быть определен с помощью карт MIME. В этом случае сервер должен информировать клиента о типе содержания документа, и в случае неизвестного типа использовать **DefaultType**. Например:

DefaultType image/gif

будет вполне адекватным для каталога, в котором находится большое количество GIF-изображений в файлах, имена которых не содержат расширения *.gif*.

6.8 Директива «Directory»

Синтаксис:	«Directory *directory* » ...«/Directory»
Контекст:	сервер, конфигурация, виртуальные хосты
Режим:	ядро

«Directory» и «/Directory» используются для формирования групп директив, которые применяются только для указанного каталога и его поддерева. В этой группе может использоваться любая директива, допустимая для контекста *"Каталоги"*. *Directory* может представлять собой либо полный путь доступа к файлу, либо строку с символами заполнителями. При этом заполнитель "?" соответствует любому одиночному символу, а заполнитель "*" — любой последовательности символов. Например:

```
<Directory /usr/local/httpd/htdocs>
```

```
Options Indexes FollowSymLinks
</Directory>
```

Если выясняется, что каталогу (или его родителям), содержащему документ соответствуют несколько секций *Directory*, то директивы из этих секций применяются последовательно, начиная с наиболее короткого пути, перемежаясь директивами из файлов .htaccess. Например:

```
<Directory />
AllowOverride None
</Directory>

<Directory /home/*>
AllowOverride FileInfo
</Directory>
```

В результате при попытке доступа по адресу **/home/web/dir/doc.html** выполняются следующие проверки для определения прав доступа:

- Применяется директива:
  AllowOverride None
  (отключающая использование файлов **.htaccess**).
- Применяется директива:
  AllowOverride FileInfo (только для каталога **/home/web**).
- Применяются директивы **FileInfo**, находящиеся в файле **/home/web.htaccess**.

Секции каталогов обычно размещаются в файле *access.conf*, но могут также появляться и в любом другом конфигурационном файле. Директивы «Directory» не могут вкладываться друг в друга и не могут появляться в секции «Limit».

6.9 Директива DocumentRoot

Синтаксис:	DocumentRoot *directory-filename*
Значение по умолчанию:	**/usr/local/etc/httpd/htdocs**
Контекст:	сервер, конфигурация, виртуальные хосты
Режим:	ядро

Эта директива устнаваливает имя каталога, откуда httpd предоставляет пользователям файлы. До тех пор, пока имя файла не совпадает с директивой Alias, сервер дописывает путь к файлу из запрошенного URL к корню документа и таким образом формирует полное имя файла. Например:

DocumentRoot /usr/web

В этом случае запрос документа, размещенного по адресу `http://www.my.host.com/index.html` приведет к загрузке файла **/usr/web/index.html**.

6.10 Директива ErrorDocument

Синтаксис:	ErrorDocument *error-code document*
Контекст:	сервер, конфигурация, виртуальные хосты
Режим:	ядро

В случае возникновения ошибок или нарушения прав доступа Apache может выполнить одно из четырех действий:

- Вести себя так же, как и NCSA httpd 1.3
- вывести сообщение, заданное пользователем
- перенаправить локальный URL на обработку ошибочной ситуации
- перенаправить пользователя на удаленный URL

Последние три типа реакции задаются при помощи директивы **DocumentError**, за которой следуют код ошибки, генерируемый в соответствии с протоколом HTTP и текст сообщения или URL.

Сообщения в данном случае начинаются с кавычек ("), которые не входят в состав самого текста. Apache в ряде случаев также предоставляет некоторую информацию об ошибках, которая может быть включена в состав сообщения с помощью спецификатора формата %s.

URL должен начинаться с символа косой черты (/) для локальных ссылок, или же представлять собой полный адрес, который обрабатывается клиентским компьютером. Например:

```
ErrorDocument 500  /cgi-bin/tester
ErrorDocument 404 /cgi-bin/bad_urls.pl
ErrorDocument 401 http://www2.foo.bar/subs_info.html
ErrorDocument 403 "Sorry, can't allow you access today"
```

Смотри также: создание заказных реакций нештатные ситуации.

6.11 Директива ErrorLog

Синтаксис:	ErrorLog *имя файла*
Значение по умолчанию:	**ErrorLog logs/error_log**
Контекст:	сервер, конфигурация, виртуальные хосты
Режим:	ядро

Директива **ErrorLog** устанавливает имя файла, в который сервер записывает все произошедшие во время работы ошибки. Если указанное имя файла не начинается с символа косой (/), то оно является относительным с корнем в **ServerRoot**. Например:

ErrorLog /dev/null

просто напросто выключает механизм регистрации ошибок, направляя весь вывод в "мусорный бак".

6.12 Директива Group

Синтаксис:	Group *unix-group*
Значение по умолчанию:	Group #-1
Контекст:	сервер, конфигурация
Режим:	ядро

Директива **Group** устанавливает номер группы, в которой будет работать сервер, отвечая на запросы. чтобы использовать эту директиву, автономный сервер первоначально должен быть запущен как `root`. *Unix-group* может принимать одно из значений:

Имя группы

Должно принадлежать к одной из групп, определенных в файле `/etc/group`.

 # за которым следует номер группы

Ссылка на номер группы (существующей!)

Рекомендуется создать специальную группу для работы сервера. Некоторые администраторы используют пользователя `nobody`, но это оказывается не всегда возможным или желательным.

`Примечание`: Если вы запускаете сервер с именем, отличным от root, то при попытке смены группы будет зафиксирована ошибка и сервер продолжит функционирование с прежними идентификаторами пользователя/группы.

`БЕЗОПАСНОСТЬ` См. раздел User относительно аспектов защиты от несанкционированного доступа.

6.13 Директива IdentityCheck

Синтаксис:	IdentityCheck *boolean*
Значение по умолчанию	`IdentityCheck off`
Контекст:	сервер, конфигурация
Режим:	ядро

Эта директива разрешает совместимую с RFC-931 регистрацию имени удаленного пользователя при установлении каждого соединения, где клиентская машина использует суперсервер inetd или подобное ему программное обеспечение. Эта информация фиксируется в журнале доступа. *Boolean* может принимать значения `on` или `off`.

Полученная информация не может считаться абсолютно достоверной и не предназначена для какого-либо использования, отличного от рудиментарной регистрации пользовательской активности.

6.14 Директива «Limit»

Синтаксис:	«Limit *method method* » ...«/Limit»
Контекст:	любой
Режим:	ядро

Пара директив «Limit» и «/Limit» используется для формирования групп управления доступом, которые затем применяются только к указанным методам доступа, где *method* представляет собой любой разрешенный HTTP метод. В группе может быть использована любая директива, за исключением «Limit» или «Directory»; впрочем следует отметить, что на большинство из них «Limit» никакого влияния не оказывает. Пример использования:

```
   <Limit GET POST>
require valid-user
</Limit>
```

Если директива контроля доступа появляется за пределами группы «Limit», она воздействует на все методы досутпа.

6.15 Директива MaxClients

Синтаксис:	MaxClients *количество*
Значение по умолчанию:	`MaxClients 150`
Контекст:	сервер, конфигурация
Режим:	ядро

Директива `MaxClients` устанавливает предельное количество запросов, обслуживаемых одновременно; ни при каких условиях сервер не породит больше дочерних процессов, чем указано в этой директиве.

6.16 Директива MaxRequestsPerChild

Синтаксис:	MaxRequestsPerChild *количество*
Значение по умолчанию:	`MaxRequestsPerChild 0`
Контекст:	сервер, конфигурация
Режим:	ядро

Директива `MaxRequestsPerChild` устанавливает предельное количество запросов, которое может обработать один дочерний процесс. После MaxRequestsChild запросов дочерний процесс будет завершен. Если же это значение равно нулю, то процесс не ведет учета количества запросов.

Установка нулевого значения позволяет добиться следующих результатов:

- ограничивает объем памяти, потребляемой процессом (в случае неверного распределения или освобождения оперативной памяти);
- путем ограничения времени жизни процесса позволяет уменьшить количество процессов в памяти при снижении нагрузки на сервер.

6.17 Директива MaxSpareServers

Синтаксис:	MaxSpareServers *количество*
Значение по умолчанию:	`MaxSpareServers 10`
Контекст:	сервер, конфигурация
Режим:	ядро

Эта директива устанавливает максимальное число *простаивающих* дочерних процессов сервера. Под простаивающим понимается процесс, который в данный момент времени не обслуживает запросы пользователей. Если в какой-то момент времени простаивает процессов больше, чем указано в MaxSpareServers, родительский процесс завершит лишние.

Корректировка данного параметра необходима только на очень занятых узлах. При этом следует иметь в виду, что присвоение больших числовых значений почти всегда является очень плохой идеей.

См. также MinSpareServers и StartServers.

6.18 Директива MinSpareServers

Синтаксис:	MinSpareServers *количество*
Значение по умолчанию:	`MinSpareServers 5`
Контекст:	сервер, конфигурация
Режим:	ядро

Директива `MinSpareServers` устанавливает необходимое минимальное количество *простаивающих* дочерних процессов сервера. Если их количество меньше, чем установленное данной директивой, родительский процесс инициализирует недостающее количество дочерних серверов. Интервал опроса состояния составляет одну секунду.

Корректировка этого параметра должна осуществляться только на очень загруженных узлах. Попытка установить большое значение этого параметра в большинстве случаев является весьма плохой идеей.

См. также MaxSpareServers и StartServers.

6.19 Директива Options

Синтаксис:	Options *опции опции* ...
Контекст:	сервер, конфигурация, каталоги, .htaccess
Переопределяет:	Опции
Режим:	ядро

Дтректива `Options` определяет, какие из возможностей и особенностей сервера используются в конкретном каталоге. *Опция* может принять значение **None**, в этом случае не допускается использование ни одной дополнительной возможности сервера, или же принять одно из следующих значений:

All

Все опции за исключением используемых в режиме MultiViews.

ExecCGI

Разрешается выполнение скриптов CGI.

FollowSymLinks

Сервер будет отслеживать символьные связи в текущем каталоге.

Includes

Разрешается включение файлов на стороне сервера.

IncludesNOEXEC

Включение файлов на сервере разрешено, но команды CGI #exec и #include заблокированы.

Indexes

Если URL отображается на запрашиваемый пользователем каталог, и в данном каталоге отсутствует файл индекса каталога (например, index.html), то сервер возвращает в ответ на запрос форматированный листинг данного каталога.

MultiViews

Разрешается согласование протоколов обмена при работе в режиме MultiViews.

SymLinksIfOwnerMatch

Сервер обрабатывает символьные ссылки только для тех файлов или каталогов, владельцем которых является тот же пользователь, что и владелец символьной связи.

В случае, если к одному и тому же каталогу применяется несколько директив Options используется та, которая указывает на наиболее длинный путь — никакого слияния опций не происходит. Например:

```
   <Directory /web/docs>
Options Indexes FollowSymLinks
</Directory>
<Directory /web/docs/spec >
Options Includes
</Directory>
```

В этом случае для каталога /web/docs/spec будет установлена только опция `Includes`.

6.20 Директива PidFile

Синтаксис:	PidFile *имя файла*
Значение по умолчанию:	`PidFile logs/httpd.pid`
Контекст:	сервер, конфигурация
Режим:	ядро

Директива `PidFile` устанавливает имя файла, в который сервер записывает идентификатор процесса демона. Если имя файла не начинается с косой (/), используется относительная адресация с корнем, определенным директивой ServerRoot. Директива `PidFile` может использоваться только в автономном режиме работы сервера.

Часто оказывается полезным иметь возможность подать серверу сигнал, который заставит Apache закрыть и вновь открыть файлы ErrorLog и TransferLog, а также повторно прочитать конфигурационные файлы. Эта операция может быть выполнена с помощью сигнала `SIGHUP (kill -1)` процессу, идентификатор которого записан в PID-файле.

6.21 Директива Port

Синтаксис:	Port *номер*
Значение по умолчанию:	`Port 80`
Контекст:	сервер, конфигурация
Режим:	ядро

Директива `Port` устанавливает номер сетевого порта, который используется для установки соединения с сервером. *Номер* представляет собой число в диапазоне от 0 до 65535; при этом необходимо учитывать, что некоторые номера портов (в частности те, которые меньше 1024) зарезервированы для конкретных протоколов. Более подробную информацию о назначении портов можно найти в файле `/etc/protocols`. Стандартным номером порта для http-протокола является порт 80.

Этот порт относится к числу специальных портов Unix. Все порты, номера которых меньше 1024 являются системными портами UNIX — это означает, что обычные пользователи (не имеющие прав root) не могут их использовать.

Поэтому для того, чтобы использовать порт 80, вы должны запустить сервер с привилегиями root. Затем Apache может изменить свои права в соответствии с директивой User.

Если вы не имеете возможности использовать порт 80, воспользуйтесь любым незанятым портом. Обычные пользователи должны выбирать номера портов за пределами 1024, например — 8000.

БЕЗОПАСНОСТЬ: Если вы запускаете сервер как суперпользователь (root), позаботьтесь о том, чтобы `не` установить User значение root. Игнорирование этого правила означает, что пользователь, соединившийся с вашим сервером получает права доступа root.

6.22 Директива require

Синтаксис:	require *entity-name entity entity* ...
Контекст:	каталоги, .htaccess
Переопределяет:	AuthConfig
Режим:	ядро

Эта директива определяет, кто из аутентифицированных пользователей имеет право доступа к каталогу. Допустимый синтаксис директивы:

- require user *userid userid* ...

  Только перечисленные пользователи имеют право доступа к каталогу.

- require group *groupname groupname* ...

  Только пользователи из перечисленных групп имеют право доступа к каталогу.

- require valid-user

Все легальные пользователи имеют право доступа к каталогу.

Если `require` используется в секции «Limit», то эта директива ограничивает доступ к указаннм методам, в противном случае ограничивается использование всех методов. Например:

```
AuthType Basic
AuthName somedomain
AuthUserFile /web/users
AuthGroupFile /web/groups
Limit <GET POST>
require group admin
</Limit>
```

Require должна использоваться совместно с директивами AuthName и AuthType, и такими директивами как AuthUserFile или
AuthGroupFile (определяющими пользователей и группы).

6.23 Директива ResourceConfig

Синтаксис:	ResourceConfig *имя файла*
Значение по умолчанию:	`ResourceConfig conf/srm.conf`
Контекст:	сервер, конфигурация, виртуальные хосты
Режим:	ядро

Сервер считывает этот файл в поисках дальнейших управляющих директив после того, как обработает файл httpd.conf. *Имя файла* адресуется относительно ServerRoot. Эта функция может быть отключена путем использования следующего значения:

```
ResourceConfig /dev/null
```

По историческим причинам этот файл содержит большую часть управляющих директив, за исключением директив конфигурации сервера и секций «Directory». Реально же вы можете поместить в него любые директивы, разрешенные для использования в контексте конфигурации сервера.

См. также AccessConfig

6.24 Директива ServerAdmin

Синтаксис:	ServerAdmin *e-mail*
Контекст:	сервер, конфигурация, виртуальные хосты
Режим:	ядро

Директива `ServerAdmin` устанавливает адрес электронной почты, который включается сервером во все сообщения об ошибках, которые возвращаются клиенту.

Имеет смысл выделять для этой цели специальный адрес, поскольку пользователи далеко не всегда вставляют в письмо упоминания о том, что они говорят о сервере WWW!

6.25 Директива ServerName

Синтаксис:	ServerName *полное доменное имя*
Контекст:	сервер, конфигурация, виртуальные хосты
Режим:	ядро

Директива `ServerName` устанавливает имя хоста, соответствующее серверу и используется только при перенаправлении URL. Если это имя не указано, сервер пытается определить его исходя из своего собственного IP-адреса; однако это может и не сработать ... Например, имя:

```
ServerName www.wibble.com
```

должно использоваться в том случае, если основное имя машины `monster.wibble.com`.

6.26 Директива ServerRoot

Синтаксис:	ServerRoot *каталог*
Значение по умолчанию:	`/usr/local/etc/httpd`
Контекст:	сервер, конфигурация
Режим:	ядро

Директива `ServerRoot` устанавливает имя каталога, в котором находятся файлы, необходимые для работы сервера. Как правило, данный каталог содержит два подкаталога: `conf/` и `logs`. Относительные пути к другим каталогом формируются с использованием данного каталога как корневого.

6.27 Директива ServerType

Синтаксис:	ServerType *type*
Значение по умолчанию:	`ServerType standalone`
Контекст:	сервер, конфигурация
Режим:	ядро

Эта директива устанавливает режим работы сервера в системе. *Type* может принимать одно из следующих значений:

- **inetd**

  Сервер будет запущен системным процессом inetd; команда для запуска сервера добавляется в управляющий файл `/etc/inetd.conf`.

- **standalone**

  Сервер запускается как процесс-демон; команда для запуска сервера помещается в системные инициализационные файлы (`/etc/rc.local` или `/etc/rc3.d/` ...).

`Inetd` используется гораздо реже, поскольку для каждого входящего http-соединения порождается новый экземпляр сервера, и после того как соединение завершено, задача снимается. Этот метод управления процессами оказывается слишком дорогим с точки зрения производительности в целом, но его предпочитают некоторые администраторы, заботящиеся о повышенной безопасности узла WWW.

`Standalone` используется значительно чаще. Сервер запускается один раз и обслуживает все входящие соединения. Если вы планируете использовать Apache для обслуживания интенсивно посещаемого узла, режим `standalone` для вас наиболее предпочтителен.

БЕЗОПАСНОСТЬ: если вы параноик во всем, что касается вопросов безопасности, запускайте сервер в режиме `inetd`. В противном случае невозможно гарантировать безопасность. И хотя большую часть пользователей вполне удовлетворяет режим `standalone`, загрузка с помощью суперсервера считается более защищенной по отношению к внешним атакам.

6.28 Директива StartServers

Синтаксис:	StartServers *количество*
Значение по умолчанию:	`StartServers 5`
Контекст:	сервер, конфигурация
Режим:	ядро

Директива `StartService` устанавливает количество дочерних процессов, которые создаются при запуске программы. Поскольку количество этих процессов постоянно изменяется в зависимости от загрузки узла, в большинстве случаев нет необходимости в изменении этого параметра.

См.также `MinSpareServers` и `MaxSpareServers`

6.29 Директива TimeOut

Синтаксис:	TimeOut *количество*
Значение по умолчанию:	`TimeOut 1200`
Контекст:	сервер, конфигурация
Режим:	ядро

Директива `TimeOut` определяет временной интервал (в секундах), в течение которого сервер будет ожидать запрос на обслуживание или завершение запроса. Если пользователю потребуется для отправки запроса или приема отклика сервера больше времени, чем указано в `TimeOut` — сервер разорвет соединение. Этим способом сервер ограничивает максимальное время передачи данных, что непосредственно влияет на объем передаваемой информации. Поэтому при необходимости предоставлять пользователям файлы большого объема или работе в медленных сетях вы должны увеличить значение этой директивы.

6.30 Директива User

Синтаксис:	User *unix-userid*
Значение по умолчанию:	`User #-1`
Контекст:	сервер, конфигурация
Режим:	ядро

Директива `User` устанавливает идентификатор пользователя, под которым работает сервер, отвечая на запросы. Чтобы использовать эту возможность, сервер, работающий в режиме `standalone` первоначально должен быть запущен под именем root. it Unix-userid может принимать одно из следующих значений:

- **Имя пользователя**

  Ссылается на идентификатор пользователя по его имени.

- **# за которым следует число**

  Ссылается на пользователя по номеру его идентификатора.

Используемый идентификатор пользователя не должен предоставлять привилегии доступа к файлам, которые не должны быть видны за пределами локального компьютера, а кроме того, не должен допускать выполнение кода, который не предназначен для обслуживания запросов httpd. Рекомендуется для этой цели создать специальный идентификатор пользователя и группу. Некоторые администраторы используют общего пользователя `nobody`, но это оказывается не всегда желательным или удобным.

БЕЗОПАСНОСТЬ: Не устанавливайте идентификатор пользователя или группы в `root`, если вы не отдаете себе полного отчета в последствиях этого шага.

6.31 Директива «VirtualHost»

Синтаксис:	«VirtualHost» *addr* «/VirtualHost»
Контекст:	сервер, конфигурация
Режим:	ядро

Эта группа используется для объединения набора директив, которые применяются только по отношению к одному конкретному виртуальному хосту. В этой группе допускается использовать любые директивы, разрешенные в контексте виртуальных хостов. Когда сервер получает запрос на предоставление документа от какого-либо из виртуальных хостов, он использует конфигурационные директивы, размещенные в секции `<VirtualHost>`. Аргумент *addr* может принимать следующие значения:

- IP-адрес виртуального хоста;
- Полное доменное имя виртуального хоста.

Например:

```
<VirtualHost robinzon.segnior.com>
ServerAdmin webmaster@robinzon.segnior.com
DocumentRoot /www/docs/robinzon
ServerName robinzon.segnior.com
ErrorLog logs/robinzon-error_log
TransferLog logs/robinzon-access_log
</VirtualHost>
```

В текущей версии Apache каждому виртуальному хосту должен соответствовать персональный IP-адрес, поэтому при конфигурации ядра системы вы должны включить поддержку многоадресной работы. Если же ваша версия операционной системы не поддерживает этот режим, вы можете воспользоваться командой `ifconfig alias`, (если, конечно, ваша система понимает эту команду), или вставить в ядро заплатку (например, VIF для SunOS 4.1.x).

Глава 7

Стандартные модули Apache

7.1 Модуль mod_access

Исходный текст данного модуля находится в файле `mod_access.c` и по умолчанию компилируется вместе с ядром сервера. Модуль обеспечивает контроль доступа на основе информации об IP-адресе или доменном имени пользователя.

7.1.1 allow

Синтаксис:	allow from *host host* ...
Контекст:	каталоги, .htaccess
Переопределяет:	Limit
Режим:	базовый
Модуль:	mod_access

Директива `allow` определяет, какие хосты могут получать доступ к данному каталогу; обычно помещается в секцию `<Limit>`. Аргумент *host* может принимать одно из следующих значений:

- `all` — разрешает доступ всем пользователям;
- `доменное (возможно частичное) имя` — хосты, имена которых совпадают с данным (хотя бы в части суффикса);

- `полный IP-адрес` — IP-адрес хоста, которому разрешен доступ;

- `частичный IP-адрес` — первые один, два или три байта адреса, формирующие подсеть, которой разрешен доступ.

Например:
`allow from .ncsa.uicuc.edu`
разрешает доступ всем пользователям, принадлежащим соответствующему домену.

Обратите внимание, что совпадение проверяется по всем компонентам — `bar.edu` отличается от `foobar.edu`!

См. также `deny` и `order`.

7.1.2 deny

Синтаксис:	deny from *host host* ...
Контекст:	каталоги, .htaccess
Переопределяет:	Limit
Режим:	базовый
Модуль:	mod_access

Директива `deny` определяет, каким хостам запрещен доступ к заданному каталогу; как правило, эта директива используется в секции `<Limit>`. Аргумент *host* принимает одно из следующих значений:

- `all` — доступ запрещен для всех узлов;

- `доменное (возможно частичное) имя` — доступ запрещается хосту или домену;

- `полный IP-адрес` — IP-адрес узла, к которому запрещен доступ;

- `частичный IP-адрес` — ограничение доступа для подсети (от одного до трех байтов IP-адреса).

Например:

```
deny from 16
```

Всем хостам, принадлежащим указанной сети запрещается доступ.

Следует иметь в виду, что что проверяется полное совпадение адресов и доменов. иначе говоря `bar.edu` не совпадает с `foobar.edu`.

См. также `allow` и `order`.

7.1.3 order

Синтаксис:	order *упорядочение*
Значение по умолчанию:	`order deny, allow`
Контекст:	каталоги, .htaccess
Переопределяет:	Limit
Режим:	базовый
Модуль:	mod_access

Директива `order` управляет порядком обработки директив `allow` и `deny`. *Упорядочение* может принимать одно из значений:

- `deny, allow` — директивы deny обрабатываются до директив allow;
- `allow, deny` — директивы allow обрабатываются до директив deny;
- `mutual-failure` — доступ разрешается только тем хостам, которые присутствуют в списках allow и отсутствуют в списках deny.

Например:

```
order deny,allow
deny from all
allow from .ncsa.uiuc.edu
```

Доступ разрешен только хостам из домена `.ncsa.uiuc.edu`.

7.2 mod_alias

Исходный текст модуля содержится в файле `mod_alias.c`; который по умолчанию компилируется вместе с ядром сервера. Модуль обеспечивает отображение различных частей файловой системы хоста на дерево документов и выполняет перенаправление URL.

7.2.1 Alias

Синтаксис:	Alias *url-путь имя-каталога*
Контекст:	конфигурация сервера, виртуальные хосты
Режим:	базовый
Модуль:	mod_alias

Директива `Alias` позволяет хранить документы в локальной файловой системе за пределами поддерева, определяемого DocumentRoot. URL с декодированным по %–правилам путями, начинающимися с *url-путь* отображаются на локальные файлы, имя которых начинается с *имя-каталога*. Например:

```
Alias /image /ftp/pub/image
```

Запрос по адресу `http://myserver/images/foo.gif` приведет к тому, что сервер вернет пользователю файл `/ftp/pub/images/foo.gif`.

См. также `ScriptAlias`.

7.2.2 Redirect

Синтаксис:	Redirect *url-путь url*
Контекст:	конфигурация сервера, виртуальные хосты
Режим:	базовый
Модуль:	mod_alias

Директива `Redirect` отображает старый URL на новый. Новый URL возвращается клиенту, который пытается загрузить документ и инициализирует повторную загрузку. *URL-путь* представляет собой адрес (с декодированием по %-правилам) все запросы к документам, начинающимся с этого пути доступа будут переадресовываться на новый адрес, начинающийся с *url*. Например:

```
Redirect /srv http://foo2.bar.com/service
```

Если клиент запрашивает `http://mysrv/srv/foo.txt`, ему вместо него будет предложен документ `http://foo2.bar.com/service/foo.txt`.

> **Внимание:** Директивы Redirect имеют приоритет по сравнению с директивами Alias и ScriptAlias, вне зависимости от их порядка следования в конфигурационном файле.

7.2.3 ScriptAlias

Синтаксис:	ScriptAlias *url-путь имя-каталога*
Контекст:	конфигурация сервера, виртуальные хосты
Режим:	базовый
Модуль:	mod_access

Директива `ScriptAlias` ведет себя аналогично директиве Alias, однако в отличие от последней отмечает целевой каталог, как содержащий CGI-скрипты. URL-адреса, начинающиеся с *url-путь* отображаются на скрипты, имя которых начинается с *имя-каталога*. Например:

```
ScriptAlias /cgi/bin/  /web/cgi-bin/
```

Запрос по адресу `http://myserver/images/foo` приведет к запуску на сервере скрипта `/web/cgi-bin/foo`.

7.3 Модуль mod_asis

Исходный код модуля размещен в файле `mod_asis.c` и по умолчанию компилируется вместе с ядром сервера. Модуль осуществляет поддержку файлов `.asis`. Любой документ, mime-тип которого определен как `httpd/send-as-is` обрабатывается этим модулем.

7.3.1 Назначение

Модуль предназначен для обработки документов, которые должны отправляться сервером Apache без добавления заголовков HTTP.

Модуль может использоваться для отправки с сервера произвольных данных, включая перенаправления и другие специальные сообщения HTTP и не требует использования cgi-скриптов или nph-скриптов.

7.3.2 Использование

В конфигурационном файле сервера определите тип mime, называемый `httpd/send-as-is`, например:

```
AddType httpd/send-as-is  asis
```

Эта строка определяет `.asis` расширение имени файла как новый тип mime. Содержимое любого файла с этим расширением будет отправлено пользователю почти без изменений. Клиенту потребуются заголовки HTTP, поэтому не забудьте добавить их самостоятельно. Кроме того, необходим заголовок `Status:`, при этом данные должны быть записаны в трехцифровом коде откликов HTTP, за которым следует текстовое сообщение.

Ниже приведен пример файла, содержимое которого отправляется *as is (как есть)*, и который извещает о перенаправлении запроса.

```
Status: 302 Now where did I leave that URL
Location: http://xyz.zbc.com/foo/bar.html
Content-type:  text/html

<HTML>
<HEAD>
<TITLE>Lame Exuses'R'us</TITLE>
</HEAD>
<BODY>
<H1>Fred's exceptionally wonderful page has moved to
<A HREF="http://xyz.abc.com/foo/bar.html">Joe's</A> site.
</H1>
```

```
</BODY>
</HTML>
```

Внимание: сервер всегда добавляет поле `Date:` и `Server:` — заголовок данных возвращается клиенту, поэтому эти поля не нужно включать в файл. Одна из недоработок, которая будет вскоре устранена, состоит в том, чттто сервер не добавляет заголовок `Last-Modified`.

7.4 Модуль mod_auth

Этот модуль находится в файле `mod_auth.c` и по умолчаниюю компилируется вместе с ядром сервера. Модуль осуществляет аутентификацию пользователя с использованием текстовых файлов.

7.4.1 AuthGroupFile

Синтаксис:	AuthGroupFile *имя-файла*
Контекст:	каталоги, .htaccess
Переопределяет:	AuthConfig
Режим:	базовый
Модуль:	mod_auth

Директива `AuthGroupFile` устанавливает имя текстового файла, содержащего список групп пользователей для их аутентификации. *Имя-файла* представляет собой полное имя файла групп.

Каждая строка файла групп содержит имя группы, за которым следует двоеточие и имена членов группы, разделенные пробелами. Например:

```
mygroup:  bob joe anne
```

Обратите внимание, что поиск в больших группах *исключительно неэффективен*; в этом случае целесообразно использовать AuthDBMGroupFile.

Безопасность: убедитесь, что AuthGroupFile хранится за пределами дерева документов WWW-сервера; ни в коем случае не помещайте его в тот каталог, который этот файл должен защищать. В противном случае ваши клиенты смогут выгрузить AuthGroupFile на свои компьютеры.

См. также AuthName, AuthType и AuthUserFile.

7.4.2 AuthUserFile

Синтаксис:	AuthUserFile *имя_файла*
Контекст:	каталоги, .htaccess
Переопределяет:	AuthConfig
Режим:	базовый
Модуль:	mod_auth

Директива AuthUserFile устанавливает имя текстового файла, содержащего список пользователей и паролей, используемых при аутентификации. *Имя_файла* представляет собой полный путь к файлу паролей.

Каждая строка файла паролей содержит имя пользователя, за которым следует двоеточие и пароль, зашифрованный с помощью системной функции *crypt()*. Поведение системы при многократном использовании одного и того же имени пользователя разработчиками не регламентировано.

Поиск пользователей в больших файлах реализован неэффективно, поэтому в таких случаях рекомендуется использовать AuthDBMUserFile.

Безопасность: убедитесь, что AuthUserFile хранится вне пределов дерева документов WWW-сервера не помещайте его в каталог, файлы которого вы защищаете; в противном случае клиенты получат возможность выгрузить файл паролей на свои компьютеры.

См. также AuthName, AuthType и AuthGroupFile.

7.5 Модуль mod_cgi

Модул размещается в файле `mod_cgi.c` и по умолчанию компилируется вместе с ядром. Модуль обеспечивает поддержку CGI-скриптов и обрабатывает все файлы, mime-тип которых соответствует `application/x-httpd-cgi`.

7.5.1 Общие сведения

Любой файл, mime-тип которого определен как `application/x-httpd-cgi`, рассматривается сервером как CGI-скрипт, который запускается сервером с возвращением результатов (выходного потока данных) клиенту. Файлы получают данный тип либо за счет присвоения им суффикса, указанного директивой AddType, либо путем помещения в каталог ScriptAlias.

При вызове сервером CGI-скрипта, в переменные окружения добавляется новая переменная с именем `DOCUMENT_ROOT`. Переменная содержит значение значение конфигруационной переменной DocumentRoot.

7.5.2 Переменные поддержки CGI

В соответствии со спецификациями CGI сервер устанавливает следующие переменные среды:

REMOTE_HOST

Устанавливается только в том случае, если сервер не был скомпилирован в режиме `MINIMAL_DNS`.

REMOTE_IDENT

Устанавливается только в том случае, если IdentityCheck установлена в `on`.

REMOTE_USER

Устанавливается только в том случае если CGI-скрипт является объектом для аутентифкации.

7.6 Модуль mod_dir

Модуль содержится в файле `mod_dir.c` и по умолчанию компилируется вместе с ядром. Назначение модуля состоит в индексации каталогов документов.

7.6.1 Общие сведения

Модуль управляет индексацией каталогов. Индекс каталога может поступать от одного из двух источников:

- Файл, созданный пользователем, обычно называемый `index.html`. Имя этого файла устанавливается директивой DirectoryIndex.
- В случае отсутствия такого файла индекс автоматически генерируется сервером. Формат получаемого листинга определяется значениями нескольких директив. Директивы AddIcon, AddIconByEncoding и AddIconByType используются для назначения иконок различным типам файлов.

7.6.2 AddDescription

Синтаксис:	AddDescription *строка файл файл...*
Контекст:	конфигурация сервера, виртуальный хост, каталоги, .htaccess
Переопределяет:	Indexes
Режим:	базовый
Модуль:	mod_dir

Директива задает описание файла, используемое в режиме FancyIndexing. *Файл* представляет собой расширение файла, частичное имя, выражение с заполнителями или полное имя файла, которому соответствует данного описание. *Строка* заключается в двойные кавычки:

```
AddDescription "The planet Mars" /web/pics/mars.gif
```

7.6.3 AddIcon

Синтаксис:	AddIcon*иконка имя имя...*
Контекст:	конфигурация сервера, виртуальный хост, каталоги, .htaccess
Переопределяет:	Indexes
Режим:	базовый
Модуль:	mod_dir

Директива устанавливает иконку, которая отображается после имени файла в режиме FancyIndexing. *Иконка* представляет собой либо относительный (%-ориентированный) URL, либо запись формата *(alttext, url)*, где *alttext* — текстовый тэг, используемый вместо иконки в броузерах, неподдерживающих графические режимы[1].

Имя может принимать одно из значений: ^^DIRECTORY^^ для каталогов или же
^^BLANKICON^^ для пустых строк (чтобы корректно выполнить форматирование списка в целом), либо содержать расширение файла, фрагмент имени с заполнителями, частичное имя или полное имя файла. Например:

```
AddIcon (IMG,/iconsimage.xbm} .gif .jpg .xbm
AddIcon /icons/dir.xbm ^^DIRECTORY^^
AddIcon /icons/backup.xbm *~
```

Отметим, что директива AddIconByType, по возможности, должна использоваться до рассмотренной выше AddIcon.

7.6.4 AddIconByEncoding

Синтаксис:	AddIconByEncoding *иконка mime-код...*
Контекст:	конфигурация сервера, виртуальный хост, каталоги, .htaccess
Переопределяет:	Indexes
Режим:	базовый
Модуль:	mod_dir

Директива устанавливает иконку, отображаемую в режиме FancyIndexing после файлов, имеющих *mime-кодирование*. *Иконка* может адресоваться либо через относительный URL-адрес, либо записываться как пара *(alttext, url)*, где *alttext* — текстовый тэг, используемый в броузерах, не имеющих графической поддержки.

Mime-кодирование представляет собой выражение с заполнителями, используемое при кодировании содержания документа. Например:

```
AddIconByEncoding /icons/compress.xbm x-compress
```

[1]например, Lynx

7.6.5 AddIconByType

Синтаксис:	AddIconByType *Иконка mime-тип*...
Контекст:	конфигурация сервера, виртуальные хосты, каталоги, .htaccess
Переопределяет:	Indexes
Режим:	базовый
Модуль:	mod_dir

Директива устанавливает иконку, отображаемую в режиме FancyIndexing после файлов, имеющих тип *mime-тип*. *Иконка* представляет собой либо относительный URL-адрес иконки, либо запись формата *(alttext, url)*, где *alttext* — текстовый тэг, используемый вместо иконки в броузерах, не имеющих графической поддержки.

Mime-тип представляет собой выражение с заполнителями, соответствующее типам mime. Например:

```
AddIconByType (IMG, /icons/image.xbm) image/*
```

7.6.6 DefaultIcon

Синтаксис:	DefaultIcon *url*
Контекст:	конфигурация сервера, виртуальные хосты, каталоги, .htaccess
Переопределяет:	Indexes
Режим:	базовый
Модуль:	mod_dir

Директива DefaultIcon устанавливает вид иконки, которая отображается после файлов, для которых не установлено никакой специфической иконки, отображаемой в режиме FancyIndexing. *URL* представляет собой относительный URL-адрес иконки. Например:

```
DefaultIcon /icon/unknown.xbm
```

7.6.7 DirectoryIndex

Синтаксис:	DirectoryIndex *локальный-url*...
Контекст:	конфигурация сервера, виртуальные хосты, каталоги, .htaccess
Значение по умолчанию:	`index.html`
Переопределяет:	Indexes
Режим:	базовый
Модуль:	mod_dir

Директива DirectoryIndex устанавливает список ресурсов, просматриваемых при запросе клиентом индекса каталога путем запроса NULL файла после имени каталога. *Локальный-URL* представляет собой адрес документа на сервере, относительно запрашиваемого каталога; как правило, это имя файла в каталоге. Допускается указание нескольких адресов — сервер вернет первый документ, который будет найден. Если же не существует ни один из них, сервер самостоятельно сгенерирует собственный листинг каталога. Например:

```
DirectoryIndex index.html
```

приведет к тому, что запрос по адресу `http://myserver/docs/` приведет к пересылке клиенту документа `http://myserver/docs/index.html`, если этот файл существует или же просто листинг каталога, если такого файла нет.

Обратите внимание, что документы вовсе не обязаны каким либо образом соотносится с каталогом:

```
DirectoryIndex index.html index.txt /cgi-bin/index.pl
```

приведет к тому, что в случае, если не будет обнаружен ни `index.html`, ни `index.txt` будет вызван CGI-скрипт `/cgi/bin/index.pl`.

7.6.8 FancyIndexing

Синтаксис:	FancyIndexing *boolean*
Контекст:	конфигурация сервера, виртуальные хосты, каталоги, .htaccess
Переопределяет:	Indexes
Режим:	базовый
Модуль:	mod_dir

FancyIndexing разрешает или запрещает использование режима форматированного вывода содержимого каталогов. *Boolean* может принимать значения `on` и `off`. По возможности вместо этой директивы рекомендуется использовать IndexOptions.

7.6.9 HeaderName

Синтаксис:	HeaderName *имя_файла*
Контекст:	конфигурация сервера, виртуальные хосты, каталоги, .htaccess
Переопределяет:	Indexes
Режим:	базовый
Модуль:	mod_dir

Директива HeaderName устанавливает имя файла, содержимое которого вставляется в начало листинга каталога.
Имя_файла представляет собой имя включаемого файла и записывается относительно индексируемого каталога. Вначале сервер пытается включить файл *Имя_файла*`.html` в виде HTML-документа, а если это не удается — то в виде обычного текстового файла. Например:

```
HeaderName HEADER
```

при индексации каталога `/web` сервер вначале ищет файл `/web/HEADER.html` и включает его, если он найден, в противном случае включается текстовый файл `/web/HEADER`, если конечно, он существует.

См.также ReadmeName.

7.6.10 IndexIgnore

Синтаксис:	IndexIgnore *файл файл...*
Контекст:	конфигурация сервера, виртуальные хосты, каталоги, .htaccess
Переопределяет:	Indexes
Режим:	базовый
Модуль:	mod_dir

Директива IndexIgnore добавляет файлы в список файлов, не включаемых в формируемый листинг. *Файл* может представлять собой расширение файла, его частичное имя, выражение с символами-заполнителями или полное имя игнорируемых файлов. Многократное использование директив IndexIgnore приводит к расширению списка игнорируемых файлов. По умолчанию, список содержит одну запись — '.'. Например:

```
IndexIgnore README .htaccess *~
```

7.6.11 IndexOptions

Синтаксис:	IndexOptions *опция опция ...*
Контекст:	конфигурация сервера, виртуальный хост, каталоги, .htaccess
Переопределяет:	Indexes
Режим:	базовый
Модуль:	mod_dir

Директива IndexOptions устанавливает режим индексирования каталогов. При этом *Опция* может приобретать одно из следующих значений:

FancyIndexing — Включает режим "оформленного" вывода листингов каталогов.

IconAreLinks — в режиме FancyIndexing делает иконки частью области захвата файла при выборе документа.

ScanHTMLTitles — в режиме FancyIndexing разрешает извлечение заголовка HTML-документа. Если файл не имеет описания, заданного с помощью AddDescription, сервер извлечет из документа тэг «TITLE». Следует отметить, что этот режим приводит к резкому возрастанию процессорной и дисковой активности сервера.

SupressLastModified — в режиме FancyIndexing подавляет вывод на печать даты последней модификации файла.

SupressSize — в режиме FancyIndexing подавляет выдачу информации о размерах файлов.

SupressDescription — в режиме FancyIndexing подавляет выдачу описания файлов.

По умолчанию все опции выключены. В случае, если в одном и том же каталоге используется несколько опций, будет выполнена та, которая имеет более узкую сферу действия — объединения опций не происходит. Например:

```
<Directory /web/docs>
IndexOptions FancyIndexing
</Directory>
<Directory /web/docs/spec>
IndexOptions ScanHTMLTitles
</Directory>
```

приведет к тому, что режим `ScanHTMLTitles` будет введен только для каталога
`/web/docs/spec`.

7.6.12 ReadmeName

Синтаксис:	ReadmeName *имя_файла*
Контекст:	конфигурация сервера, виртуальный хост, каталоги, .htaccess
Переопределяет:	Indexes
Режим:	базовый
Модуль:	mod_dir

Директива ReadmeName устанавливает имя файла, содержимое которого дописывается в конец листинга каталога. *Имя_файла* представляет собой имя включаемого в состав документа файла и рассматривается относительно индексируемого каталога. Вначале сервер пытается включить *имя_файла*`.html` как HTML-документ, а если это не удается, включает *имя_файла* как обычный текс. Например:

```
ReadmeName README
```

при индексировании каталога `/web` приведет к тому, что сервер вначале попытается загрузить HTML-файл `/web/README.html` и включить его в выходной документ, а если это не удастся, то включить текстовый файл `/web/README`, если конечно он существует.

См. также HeaderName

7.7 Модуль mod_imap

Этот модуль содержится в файле `mod_imap.c` и по умолчанию компилируется вместе с ядром сервера. Модуль обеспечивает поддержку `.map` файлов, замещая таким образом, функциональные возможности CGI-программ. Этот модуль обрабатывает любые документы, с mime-типом `application/x-httpd-imap`.

7.7.1 Общие сведения

Для того чтобы использовать сервероориентированные карты изображений, вы должны прежде всего включить этот модуль в состав сервера и добавить в конфигурационный файл приведенную ниже строку.

```
AddType application/x-httpd-imap map
```

Эта запись означает, что файлы карт изображений будут иметь расширение `.map`.

7.7.2 Новые возможности

Модуль обработки карт изображений имеет ряд особенностей и возможностей, которые не были доступны при использовании других версий программ обработки карт.

- Ссылки URL относительны, за базу берется информация от Referer:.
- По умолчанию назначение <Base> выполняется через новое поле карты — `base_url`.
- Файл `imagemap.conf` более не используется.
- Реализованы точечные ссылки.

Опции, поддерживаемые `base_url`:

- `map` — включена по умолчанию; обеспечивает стандартное (старое) поведение системы интерпретации карт.
- `referer` — использует данные от Referer: — заголовочную информацию, позволяющую обращаться к URL относительно текущего документа.
- `http://любой_url` — обеспечивает установку «Base» таким образом, что URL осуществляет все ссылки относительно этого базового адреса.

Пример построения mapfile

```
default http:/lin/
base_url referer
rect  ..    0,0 77,27
poly http://www.inet.com/ 78,0  194,27
circle http://www.inet.com/lin/fdb/  196,0 305,27
rect  search_index 306,0 419,27
point http://www.zyzzyva.com/  420,0  549,27
```

Обращение к карте изображений

```
<A HREF="http:/maps/imagemap1.map">
<IMG ISMAP SRC="http:/images/imagemap1.gif">
</A>
```

7.8 Модуль mod_include

Этот модуль находится в файле `mod_include.c` и по умолчанию компилируется вместе с ядром. Модуль поддерживает html-документы, разбираемые сервером, так называемые SPML-документы[2]. Любой документ, которому присвоен mime-тип `text/x-server-parsed-html` или `text/x-server-parsed-html3` будет разбираться с использованием этого модуля, а результат обработки получит тип `text/html`.

7.8.1 Формат включаемых файлов SPML

SPML-документ представляет собой обычный HTML-файл со специальными включениями — команндами SGML. Команды имеют следующий синтаксис:

```
<!--#элемент атрибут=значение... -->
```

Поле "значение" часто заключается в двойные кавычки; большая часть команд допускает только одну пару атрибут-значение.

Разрешается использовать следующие элементы:

– **config**

  Эта команда управляет различными аспектами работы анализатора, разбирающего строку. Допустимыми атрибутами являются следующие:

[2]от Server Parsed Markup Language

* **errmsg** — значением этого атрибута является сообщение, которое возвращается клиенту в том случае, если возникает ошибка во время разбора документа.
* **sizefmt** — Устанавливает формат, используемый при отображении размера файла. Допустимыми значениями являются `bytes` для отображения количества байтов и `abbrev`, который выводит количество килобайт или мегабайт, занимаемых файлом.
* **timefmt** — значением атрибута является строка, используемая системной процедурой `strftime(3)` при печати даты и времени.

echo

Эта команда выводит на печать одну из переменных, как это определено ниже. Если значение переменной не задано, она печатается в виде `(none)`. Если в переменной содержится какая-либо дата, она печатается через процедуру `timefmt`[3]. Атрибуты:

– **var** — Значением является имя переменной, которая выводится на печать.

item **exec**

Команда exec выполняет заданную команду оболочки или CGI-сценарий. Опция `IncludesNOEXEC` полностью запрещает действие этой команды. Допустимыми атрибутами являются:

– **cgi** — Значением является относительный URL-адрес CGI-скрипта. Если путь доступа не начинается с символа "/", за точку отсчета берется каталог текущего документа. Документ, на который осуществляется ссылка по этому адресу, вызывается как CGI-скрипт даже в том случае, если обычно он не рассматривается и не распознается как таковой. Однако при этом каталог, содержащий скрипт, должен быть назначен для хранения скриптов[4].

[3]В соответствии с ее текущей конфигурацией
[4]с помощью директивы ScriptAlias или опции ExecCGI

CGI-скрипт получает PATH_INFO и строку запроса QUERY_STRING из первоначального запроса пользователя; эта информация не может быть задана в пути URL. Кроме того, наряду со стандартными переменными окружения CGI, скрипту доступны определенные пользователем переменные.

Если скрипт вместо потока данных возвращает заголовок `Location:`, он будет преобразован в гипертекстовую ссылку HTML.

Рекомендуется при проектировании узла использовать вместо конструкции `exec cgi` элемент `include virtual`.

– **cmd** — Сервер запустит на выполнение указанную команду с помощью оболочки `/bin/sh`. Эта команда может использовать включаемые пользователем переменные.

- **fsize**

Команда выводит на печатьь размер заданного файла в соответствии со спецификацией `sizefmt`. Атрибуты команды:

– **file** — значением является относительный путь доступа к каталогу, содержащему разбираемый в данный момент документ.

– **virtual** — значением является URL-адрес с корнем в текущем каталоге разбираемого документа, если первым символом пути не является "/".

- **flastmod**

Команда печатает дату последней модификации указанного файла с использованием спецификации `timefmt`. Атрибуты команды те же, что и у `fsize`.

- **include**

Команда вставляет в разбираемый файл содержимое другого текстового документа. Любой из вставляемых файлов подчиняется стандартным правилам контроля прав доступа. Если для каталога, содержащего разбираемый файл установлена опция `IncludesNOEXEC`, и включаемый документ приведет к вызову исполняемой программы, его включение будет заблокирвано — это позволяет предотвратить несанкционированное выполнение CGI-скриптов. В остальных случаях CGI-скрипты вызываются как обычно с использованием полного URL-адреса, задаваемого в команде, включая произвольные строки запроса.

Атрибут определяет местоположение документа; включение осуществляется для любого атрибута, заданного с командой `include`. Разрешенными являются следующие атрибуты:

- **file** — значением является относительный путь доступа к файлу[5]. Путь не может содержать фаргменты типа `../`, равно как и быть абсолютным путем доступа. В большинстве случаев рекомендуется вместо этого атрибута использовать рассматриваемый ниже атрибут `virtual`.
- **virtual** — значением является URL-адрес, записанный относительно разбираемого в данный момент документа. URL не может содержать схему или имя хоста, а только путь доступа и необязательную строку запроса. Если путь не начинается с символа "/", он рассматривается как относительный.

URL конструируется из значения атрибутов, и получаемый с его использованием документ включается в отклик сервера, передаваемый пользователю. Допускается вложение файлов при разборе.

7.8.2 Включаемые переменные

Переменные доступны для использования в команде `echo`, а также в любой программе, вызванной документом.

- **DATE_GMT** — содержит текущую дату по Гринвичу.

[5] корнем является текущий каталог разбираемого документа

- **DATE_LOCAL** — текущая дата для местного временного пояса.
- **DOCUMENT_NAME** — Имя файла (за исключением каталогов), запрошенного пользователем.
- **DOCUMENT_URL** — URL-адрес документа, запрошенного пользователем. Обратите внимание, что в случае вложения нескольких файлов этот адрес *не является* URL-адресом текущего документа.
- **LAST_MODIFIED** — Дата последней модификации документа, который был запрошен пользователем.

7.8.3 XBitHack

Синтаксис:	XBitHack *статус*
Значение по умолчанию:	`XBitHack off`
Контекст:	конфигурация сервера, виртуальные хосты, каталоги, .htaccess
Переопределяет:	Options
Режим:	базовый
Модуль:	mod_include

Директива XBitHack управляет порядком разбора обычных HTML-документов. *Статус* может принимать одно из следующих значений:

- **off** — никакой специальной обработки исполняемых файлов.
- **on** — любой файл, для которого установлен режим выполнения в поле прав доступа user, рассматривается как разбираемый сервером html-документ.
- **full** — аналогичен режиму `on`, но кроме того тестирует значение группового поля доступа. Если он установлен, то возвращается дата последней модификации файла. На практике установка этого бита позволяет клиентам и прокси-серверам кэшировать результаты запроса.

7.9 Модуль mod_log_common

Этот модуль размещается в файле `mod_log_common.c` и по умолчанию компилируется с ядром. Модуль обеспечивает регистрацию запросов, направленных серверу с использованием стандартного формата CLF[6].

7.9.1 Формат регистрационных файлов

Каждому запросу в регистрационном файле соответствует отдельная строка, которая состоит из нескольких полей, разделенных пробелами:

```
host ident authuser date request status bytes
```

Если значение какого-либо поля не определено, оно заменяется знаком "-"[7]. Содержание и формат полей следующий:

– **host**

  Полностью определенное доменное имя клиента или его IP-адрес[8].

– **ident**

  Если включен режим `IdentityCheck` и на компьютере клиента запущен сервер `identd`, поле содержит идентификационную информацию, сообщаемую клиентом.

– **authuser**

  В случае, если пользователь пытается получить доступ к защищенному паролем документу, в это поле помещается идентификатор пользователя, использованный при запросе.

[6] CLF — Common Logfile Format

[7] Что весьма облегчает последующую обработку регистрационного файла, например, с помощью awk-программ.

[8] В тех случаях, когда доменное имя отсутствует или недоступно

– **date**

Дата и время запроса в следующем формате:

```
date  = [день/месяц/год:часы:мин:сек пояс]
день  = 2*цифра
месяц = 3*буква
год   = 4*цифра
часы  = 2*цифра
мин   = 2*цифра
сек   = 2*цифра
пояс    = ('+' | '-') 4*цифра
```

– **request**

Запрос, сделанный пользователем, заключенный в двойные кавычки (").

– **status**

Трехзначный цифровой код состояния запроса, возвращаемый клиенту.

– **bytes**

Общее количество данных в об[екте, переданному пользователю без учета служебных заголовков.

7.9.2 TransferLog

Синтаксис:	TransferLog *file_pipe*
Значение по умолчанию	`logs/transfer_log`
Контекст:	конфигурация сервера, виртуальные хосты
Режим:	базовый
Модуль:	mod_log_common

Директива TransferLog устанавливает имя файла, в котором сервер регистрирует входящие запросы. *File_pipe* принимает одно из следующих значений:

– **имя файла**

Имя файла, привязанное к ServerRoot.

– **— за которым следует команда**

Программа, которая получает информацию от регистрационного агента по каналу стандартного ввода. Обратите внимание, что для режима VirtualHost нельзя запустить отдельную программу, если она наследует TransferLog от основного сервера.

Безопасность: если используется внешняя программа обработки журнала, она запускается с идентификаторомтого пользователя, кто запустил httpd. Если запуск осуществлялся администратором, то и эта программа будет иметь идентификатор root. Поэтому необходимо убедитьсяя в надежности этой программы.

7.10 Модуль mod_mime

Модуль содержится в файле `mod_mime.c` и по умолчанию компилируется с ядром сервера. Модуль обеспечивает определение типов файлов по их имени.

7.10.1 Общие сведения

Данный модуль используется для определения типов документов. Некоторые mime-типы требуют от сервера специальной обработки, а другие просто возвращаются клиенту, броузер которого самостоятельно выполняет все необходимые операции.

Имя файла рассматривается этим модулем как точечная запись следующего формата:

base.type.language.enc

При этом *base* соответсвует базовому (основному) имени файла, позволяющему отличить его от прочих; расширение *type* устанавливает тип документа[9]; расширение *language* определяет язык документа в соответствии с директивой `AddLanguage`.

И наконец, расширение *enc* устанавливает способ кодирования документа в соответствии с директивой `AddEncoding`.

7.10.2 AddEncoding

Синтаксис:	AddEncoding *mime расширение...*
Контекст:	конфигурация сервера, виртуальные хосты, каталоги, .htaccess
Переопределяет:	FileInfo
Режим:	базовый
Модуль:	mod_mime

Директива AddEncoding добавляет к списку расширений имен файлов новые типы кодирования. *Mime* представляет собой тип mime-кодирования, используемый документами, в конце имени которых стоит *расширение*. Например:

```
AddEncoding x-gzip gz
AddEncoding x-compress Z
```

Это приведет к тому, что файлы, заканчивающиеся на .gz будут отмечены, как закодированные с помощью x-gzip, а файлы .Z — как закодированные методом x-compress.

7.10.3 AddLanguage

Синтаксис:	AddLanguage *mime-lang расширение...*
Контекст:	конфигурация сервера, виртуальные хосты, каталоги, .htaccess
Режим:	базовый
Модуль:	mod_mime

[9]Типы документа определяются в файле `TypesConfig` и директивой `AddType`.

Директива AddLanguage добавляет в список расширений имен файлов информацию о языках, на которых выполнены документы. *Mime-lang* — это mime-код языка документов, имена которых заканчиваются на *расширение*, после того, как из имени файла будет удалено расширение *enc*. Например:

```
AddEncoding x-compress Z
AddLanguage en .en
AddLanguage fr .fr
```

В этом случае документ `xxxx.en.Z` будет интерпретироваться сервером как упакованный англоязычный документ. К сожалению, хотя информация о языке содержания сообщается клиенту, большинство броузеров просто игнорируют эти сведения. Поэтому использование директивы AddLanguage на практике оказывается более удобным при переговорах относительно содержания документа, при которых сервер возвращает один из нескольких документтов в соответствии с предпочтениями пользователя.

7.10.4 AddType

Синтаксис:	AddType *mime-type расширение...*
Контекст:	конфигурация сервера, виртуальные хосты, каталоги, .htaccess
Режим:	базовый
Модуль:	mod_mime

Директива AddType добавляет в список расширений имен файлов сведения о типах содержания документов. *Mime-type* представляет собой mime-тип документов, которые заканчиваются расширением *расширение*, после того, как из имени файла удалены расширения, соответствующие методу кодирования и языку. Например:

```
AddType image/gif GIF
```

> Рекомендуется добавлять новые типы документов с использованием директивы AddType, а не модифицировать файл TypesConfig.

Обратите внимание, что в отличие от сервера NCSA httpd, эта директива не может использоваться для назначения типов *отдельных* файлов.

7.10.5 TypesConfig

Синтаксис:	TypesConfig *имя_файла*
Значение по умолчанию	`conf/mime.types`
Контекст:	конфигурация сервера
Режим:	базовый
Модуль:	mod_mime

Эта директива определяет имя файла, в котором хранится информация о mime-типах. *Имя_файла* привязано к ServerRoot. В этом файле хранится список используемых по умолчанию связей между расширением файла и типом содержания хранящихся в них документтов, поэтому изменять этот файл не рекомендуется. В случае необходимости можно внести необходимые изменениея с помощью директивы AddType. Файл состоит из текстовых строк, записанных в формате аргументов директивы AddType:

mime-type расширение расширение...

Расширения записываются строчными буквами. Пустые строки и строки, начинающиеся с символа "`#`" игнорируются.

7.11 Модуль mod_negotiation

Модуль размещается в файле `mod_negotiation.c` и по умолчанию компилируется в составе ядра сервера. Назначением модуля является осуществлвение переговоров с программой клиента относительно содержания документов. Любой документ, mime-тип которого принадлежит к типам семейства `application/x-type-map` будет обрабатываться этим модулем.

7.11.1 Общие сведения

Переговоры о содержании, или, что более корректно, выбор содержания документов, предствляет на практике выбор одного из нескольких доступных документов, который в макисмальной степени удовлетворяет возможностям программных и аппаратных средств клиента. В сервер Apache это реализуется двумя способами:

– Карта типов (файл mime-типа `application/x-type-map`, который в явном виде перечисляет файлы, содержащие различные варианты документа.

– Поиск MultiView, включаемый опцией MultiViews, при котором сервер выполняет самостоятельный поиск файлов по шаблонам и выбирает наиболее подходящий файл из полученных результатов.

Карты типов

Карта типов имеет тот же формат, что и заголовки писем электронной почты[10]. Карта содержит описания документов, разделенных пустыми строками, при этом строки, начинающиеся с символа # интерпретируются как комментарии. Описание документов состоит из нескольких записей, каждая из которых может занимать несколько строк – строка считается продолжением предыдущей, если начинается с одного или нескольких пробелов. При обработке первый пробел строки удаляется и полученная строка объединяется с предыдущей.

[10] RFC 822

Запись состоит из заголовка, представляющего собой ключевое слово, которое завершается двоеточием (:) после которого следует значение этой записи. Допускается произвольное использование пробелов между заголовком записи и ее значением, а также между отдельными словами в заначении. Список разрешенных значений заголовков приведен ниже:

– **Content-Encoding:**

  Метод кодирования файла. В настоящее время сервером распознается только два типа кодирования *x-compress* для файлов, упакованнных утилитой compress и *x-gzip* для файлов, упакованных с помощью gzip.

– **Content-Language:**

  Язык, на котором составлен данный вариант документа в соответствии со стандартами Internet, например *en* для английского языка.

– **Content-Length:**

  Размер файда в байтах. Если это поле описания отсутствует, сервер использует реальную длину файла.

– **Content-Type:**

  MIME-тип документа, с дополнительными параметрами, которые отделяются от типа документа и друг отдруга символом точки с запятой (;). Синтаксис записи стандартный: *имя=значение*. При этом сервером обрабатываются следующие параметры:
  level — значением является целое число, которое устанавливает версию типа данных. Для документов типа `text\html` этот параметр по умолчанию равен 2, для остальных типов — 0.
  qs — значением является число с плавающей запятой в диапазоне от 0 до 1. Его значением является "качество" данного варианта документа.

Например:

`Content-Type:   image/jpeg; qs=0.8`

– **URL:**

Путь к документу, содержащему описанный вариант файла относительно к файлу карты типов.

MultiViews

Поиск в режиме MultiViews разрешен при установке опции MultiViews. Если сервер получает запрос на `/some/dir/foo` и при этом `/some/dir/foo` *не* существует, то сервер просматривает каталог на предмет всех файлов `foo.*`[11], и назначает всем записям, соответствующим указаному шаблону поиска те же типы кодирования содержимого и типов документов, которые используются при прямом доступе к этим документам по имени. После этого сервер выбирает документ, в максимальной степени соответствующий входному запросу и возвращает его клиенту.

7.11.2 LanguagePriority

Синтаксис:	LanguagePriority *mime-lang* ...
Контекст:	конфигурация сервера, виртуальные хосты, каталоги .htaccess
Переопределяет:	FileInfo
Режим:	базовый
Модуль:	mod_mime

Директива LanguagePriority устанавливает последовательность просмотра языковых вариантов документов в том случае, когда клиент не установил самостоятельно приоритеты при использовании запросов MultiViews. Список *mime-lang* составляется в порядке уменьшения предпочтения. Например:

```
LanguagePriority   en fr de
```

[11] обратите внимание на наличие точки

При запросе документа `foo.html`, в случае, если в каталоге находятся и
`foo.html.fr` и `foo.html.de`, но броузер не заявил о своих предпочтениях, пользователю будет возвращен документ `foo.html.fr`.

7.12 Модуль mod_userdir

Этот модуль находится в файле `mod_userdir.c` и по умолчанию компилируется вместе с ядром. Модуль поддерживает управление пользовательскими каталогами.

7.12.1 UserDir

Синтаксис:	UserDir *каталог*
Значение по умолчанию	`public_html`
Контекст:	конфигурация сервера, виртуальные хосты
Режим:	базовый
Модуль:	mod_userdir

Директива UserDir устанавливает реальный каталог в домашнем каталоге пользователя, который используется при получении запроса на документ для данного пользователя. *Каталог* может либо иметь значение `disabled` — в этом случае данная функция отключается, либо представлять собой имя каталога.

В последнем случае запрос документ по URL, начинающемуся с
`http://myserver/` *unix-username*
будет транслирован в имя файла, начинающегося с
/home-dir/directory, где */home-dir* является домашним каталогом для пользователя *unix-username*. Например:

```
UserDir public_html
```

В этом случае запрос по адресу:

```
http://myserver/~foo56/adir/file.html
```

вернет документ:

```
http://myserver/home/foo56/public_html/adir/file.html
```

Глава 8

Модули расширения

В этой главе мы рассмотрим модули сервера Apache, которые включены в состав дистрибутива сервера, но не входят в состав базового ядра. Иначе говоря, для использования этих функций вам потребуется самостоятельно перекомпилировать сервер.

8.1 Модуль mod_auth_dbm

Этот модуль помещается в файле `mod_auth_dbm.c` и по умолчанию *не включается* в состав ядра. Модуль используется для аутентификации пользователей с помощью DBM-файлов[1].

8.1.1 AuthDBMGroupFile

Синтаксис:	AuthDBMGroupFile *имя_файла*
Контекст:	каталоги, .htaccess
Переопределяет:	AuthConfig
Режим:	расширение
Модуль:	mod_auth_dbm

Директива AuthDBMGroupFile устанавливает имя DBM-файла, содержащего список групп пользователей, используемый при аутентификации пользователей. *Имя_файла* является абсолютным путем

[1] См. раздел, посвященный работе с DBM

к файлу групп. Файл групп использует в качестве ключа имя пользователя.

Для каждого из пользователей формируется список, содержащий перечень групп (разделенных запятыми, к которым принадлежит данный пользователь). При этом в строке описывающей группы не допускается использование пробелов и двоеточий.

Безопасность: убедитесь, что AuthDBMGroupFile размещается за пределами дерева документов, используемого сервером — не помещайте его в каталоги, которые им защищаются. В противном случае пользователи получат возможность выгрузить этот файл на свои компьютеры.

См. также AuthName, AuthType и AuthDBMUserFile.

8.1.2 AuthDBMUserFile

Синтаксис:	AuthDBMUserFile *имя_файла*
Контекст:	каталоги, .htaccess
Переопределяет:	AuthConfig
Режим:	расширение
Модуль:	mod_auth_dbm

Директива AuthDBMUserFile устанавливает имя DBM-файла, в котором содержится список пользователей и паролей для аутентификации пользователей. *Имя_файла* представляет собой полное имя к файлу со списком пользователей.

Файл пользователей использует в качестве ключа имя пользователя. Значением записей о пользователях является регистрационный пароль, зашифрованный с помощью функции `crypt()` и дополнительные сведения, отделенные от пароля двоеточием. Фактически, все символы строки после двоеточия сервером игнорируются.

Безопасность: убедитесь, что AuthDBMUserFile находится за пределами дерева документов Web-сервера; ни в коем случае не помещайте его в те каталоги, которые он должен защищать. В противном случае пользователи получат возможность выгрузки AuthDBMUserFile на свои компьютеры.

См. также AuthName, AuthType и AuthDBMGroupFile.

8.2 Модуль mod_cookies

Модуль находится в файле `mod_cookies.c` и по умолчанию в состав ядра не включается. Назначение модуля состоит в поддержке идентификационных сообщений (cookies), разработанных фирмой Netscape и используемых в ее броузерах.

8.2.1 CookieLog

Синтаксис:	CookieLog *имя_файла*
Контекст:	конфигурация сервера, виртуальные хосты
Режим:	экспериментальный
Модуль:	mod_cookies

Эта директива устанавливает имя файла, в котором производится регистрация всех идентификационных сообщений. Имя файла привязано к ServerRoot.

8.3 Модуль mod_dld

Модуль содержится в файле `mod_dld.c` и по умолчанию в состав ядра не включается. Модуль осуществляет загрузку исполняемого кода и модулей в сервер на этапе загрузки, используя при этом библиотеку dld, входящую в состав разработок GNU.

8.3.1 Общие сведения

Модуль dld представляет собой фрагмент кода, осуществляющий загрузку различных модулей в соответствии с конфигурацией сервера[2], с использованием библиотки динамической компоновки GNU DLD. По умолчанию этот модуль не используется, поскольку сама библиотека DLD используется не на всех машинах.

[2]Эта операция осуществляется однократно, только при загрузке программы; повторное считывание конфигурационных файлов не приводит к изменению состояния загруженных модулей.

Необходимо отметить, что по ряду причин наряду с прикладными модулями в состав конфигурационных файлов должна быть включена строка:

```
LoadFile  /lib/libc.a
```

Примечание: поскольку DLD при загрузке должна считывать таблицу символов непосредственно из двоичного кода сервера, все эти команды окончатся сообщением об ошибке, если сервер не сумеет найти собственного образа на диске.

8.3.2 LoadFile

Синтаксис:	Loadfile *имя_файла...*
Контекст:	конфигурация сервера
Режим:	экспериментальный
Модуль:	mod_dld

Директива LoadFile подключает в загружаемый сервер другие исполняемые файлы или библиотеки; эта функция используется для включения в состав Apache дополнительных библиотек, которые могут потребоваться для поддержки работы различных модулей. Аргумент директивы *имя_файла* привязан к переменной ServerRoot.

8.3.3 LoadModule

Синтаксис:	LoadModule *модуль имя_файла*
Контекст:	конфигурация сервера
Режим:	экспериментальный
Модуль:	mod_dld

Директива LoadModule осуществляет связь сервера с объектным или библиотечным файлом *имя_файла* и добавляет структуру *модуль* к списку активных модулей. *Модуль* представляет собой имя внешней переменной типа `module`, определенной в файле. Например:

```
LoadModule ai_backcompat_mod modules/mod_ai_back.o
LoadFile   /lib/libc.a
```

загружает модули из подкаталога `modules`, привязанного к корневому каталогу сервера, определяемому переменной ServerRoot.

8.4 Модуль mod_log_agent

Этот модуль помещен в файл `mod_log_agent.c` и по умолчанию вместе с ядром не компилируется. Этот модуль осуществляет регистрацию работы пользовательских программ-агентов.

8.4.1 AgentLog

Синтаксис:	AgentLog *файл-конвейер*
Значение по умолчанию	`AgentLog logs/agent_log`
Контекст:	конфигурация сервера, виртуальные хосты
Режим:	расширение
Модуль:	mod_log_agent

Директива AgentLog устанавливает имя файла, в котором сервер будет регистрировать заголовки запросов типа UserAgent. *Файл-конвейер* может принимать одно из следующих значений:

- **Имя файла** — имя файла, привязанное к ServerRoot.
- **Символ "—" за которым следует команда** — программа, принимающая регистрационую информацию от агента по каналу стандартного ввода. При этом, не допускается инициализация новой программы регистрации в режиме виртуального хоста, если она наследует регистратор AgentLog, запущенный основным сервером.

Безопасность

Если используется программа обработки регистрационных записей, она будет записана с идентификатором пользователя, запустившего httpd. Поэтому, если сервер запускал администратор, то ID программы также будет root; поэтому убедитесь, что программа обработки не компрометирует вашу систему защиты.

Данная директива предназначена для обеспечения совместимости с сервером NCSA 1.4

8.5 Модуль mod_log_config

Данный модуль помещен в файл `mod_log_config.c` и по умолчанию с ядром не компилируется. Модуль обеспечивает регистрацию запросов к серверу с использованием форматов, определяемых администратором узла.

8.5.1 Общие сведения

Это *экспериментальный* модуль, который поддерживает как стандартную директиву описания конфигурации TransferLog[3], так и дополнительную директиву LogFormat.

Аргументом директивы LogFormat является строкка, которая может включать печатные символы, копируемые в журнальные файлы и следующие '%'-директивы:

%..h:	Удаленный хост
%..l:	Регистрационное имя пользователя (identd)
%..u:	Удаленный пользователь[4] (от auth)
%..t:	Время в общем формате журнальных файлов
%..r:	Первая строка запроса пользователя
%..s:	Состояние. Для запросов, которые перенаправляются в пределах сервера, это состояние **первоначального** запроса — %..>s для получения последнего.
%..b:	Количество переданных байтов.
%..Foobari:	Содержание Foobar: строки заголовков в запросе, отправленные клиенту
%..Foobaro:	Содержание Foobar: строки заголовков в ответе.

Приведенная в таблице последовательность '...' может либо вообще отсутствовать (например, `"%h %u %r %s %b"`), или же могут содержать условия включения объекта в журнальную запись (если

[3]определенную в модуле log, который включается в состав сервера по умолчанию

условие не удовлетворяется, вместо поля будет помещен прочерк — '-'). Обратите внимание, что над строками от %r, %...i и %...o не выполняется обработка ESC-последовательностей.

Условие включения поля в запись журнала представляет собой список кодов состояний HTTP, которым может предшестовать символ '!'. Таким образом, запись: '%400,501User-agenti:' осуществляет запись в журнале только при возникновении ошибок 400 и 501[5], а запись '%!200,304,302Refereri:' регистрирует Referer-заголовок для всех запросов, которые **не возвращают** код нормального завершения.

По умолчанию LogFormat воспроизводит CLF (см. ниже).

При работе с виртуальными хостами данный модуль ведет себя следующим образом: виртуальный хост может иметь свой собственный LogFormat или свой собственный
TransferLog. Если собственный LogFormat не определен, он наследуется у основного сервера. Если же у виртуального хоста не определен собственный TransferLog, он осуществляет запись в тот же дескриптор (то есть процесс '—...').

Это означает, что допустимо использование трюков типа:

```
<VirtualHost hosta.com>
LogFormat "hosta ..."
...
</VirtualHost>

<VirtualHost hostb.com>
LogFormat "hostb ..."
...
</VirtualHost>
```

В этом примере различные виртуальные хосты осуществляют запись в один и тот же файл, поэтому возникает необходимость в установке идентификатора, показывающего, к какому хосту относится регистрационная запись[6].

[5]Неверный запрос, Не реализовано
[6]вообще то в этом случае целесообразно использовать поле %v

8.5.2 LogFormat

Синтаксис:	LogFormat *строка*
По умолчанию	`LogFomat "%h %l %u %t %r %s %b"`
Контекст:	конфигурация сервера, виртуальные хосты
Режим:	Эксспериментальный
Модуль:	mod_log_config

Эта директива устанавливает формат регистрационного файла.

8.5.3 TransferLog

Синтаксис:	TransferLog *файл-конвейер*
По умолчанию	`TransferLog logs/transfer_log`
Контекст:	конфигурация сервера, виртуальные хосты
Режим:	Экспериментальный
Модуль:	mod_log_config

Директива TransferLog устанавливает имя файла, в котором сервер регистрирует входящие запросы. *Файл-конвейер* принимает одно из значений:

- **Имя файла** — имя файла, привязанное к ServerRoot.
- **символ '—' за которым следует команда** — программа, принимающая информацию от агента журнала по каналу стандартного ввода. Обратите внимание, что новая программа в режиме VirtualHost не может быть запущена если наследует TransferLog от основного сервера.

Безопасность: если используется внешняя программа, она будет запущена под идентификатором пользователя, стартовавшего httpd. Если сервер запускается администратором, необходимо убедиться, что программа обработки журналов не компрометирует вашу систему защиты.

8.6 Модуль mod_log_referer

Этот модуль содержится в файле `mod_log_referer.c` и по умолчанию в состав ядра не включается. Модуль обеспечивает журналирование обращений к документам, которые ссылаются на документы, находящиеся на сервере.

8.6.1 Формат регистрационного файла

Регистрационный файл для каждой ссылки использует отдельную строку вида:

```
uri -> document
```

где *uri* — URL документа, который запрошен клиентом, а *url* — локальный адрес документа, на который производится ссылка.

8.6.2 ReferIgnore

Синтаксис:	ReferIgnore *строка строка...*
Контекст:	конфигурация сервера, виртуальные хосты
Режим:	Расширение
Модуль:	mod_log_referer

Директива ReferIgnore добавляет в список игнорируемых заголовков Referer новые записи. Если какая-либо из строк списка содержится в заголовке Referer, тогда при запросе соответствующая запись в регистрационном журнале помещаться не будет. Например:

```
RefererIgnore www.ncsa.uiuc.edu
```

Эта строка приведет к игнорированию упоминания в журнале ссылок, связанных с узлом www.ncsa.uiuc.edu.

8.6.3 RefererLog

Синтаксис:	RefererLog *файл-конвейер*
По умолчанию	`RefererLog logs/referer_log`
Контекст:	конфигурация сервера, виртуальные хосты
Режим:	расширение
Модуль:	mod_log_referer

Директива RefererLog устанавливает имя файла, в котором будет осуществляться запись заголовков Referer для входящих запросов. *Файл-конвейер* может принимать одно из следующих значений:

- **Имя файла** — имя файла, привязанное к ServerRoot.
- **символ '—' за которым следует команда** — программа, принимающая информацию от агента журнала по каналу стандартного ввода. Обратите внимание, что новая программа в режиме VirtualHost не может быть запущена если наследует RefererLog от основного сервера.

Безопасность: если используется внешняя программа, она будет запущена под идентификатором пользователя, стартовавшего httpd. Если сервер запускается администратором, необходимо убедиться, что программа обработки журналов не компрометирует вашу систему защиты.

Эта директива введена для обеспечения совместимости с NCSA 1.4.

8.7 Директивы Russian Apache

В этом параграфе подробно рассмотрены директивы управления работой сервера, реализованные в версии *Russian Apache*. Все приведенные ниже команды вполне стандартным образом включаются в конфигурационные файлы.

8.7.1 AgentCharset

Синтаксис:	AgentCharset *имя_таблицы Шаблоны...*
Контекст:	конфигурация сервера, виртуальные хосты
Режим:	расширение
Модуль:	Russian Apache

Директива **AgentCharset** определяет таблицу, которая может быть использована при нахождении в запросе клиента подстроки, являющейся идентификатором клиента. Подстрока ищется в поле *User-Agent* запроса.

Параметры директивы:

- **Имя_таблицы** — официальное имя таблицы кодировок.

- **Шаблон** — один или несколько шаблонов для поиска в поле User-Agent клиентского запроса.

Следует отметить, что шаблоны не могут являться регулярными выражениями, допускается только использование подстрок[7].

Примеры использования директивы AgentCharset:

```
AgentCharset windows-1251 AIR_Mosaic IWENG/1 MSIE
AgentCharset windows-1251 WinMosaic (Win
AgentCharset windows-1251 (Win16; (Win95; (16-bit)
AgentCharset koi8-r Arena Ariadna Macintosh
AgentCharset koi8-r OmniWeb Sextant PRD (X11
AgentCharset ibm866 DosLynx
```

8.7.2 BadAgent

Синтаксис:	BadAgent *Шаблоны...*
Контекст:	конфигурация сервера, виртуальные хосты
Режим:	расширение
Модуль:	Russian Apache

Некоторые клиентские программы (броузеры) не могут адекватно среагировать на MIME-заголовки, что в ряде случаев приводит к конфузам. Примером может служить простейший заголовок вида:

```
Content-type: text/html; charset=koi8-r; level=3
```

Обычно сервер в таких случаях автоматически выдает документ с указанной кодировкой, однако может потребоваться изменить этот порядок. Для решения этой задачи и предназначена директива

[7]Последовательности символов

BadAgent. Параметрами директивы являются подстроки[8], которые сервер при разборе запроса пытается найти в поле User-Agent. Если одна из подстрок директивы будет найдена, сервер воздержится от выдачи документа в кодировке, указанной в charset.

Пример использования директивы:

```
BadAgent lynx/2.1 arena
```

По вполне очевидным причинам целесообразно указывать не только название клиентской программы, но и номер ее версии. На сегодняшний день полный список программ, которые должны быть учтены с помощью директивы BadAgent выглядит следующим образом:

- arena
- Lynx/2.0
- Lynx/2.1
- Lynx/2.2
- Lynx/2.3
- Lynx/2.4
- "MSIE 2.0;"[9]

8.7.3 CharsetAgentPriority

Синтаксис:	CharsetAgentPriority *On—Off*
Контекст:	конфигурация сервера, виртуальные хосты
Значение по умолчанию:	Off
Режим:	расширение
Модуль:	Russian Apache

[8]Именно подстроки, регулярные выражения не поддерживаются

[9]Обратите внимание на кавычки. Они необходимы для того, чтобы включить в состав подстроки, анализируемой сервером символ пробела.

Директива **CharsetAgentPriority** определяет приоритетность при поиске необходимой кодировки, если из анализа URL и User-Agent возникает конфликт. При установке параметра директивы в *On* сервер, в первую очередь, будет смотреть на поле User-Agent. И если в этом поле будет найден известный серверу шаблон, то будет использована соответствующая этому шаблону кодировка, заданная с помощью AgentCharset. Если же шаблон найден не будет — сервер попытается извлечь кодировку из префикса имени виртуального сервера, к которому обращен запрос или из префикса затребованного URI. Если директива имеет значение *Off*, приоритеты поиска меняются местами.

См. также AgentCharset

8.7.4 CharsetAlias

Синтаксис:	CharsetAlias *имя_таблицы Синоним...*
Контекст:	конфигурация сервера, виртуальные хосты
Режим:	расширение
Модуль:	Russian Apache

Директива используется для описания имен синонимов (псевдонимов) указанной таблицы кодировки. Все синонимы определяются для каждого виртуального сервера независимо друг от друга.

Параметры директивы:

- **Имя_таблицы** — официальное имя таблицы кодировки. Должно быть определено до данной директивы с помощью CharsetTable.

- **Синоним** — один или несколько синонимов для данного официального имени таблицы.

Примеры использования данной директивы:

```
CharsetAlias iso_8859-5:1988 cyrillic iso iso-8859-5
CharsetAlias ibm866 csibm866 866 cp866 x-cp866 cp-866 alt
CharsetAlias windows-1251 win x-win cp1251 cp-1251
```

См. также CharsetTable.

8.7.5 CharsetPriority

Синтаксис: CharsetPriority *Табл1 Табл2 Табл3* ...
Контекст: конфигурация сервера, виртуальные хосты
Режим: расширение
Модуль: Russian Apache

Директива **CharsetPriority** предназначена для расстановки приоритетов таблиц кодировок, установленных для сервера. Установленные приоритеты используются в том случае, если в запросе клиента для различных таблиц указаны равные весовые коэффициенты или же пользователем не указан NativeCharset. В этом случае будет использована таблица, имеющая наивысший приоритет.

Параметры директивы — это список таблиц в порядке *убывания* приоритета[10]. Все используемые в этой директиве имена таблиц должны быть определены ранее директивами CharsetTable. Допускается использование только официальных имен таблиц.

Пример:

```
CharsetPriority windows-1251 koi8-r ibm866
```

См. также CharsetTable, NativeCharset

8.7.6 CharsetTable

Синтаксис: CharsetTable *имя_таблицы Табл1 [Табл2] Язык*
Контекст: конфигурация сервера, виртуальные хосты
Режим: расширение
Модуль: Russian Apache

Директива **CharsetTable** служит для описания таблиц символов, используемых в сервере. Каждая директива описывает только один набор. Если в системе работает несколько виртуальных серверов, и в каждом из них необходимо использовать несколько типов кодировок, то они должны быть описаны *для каждого виртуального сервера* в разделе <VirtualHost>.

Основные параметры:

[10]Самый левый имеет наивысший приоритет.

- **Имя_таблицы** — официальное название таблицы символов, например, *windows-1251, koi8-r, ibm866, iso_8859-5:1988* и так далее).

- **Табл1** — Имя файла таблицы, устанавливающей соответствие между внутренним представлением документов и документов, в данной кодировке, используемой при передаче их клиенту.

- **Табл2** — Имя файла таблицы, устанавливающей соответствие между внутренним представлением документов и документов, в данной кодировке при приеме их от клиента. В случае отсутствия этой таблицы в числе параметров сервер автоматически осуществляет преобразование, обратное **Табл1**.

- **Язык** — Двухбуквенный код языка, к которому принадлежит описываемая в данной директиве таблица. Этот код должен быть зарегистрирован с помощью директив **AddLanguage** и **LanguagePriority**.

Примеры использования директивы CharsetTable:

```
CharsetTable iso_8859-5:1988 conf/koi-iso.tab ru
CharsetTable ibm866 conf/koi-alt.tab ru
CharsetTable windows-1251 conf/koi-win.tab ru
CharsetTable koi8-r conf/koi-koi.tab ru
```

Несмотря на кажущуюся избыточность, достаточно интересным решением представляется возможность использования специальной таблицы для перекодирования документов, поступающих от клиента. В самом деле, из того, что вы отправили документ пользователю в кодировке Windows вовсе не следует, что и загружаемые им документы будут записаны в этом коде. Например, Netscape Navigator for Windows 3.0 принимает документы в любой указанной вами кодировке, а отправляет только в КОИ-8, осуществляя их преобразование из CP-1251 самостоятельно.

См. также AddLanguage, LanguagePriority

8.7.7 NativeCharset

Синтаксис:	NativeCharset *имя_таблицы*
Контекст:	конфигурация сервера, виртуальные хосты
Режим:	расширение
Модуль:	Russian Apache

Директива определяет имя таблицы перекодирования, которая используется при работе с конкретным сервером. Параметром является официальное имя таблицы. Например:

```
NativeCharset koi8-r
NativeCharset windows-1251
```

Директива используется для каждого виртуального сервера отдельно. При этом имя таблицы должно быть определено ранее с помощью директивы CharsetTable.

8.7.8 NoHostnameCharset

Синтаксис:	NoHostnameCharset *On—Off*
Контекст:	конфигурация сервера, виртуальные хосты
Значение по умолчанию:	Off
Режим:	расширение
Модуль:	Russian Apache

По умолчанию, если клиентом не запрошена конкретный тип кодировки, сервер пытается ее определить по своему имени, например, *win.www.company.com*. Использование данной директивы позволяет запретить серверу выполнять эту операцию. Пример:

```
NoHostnameCharset On}
```

8.7.9 NoSoBad

Синтаксис:	NoSoBad *Шаблоны...*
Контекст:	конфигурация сервера, виртуальные хосты
Режим:	расширение
Модуль:	Russian Apache

Директива определяет исключения из "черного списка", установленного с помощью директивы BadAgent. Введена в связи с тем, что новые версии Lynx требуют-таки указания *charset=*, в то время как старые, как это уже упоминалось ранее, просто безобразничают. Поскольку в ходу сегодня старых версий броузеров довольно много, проще описать новые как исключения. Все шаблоны, используемые в качестве параметра являются подстроками[11].

Пример использования директивы:

```
BadAgent Lynx
NoSoBad Lynx/2.7 Lynx/2.6 Lynx/2.5FM
```

В результате все версии броузера Lynx за исключением версий 2.5FM, 2.6 и 2.7 будут считаться "плохими".

См.также BadAgent

8.7.10 NoUriCharset

Синтаксис:	NoUriCharset *On* — *Off*
Контекст:	конфигурация сервера, виртуальные хосты
Значение по умолчанию:	Off
Режим:	расширение
Модуль:	Russian Apache

Директива позволяет серверу запретить определение требуемой клиенту кодировки по URI. Например, сервер по умолчанию полагает, что документы типа:

```
http://www.company.com/win/name.html
```

должны выдаваться в кодировке windows-1251. Использование директивы:

```
NoUriCharset On
```

позволит заблокировать этот режим и приведет к выдаче запрошенной страницы в кодировке, указанной директивой NativeCharset.

См. также NativeCharset

[11]Использование регулярных выражений не допускается.

8.7.11 RejectErrorCharset

Синтаксис:	RejectErrorCharset *On—Off*
Контекст:	конфигурация сервера, виртуальные хосты
Значение по умолчанию:	Off
Режим:	расширение
Модуль:	Russian Apache

Директива предназначена для определения действий сервера при получении в запросе клиента требования выдачи документов в кодировке (charset=...), неизвестной серверу. При установке параметра этой директивы в *On* сервер не будет выдавать документ в кодировке, указаной директивой NativeCharset, а вернет клиенту сообщение об ошибке в запросе. Если же значением директивы является Off — сервер выдаст документ как сможет.

Пример использования:

```
RejectErrorCharset  On
```

См. также NativeCharset

8.8 Модуль mod_mc_htmls1.2

Это модуль, который позволяет хранить на диске документы HTML в разных кодировках (koi8, cp1251 etc) и "прозрачно" перекодировать их в нужные моменты времени во "внутреннее представление" сервера. При этом, вне зависимости от "кодировки хранения", все документы для пользователя будут выглядеть совершенно одинаково.

Модуль рассчитан на использование совместно с любой версией Apache, начиная с 1.1, хотя пока тестировалась только работа с 1.1.3.

Рекомендуется использование совместно с русифицированной версией сервера *Russian Apache*, что позволит не только хранить документы на диске в разных кодировках, но и показывать их клиенту в соответствии с его предпочтениями. При этом mod_mc_html оформлен независимо от *Russian Apache* и (теоретически) должен работать и с любой другой *корректной* русификацией этого сервера (то есть такой, при которой "русифицируются" не только показ html/shtml, но и результаты работы любого модуля).

Модуль не входит в состав дистрибутивного комплекта сервера Apache и может быть получен по адресу
ftp://ftp.lexa.ru/pub/domestic/lexa в файле
mod_mc_htmls1.2.tar.gz.

Модуль предназначен для некоммерческого и/или нетиражируемого использования, что позволяет устанавливать его на коммерческих серверах WWW без заключения каких-либо дополнительных соглашений с автором.

8.8.1 Установка модуля

Mod_mc_htmls представлет собой обычный модуль Apache. Чтобы его собрать, нужно скопировать mod_mc_htmls.c в директорию, где лежат исходные тексты Apache, добавить строчку

```
Module mc_htmls_module     mod_mc_htmls.o
```

в файл Configuration и запустить

```
./Configure; make
```

после чего получится файл httpd, который и является www-сервером.

> **Внимание!** Для использования этого модуля, Apache должен быть собран с *mime_module*.

8.8.2 FileCharSet

Синтаксис:	FileCharset *имя_набора таблица*
Контекст:	сервер, виртуальный хост
Режим:	расширение
Модуль:	mod_mc_htmls

Эта директива устанавливает используемое сервером правило перекодировки файла во внутреннюю кодировку сервера.

Параметр может принимать одно из следующих значений:

- имя_набора (произвольный набор букв и цифр без пробелов), используется для дальнейших ссылок на это правило перекодировки

- Таблица — имя файла с таблицей перекодировки относительно $ServerRoot. Таблица описывает перекодировку из "файла на диске" во внутреннее представление сервера[12]. Формат таблиц аналогичен используемому в Russian Apache

Если введено несколько директив **FileCharset** с одинаковым именем, то будет использоваться описание, введенное первым.

Пример:

```
FileCharset cp1251 conf/win-koi.tab
```

описывает charset с именем cp1251 и перекодировкой во "внутреннее" представление по таблице $SERVER_ROOT/conf/win-koi.tab.

8.8.3 UseFileCharset

Синтаксис:	UseFileCharset *имя_набора расширение1...*
Контекст:	конфигурация сервера, виртуальный хост, каталоги, .htaccess
Режим:	расширение
Модуль:	mod_mc_htmls

Директива осуществляет связывание правила перекодировки с расширением файла.

Параметры принимают следующие значения:

- *имя_набора* — имя правила, описанного директивой **FileCharset**

- *расширениеN* — одно или несколько расширений имен файлов, для которых нужно применять это правило. Расширение может включать ведущую точку.

[12]Внутреннее представление - это кодировка русского языка, принятая в вашей системе, обычно это KOI8-R

В случае, когда одно расширение имени файла используется в нескольких директивах с разными UseFileCharset на одном уровне видимости, то сработает более позднее описание. Порядок приоритета для описания на разных уровнях видомости обычный для Apache — наибольший приоритет у .htaccess (если разрешено AllowOverride), наименьший — у server-wide конфигурации.

Пример:

```
UseFileCharset koi8 .html .htm
UseFileCharset cp1251 .htmw .h1251
<Directory /some/where>
UseFileCharset cp1251 .html
</Directory>
```

Приведенные выше директивы предписывают использовать для файлов .html и .htm таблицу, описанную в *FileCharset koi8*, а для .htmlw и .h1251 — описанную в *FileCharset cp1251*. В каталоге /some/where для файлов .html будет использована таблица cp1251.

8.8.4 DefaultFileCharset

Синтаксис:	DefaultFileCharset *имя_набора*
Контекст:	конфигурация сервера, виртуальные хосты, каталоги, .htaccess
Режим:	расширение
Модуль:	mod_mc_htmls

Директива указывает серверу, какой из наборов перекодировок использовать для перекодировки файлов в случае, когда расширение файла не описано в **UseFileCharset**. Параметр представляет собой название кодировки из директивы *FileCharset*. В случае, когда *DefaultFileCharset* не определена и режим обработки данного типа файлов не задан с помощью *UseFileCharset*, поведение сервера определяется следующим:

- Если включен режим **RecodeAllFiles on** (см ниже), то при неустановленом парметре **DefaultFileCharset** управление передается дальше по цепочке обработчиков,

- Если файл "подпал" под обработку в результате действия директивы **AddHandler**, то файл просто не перекодируется.

Зарезервированное имя **none** отменяет кодировку DefaultFileCharset, унаследованную от одного из родительскогих каталогов или глобальной конфигурации сервера. В остальных случаях применются стандартные правила определения прав наследования свойств, принятые в Apache.

Пример:

```
DefaultFileCharset koi8
<Directory /some/where>
AllowOverride FileInfo
DefaultFileCharset cp1251
</Directory>
<Directory /another/place>
DefaultFileCharset none
</Directory>
```

Директивы, приведенные выше, требуют от сервера использовать DefaultCharset с именем koi8 для всего сервера, кроме каталогов */some/where*, где необходимо использовать cp1251 и */another/place*, для которого[13] отменить DefaultCharset совсем.

8.8.5 RecodeAllFiles

Синтаксис:	RecodeAllFiles *on — off*
Значение по умолчанию:	off
Контекст:	конфигурация сервера, виртуальные хосты, каталоги, .htaccess
Режим:	расширение
Модуль:	mod_mc_htmls

Включение этой директивы указывает серверу, что перекодировку нужно производить для всех файлов, параметр Content-Type:

[13]И его подкаталогов, если для них не приведены специальные объявления.

которых имеет значение text/html. При этом, для расширений файлов, описанных в директиве **UseFileCharset** перекодировка производится в соответствии с этой директивой, а для всех прочих — в соответствии с директивой **DefaultFileCharset**. Если **DefaultFileCharset** не установлена, то перекодировка не производится, а управление передается дальше по цепочке обработчиков для типа text/html.

ВНИМАНИЕ! У сервера может быть установлено несколько обработчиков типа text/html. Например, директива XBitHack включает обработку файлов с установленным битом признака исполняемой программы как server-parsed (SHTML) документов. Порядок вызова обработчиков зависит от порядка их описания в файле Configuration сервера. Вследствие этого, при наличии более одного обработчика для text/html можно получить конфигурацию, которая ведет себя достаточно странно. Например, mod_mc_htmls в включенной опцией RecodeAllFiles может "отбирать" документы[14] от XBitHack, а может получиться и наоборот.

Если опция **RecodeAllFiles** выключена, то перекодировка производится только для документов, для которых установлен режим обработки *server-recoded*. Этот обработчик включен в состав модуля и активизируется с помощью директивы AddHandler.

8.8.6 StripHttpEquivs

Синтаксис:	StripHttpEquivs *on* — *off*
Значение по умолчанию:	off
Контекст:	конфигурация сервера, виртуальные хосты, каталоги, .htaccess
Режим:	расширение
Модуль:	mod_mc_htmls

Директива разрешает или запрещает удаление строчек «meta http-equiv ...» из заголовка документа. Эта директива введена для борьбы с очевидным противоречием — ряд HTML-редакторов устанавливает с помощью HTTP-equiv параметр Content-Type до-

[14]Или часть документов

кументов, в результате чего в HTTP-заголовках и в самом документе Content-Type может оказаться несогласованным, что приводит к неприятным последствиям. Во-первых, ряд броузеров ошибочно предпочитают HTTP-Equiv HTTP-заголовкам, что неправильно. Во-вторых, если документ извлекается из локального кэша броузера, то http-заголовков он уже не содержит и броузер берет информацию о Content-Type из «META». Простейшим способом убрать это противоречие оказалось удаление всех «META HTTP-EQUIV из документа. Лучше, конечно, удалять его прямо в файле, но не все авторы HTML-документов на это способны. Для неспособных предназначена директива *StripHttpEquivs*. Эта директива работает только в тех случаях, когда документ обрабатывается данным модулем[15].

Для того, чтобы описанные команды работали в .htaccess, для каталога должен быть установлен параметр *AllowOverride FileInfo*.

8.8.7 Имеющиеся ограничения

- В настоящее время поддерживаются только plain html. Это означает, что ни SSI, ни CGI работать в кодировках, отличных от ”внутренней кодировки сервера” работать не будут. Фактически, модуль подменяет только обработчик, используемый по умолчанию, то есть. тот фрагмент кода сервера, который используется в тех случаях, когда никакие другие методы обработки для данного файла не определены.

- Поддерживается только версия для Unix. Код, обеспечивающий работу под OS/2 (EMX) не реализован.

- При указании в теле документа ссылок (**href**) с использованием русских букв *необходимо* писать их именно как буквы. Конструкции вида
«!– href=”script.cgi?value=перекодироваться не будут.

- Аналогичным образом не перекодируются конструкции вида &code; (Ê и им подобные)

[15] То есть если включены режимы *RecodeAllFiles и DefaultFileCharset* или расширение документа описано в директиве UseFileCharset.

8.9 Директивы Russian Apache

В этом параграфе подробно рассмотрены директивы управления работой сервера, реализованные в версии *Russian Apache*. Все приведенные ниже команды вполне стандартным образом включаются в конфигурационные файлы.

8.9.1 AgentCharset

Синтаксис:	AgentCharset *имя_таблицы Шаблоны...*
Контекст:	конфигурация сервера, виртуальные хосты
Режим:	расширение
Модуль:	Russian Apache

Директива **AgentCharset** определяет таблицу, которая может быть использована при нахождении в запросе клиента подстроки, являющейся идентификатором клиента. Подстрока ищется в поле *User-Agent* запроса.

Параметры директивы:

- **Имя_таблицы** — официальное имя таблицы кодировок.

- **Шаблон** — один или несколько шаблонов для поиска в поле User-Agent клиентского запроса.

Следует отметить, что шаблоны не могут являться регулярными выражениями, допускается только использование подстрок[16].

Примеры использования директивы AgentCharset:

```
AgentCharset windows-1251 AIR_Mosaic IWENG/1 MSIE
AgentCharset windows-1251 WinMosaic (Win
AgentCharset windows-1251 (Win16; (Win95; (16-bit)
AgentCharset koi8-r Arena Ariadna Macintosh
AgentCharset koi8-r OmniWeb Sextant PRD (X11
AgentCharset ibm866 DosLynx
```

[16] Последовательности символов

8.9.2 BadAgent

Синтаксис: BadAgent *Шаблоны...*
Контекст: конфигурация сервера, виртуальные хосты
Режим: расширение
Модуль: Russian Apache

Некоторые клиентские программы (броузеры) не могут адекватно среагировать на MIME-заголовки, что в ряде случаев приводит к конфузам. Примером может служить простейший заголовок вида:

```
Content-type: text/html; charset=koi8-r; level=3
```

Обычно сервер в таких случаях автоматически выдает документ с указанной кодировкой, однако может потребоваться изменить этот порядок. Для решения этой задачи и предназначена директива
BadAgent. Параметрами директивы являются подстроки[17], которые сервер при разборе запроса пытается найти в поле User-Agent. Если одна из подстрок директивы будет найдена, сервер воздержится от выдачи документа в кодировке, указанной в charset.

Пример использования директивы:

```
BadAgent lynx/2.1 arena
```

По вполне очевидным причинам целесообразно указывать
не только название клиентской программы, но и номер ее версии. На сегодняшний день полный список программ, которые должны быть учтены с помощью директивы BadAgent выглядит следующим образом:

- arena
- Lynx/2.0
- Lynx/2.1
- Lynx/2.2
- Lynx/2.3

[17]Именно подстроки, регулярные выражения не поддерживаются

- Lynx/2.4
- "MSIE 2.0;"[18]

8.9.3 CharsetAgentPriority

Синтаксис:	CharsetAgentPriority *On—Off*
Контекст:	конфигурация сервера, виртуальные хосты
Значение по умолчанию:	Off
Режим:	расширение
Модуль:	Russian Apache

Директива **CharsetAgentPriority** определяет приоритетность при поиске необходимой кодировки, если из анализа URL и User-Agent возникает конфликт. При установке параметра директивы в *On* сервер, в первую очередь, будет смотреть на поле User-Agent. И если в этом поле будет найден известный серверу шаблон, то будет использована соответствующая этому шаблону кодировка, заданная с помощью AgentCharset. Если же шаблон найден не будет — сервер попытается извлечь кодировку из префикса имени виртуального сервера, к которому обращен запрос или из префикса затребованного URI. Если директива имеет значение *Off*, приоритеты поиска меняются местами.

См. также AgentCharset

8.9.4 CharsetAlias

Синтаксис:	CharsetAlias *имя_таблицы Синоним...*
Контекст:	конфигурация сервера, виртуальные хосты
Режим:	расширение
Модуль:	Russian Apache

Директива используется для описания имен синонимов (псевдонимов) указанной таблицы кодировки. Все синонимы определяются для каждого виртуального сервера независимо друг от друга.

[18] Обратите внимание на кавычки. Они необходимы для того, чтобы включить в состав подстроки, анализируемой сервером символ пробела.

Параметры директивы:

- **Имя_таблицы** — официальное имя таблицы кодировки. Должно быть определено до данной директивы с помощью CharsetTable.
- **Синоним** — один или несколько синонимов для данного официального имени таблицы.

Примеры использования данной директивы:

```
CharsetAlias iso_8859-5:1988 cyrillic iso iso-8859-5
CharsetAlias ibm866 csibm866 866 cp866 x-cp866 cp-866 alt
CharsetAlias windows-1251 win x-win cp1251 cp-1251
```

См. также CharsetTable.

8.9.5 CharsetPriority

Синтаксис:	CharsetPriority *Табл1 Табл2 Табл3* ...
Контекст:	конфигурация сервера, виртуальные хосты
Режим:	расширение
Модуль:	Russian Apache

Директива **CharsetPriority** предназначена для расстановки приоритетов таблиц кодировок, установленных для сервера. Установленные приоритеты используются в том случае, если в запросе клиента для различных таблиц указаны равные весовые коэффициенты или же пользователем не указан NativeCharset. В этом случае будет использована таблица, имеющая наивысший приоритет.

Параметры директивы — это список таблиц в порядке *убывания* приоритета[19]. Все используемые в этой директиве имена таблиц должны быть определены ранее директивами CharsetTable. Допускается использование только официальных имен таблиц.

Пример:

```
CharsetPriority windows-1251 koi8-r ibm866
```

См. также CharsetTable, NativeCharset

[19]Самый левый имеет наивысший приоритет.

8.9.6 CharsetTable

Синтаксис:	CharsetTable *имя_таблицы Табл1 [Табл2] Язык*
Контекст:	конфигурация сервера, виртуальные хосты
Режим:	расширение
Модуль:	Russian Apache

Директива **CharsetTable** служит для описания таблиц символов, используемых в сервере. Каждая директива описывает только один набор. Если в системе работает несколько виртуальных серверов, и в каждом из них необходимо использовать несколько типов кодировок, то они должны быть описаны *для каждого виртуального сервера* в разделе <VirtualHost>.

Основные параметры:

- **Имя_таблицы** — официальное название таблицы символов, например, *windows-1251, koi8-r, ibm866, iso_8859-5:1988* и так далее).

- **Табл1** — Имя файла таблицы, устанавливающей соответствие между внутренним представлением документов и документов, в данной кодировке, используемой при передаче их клиенту.

- **Табл2** — Имя файла таблицы, устанавливающей соответствие между внутренним представлением документов и документов, в данной кодировке при приеме их от клиента. В случае отсутствия этой таблицы в числе параметров сервер автоматически осуществляет преобразование, обратное **Табл1**.

- **Язык** — Двухбуквенный код языка, к которому принадлежит описываемая в данной директиве таблица. Этот код должен быть зарегистрирован с помощью директив **AddLanguage** и **LanguagePriority**.

Примеры использования директивы CharsetTable:

```
CharsetTable iso_8859-5:1988 conf/koi-iso.tab ru
CharsetTable ibm866 conf/koi-alt.tab ru
CharsetTable windows-1251 conf/koi-win.tab ru
```

```
CharsetTable koi8-r conf/koi-koi.tab ru
```

Несмотря на кажущуюся избыточность, достаточно интересным решением представляется возможность использования специальной таблицы для перекодирования документов, поступающих от клиента. В самом деле, из того, что вы отправили документ пользователю в кодировке Windows вовсе не следует, что и загружаемые им документы будут записаны в этом коде. Например, Netscape Navigator for Windows 3.0 принимает документы в любой указанной вами кодировке, а отправляет только в КОИ-8, осуществляя их преобразование из CP-1251 самостоятельно.

См. также AddLanguage, LanguagePriority

8.9.7 NativeCharset

Синтаксис:	NativeCharset *имя_таблицы*
Контекст:	конфигурация сервера, виртуальные хосты
Режим:	расширение
Модуль:	Russian Apache

Директива определяет имя таблицы перекодирования, которая используется при работе с конкретным сервером. Параметром является официальное имя таблицы. Например:

```
NativeCharset koi8-r
NativeCharset windows-1251
```

Директива используется для каждого виртуального сервера отдельно. При этом имя таблицы должно быть определено ранее с помощью директивы CharsetTable.

8.9.8 NoHostnameCharset

Синтаксис:	NoHostnameCharset *On—Off*
Контекст:	конфигурация сервера, виртуальные хосты
Значение по умолчанию:	Off
Режим:	расширение
Модуль:	Russian Apache

По умолчанию, если клиентом не запрошена конкретный тип кодировки, сервер пытается ее определить по своему имени, например, *win.www.company.com*. Использование данной директивы позволяет запретить серверу выполнять эту операцию. Пример:

```
NoHostnameCharset On}
```

8.9.9 NoSoBad

Синтаксис:	NoSoBad *Шаблоны...*
Контекст:	конфигурация сервера, виртуальные хосты
Режим:	расширение
Модуль:	Russian Apache

Директива определяет исключения из "черного списка", установленного с помощью директивы BadAgent. Введена в связи с тем, что новые версии Lynx требуют-таки указания *charset=*, в то время как старые, как это уже упоминалось ранее, просто безобразничают. Поскольку в ходу сегодня старых версий броузеров довольно много, проще описать новые как исключения. Все шаблоны, используемые в качестве параметра являются подстроками[20].

Пример использования директивы:

```
BadAgent Lynx
NoSoBad Lynx/2.7 Lynx/2.6 Lynx/2.5FM
```

В результате все версии броузера Lynx за исключением версий 2.5FM, 2.6 и 2.7 будут считаться "плохими".

См.также BadAgent

8.9.10 NoUriCharset

Синтаксис:	NoUriCharset *On — Off*
Контекст:	конфигурация сервера, виртуальные хосты
Значение по умолчанию:	Off
Режим:	расширение
Модуль:	Russian Apache

[20]Использование регулярных выражений не допускается.

Директива позволяет серверу запретить определение требуемой клиенту кодировки по URI. Например, сервер по умолчанию полагает, что документы типа:

```
http://www.company.com/win/name.html
```

должны выдаваться в кодировке windows-1251. Использование директивы:

```
NoUriCharset On
```

позволит заблокировать этот режим и приведет к выдаче запрошенной страницы в кодировке, указанной директивой NativeCharset.

См. также NativeCharset

8.9.11 RejectErrorCharset

Синтаксис:	RejectErrorCharset *On—Off*
Контекст:	конфигурация сервера, виртуальные хосты
Значение по умолчанию:	Off
Режим:	расширение
Модуль:	Russian Apache

Директива предназначена для определения действий сервера при получении в запросе клиента требования выдачи документов в кодировке (charset=...), неизвестной серверу. При установке параметра этой директивы в *On* сервер не будет выдавать документ в кодировке, указаной директивой NativeCharset, а вернет клиенту сообщение об ошибке в запросе. Если же значением директивы является Off — сервер выдаст документ как сможет.

Пример использования:

```
RejectErrorCharset  On
```

См. также NativeCharset

Глава 9

Рекомендации по программированию модулей расширения Apache

В данной главе собраны рекомендации по использованию интерфейса программиста сервера Apache применительно к разработке дополнительных модулей, расширяющих функциональные возможности сервера и описаны все необходимые структуры данных. Приведенная информация окажется полезной как при создании принципиально новых модулей, так и при адаптации уже существующих к нуждам конкретных пользователей.

В целях обеспечения расширяемости интерфейса программиста все объявления структур данных, приведенные в этой главе, неполные — на самом деле все они имеют большее количество полей, которые в настоящей главе не рассматриваются. В большинстве случаев эти поля резервируются для использования тем или иным компонентом сервера, а поэтому использовать их в своих целях нужно очень осторожно. В то же время, в некоторых случаях эти поля содержат полезные функции, которые обойти молчанием просто невозможно. Поэтому остается только предупредить читателя, что программирование собственных модулей, не слишком простое занятие, его скорее можно сравнить с прогулкой по лезвию бритвы.

Ну а теперь, после того, как я честно попытался отговорить вас читать эту главу, давайте посмотрим, о чем в ней пойдет речь.

- Основные концепции.
- Обработчики, Модули и Запросы
- Краткий обзор устройства модуля
- Как работают обработчики
- Краткий обзор `request_rec`
- Откуда возникают структуры request_rec
- Обработка запросов, отказ от их выполнения и возвращаемые коды ошибок
- Особенности построения обработчиков запросов пользователя
- Особенности построения обработчиков аутентификационной информации
- Особенности построения обработчиков журнальной информации
- Выделение ресурсов и пул ресурсов сервера
- Конфигурация, команды и так далее
- Конфигурационные структуры, привязанные к каталогам
- Выполнение команд
- Заметки на полях — конфигурация сервера, виртуальные серверы и так далее

9.1 Общие сведения

Мы начнем наш обзор с основных концепций программного интерфейса, и способа их реализации в программном коде сервера.

9.1.1 Обработчики, модули и запросы

Apache разбивает процесс обработки запросов на несколько шагов примерно таким же образом, как это делает сервер Netscape[1]. Ниже приведен общий список этих этапов:

- URI -> преобразование имени файла.
- Аутентификационная провеорка ID (пользователь в самом деле тот, за кого себя выдает?).
- Аутентификационная проверка прав доступа (разрешен ли пользователю доступ *сюда*?).
- Прочие проверки прав доступа.
- Определение MIME-типа запрошенного объекта.
- 'Fixups' — пока не реализованные режимы, зарезервированы для последующего расширения функциональных возможностей сервера подобных `SetEnv`, которые могут оказаться пригодны только для каких-то специфических приложений.
- Возвращение запрошенного документа клиенту.
- Протоколирование выполненного запроса в журнале.

Эти фазы обслуживаются путем просмотра каждого модуля на предмет наличия соответствующего обработчика текущей фазы и вызове соответствующей процедуры, если такой обработчик обнаружен. Как правило, обработчик выполняет одну из трех операций:

- *Обрабатывает* запрос и сообщает о выполнении возвращая код возврата — константу `OK`.

[1] Часть обработчиков этих этапов на сегодняшний день представляет собой просто заглушки, которые предназначены для совместимости с будущим версиями сервера.

- *Отказывается* от обслуживания пользовательского запроса, возвращая целочисленную константу `DECLINED`. В этом случае сервер ведет себя точно так же, как и при отсутствии обработчика.

- Сообщает об ошибке, возвращая один из кодов, определенных протоколом HTTP. В этом случае нормальная обработка запроса прекращается, хотя при необходимости может быть вызван обработчик ErrorDocument, и в любом случае производится запись в журнал.

Большинство фаз завершаются первым же модулем, в котором найден обработчик; однако, для регистрации запросов запускаются все обработчики 'fixups' и проверки прав доступа[2] Сделано это с целью более точного дигностирования и обработки ошибочных ситуаций. Кроме того, фаза формирования отклика сервера является уникальной для тех модулей, которые могут генерировать несколько различных обработчиков для формирования выходных документов через таблицу выбора, основанную на MIME-типе запрошенных документов. Модули могут генерировать обработчик фазы генерации, который способен обслужить *любой* запрос путем передачи ему ключа `*/*` (иначе говоря, укзателя на произвольный MIME-тип документа). Однако, обработчики классов документов, использующих заполнители, вызываются только в тех случаях, когда сервер уже пытался, и не смог найти более конкретный обработчик MIME-типа запрошенного объекта (если не существует ни один из них, отклоняются все).

Сами обработчики представляют собой функции с одним аргументом — структурой `request_rec`, которые возвращают целое число, как это указано выше.

9.1.2 Краткий обзор модуля

В этом разделе мы рассмотрим структуру модуля. Нашим подобытным кроликом будет модуль CGI, который обслуживает обработку

[2] Не связанные с аутентификационной проверкой.

CGI–скриптов и конфигурационную команду ScriptAlias. Конечно, на практике встречается куда больше тонкостей, чем можно судить по одному модулю, но мы хотим просто привести пример создания полезного модуля.

Начнем с обработчиков. Для того, чтобы обслуживать CGI-скрипты модуль объявляет, что имеет генератор выходных документов для запросов этого класса.

А благодаря наличию средств анализа ScriptAlias, модуль имеет также обработчики для фазы трансляции[3] и фазы проверки типов документов[4].

Этот модуль вынужден поддерживать некоторый набор данных для каждого запущенного сервера (в том числе и для каждого вирутального сервера), а именно — информацию о действующих назначениях ScriptAliases; поэтому структура модуля содержит указатели на функции, которые формируют соответствующие структуры данных и на ряд других, которые комбинируют две из них (в том случае, когда директива ScriptAlias определена и для основного сервера, и для виртуального).

И наконец, в этом модуле содержится код, предназначенный для обработки самой команды ScriptAlias. В нашем конкретном случае модуль определяет только одну команду, но их может быть и больше, поэтому модули имеют в своем составе *таблицу команд*, описывающую состав команд модуля, условия, при которых эти команды могут выполняться и способ вызова их обработчиков.

И последнее замечение относительно объявлений типов аргументов для некоторых из этих команд: pool является указателем на структуру, в которой хранится пул ресурсов, используемый сервером для контроля за выделенными областями памяти, открытыми файлами и так далее, либо при обслуживании конкретного запроса, либо для обслуживания самого сервера.

Поэтому при завершении запроса (либо, для конфигурационного пула — при рестарте сервера), память может быть освобождена, а файлы закрыты без написания специального кода, в явном виде выполняющего эти операции. Кроме того, в модуле определена

[3]Чтобы обрабатывать URI, основанные на ScriptAlias

[4]любой запрос, связанный со ScriptAlias рассматривается как скрипт CGI

структура cmd_parms, которая содержит информацию о считываемом конфигурационном файле, а также иную статусную информацию, которая иногда используется функциями, обрабатывающими команды конфигурации сервера[5].

Без дальнейших проволочек приведем текст модуля:

```
/* Объявления обработчиков. */

int translate_scriptalias (request_rec *);
int type_scriptalias (request_rec *);
int cgi_handler (request_rec *);

/* Таблица выбора обрабочиков фазы генерации выходного
 * документа в соотвтетсвии с их MIME- типами
 */

handler_rec cgi_handlers[] = {
{ "application/x-httpd-cgi", cgi_handler },
{ NULL }
};

/* Объявления процедур, предназначенных для
 * манипулирования информацией
 * о конфигурации сервера. Обратите внимание,
 * что они передают и
 * возвращают значения как void* - ядро
 * сервера осуществляет
 * контроль за их выполнением, но не знает,
 * да и не может знать
 * об их внутренней структуре.
 */

void *make_cgi_server_config (pool *);
void *merge_cgi_server_config (pool *, void *, void *);
```

[5]такие как ScriptAlias

```
/*
   Объявление процедур, обслуживающих
   команды конфигурационных файлов
*/

char *script_alias (cmd_parms *,
                    void *per_dir_config,
    char *fake, char *real);

command_rec cgi_cmds[] = {
{ "ScriptAlias", script_alias, NULL, RSRC_CONF, TAKE2,
    "a fakename and a realname"},
{ NULL }
};

module cgi_module = {
   STANDARD_MODULE_STUFF,
   NULL,                   /* инициализатор */
   NULL,                   /* конструктор конфигурации
      каталогов  */
   NULL,                   /* слияние каталогов */
   make_cgi_server_config, /* конфигурация сервера */
   merge_cgi_server_config,/* конфигурация слияния
      серверов */
   cgi_cmds,               /* таблица команд */
   cgi_handlers,           /* обработчики */
   translate_scriptalias,  /* преобразование имен */
   NULL,                  /* проверка USER-ID */
   NULL,                  /* аутентификация */
   NULL,                  /* проверка прав доступа */
   type_scriptalias,      /* проверка типа документа */
   NULL,                  /* fixups */
   NULL                   /* журнал */
};
```

9.2 Обработчики в действии

Единственным аргументом функций–обработчиков является структура `request_rec`. В ней описывается конкретный запрос, который выставляетсся серверу. В большинстве случаев любое подключение к клиенту генерирует только одну структуру `request_rec`.

9.2.1 Краткий обзор request_rec

Структура `request_rec` содержит указатели на пул ресурсов, который очищается после завершения сервером обработки запроса, и содержит временные структуры данных, содержащих информацию о текущем сервере и соединении, а также, что самое важное, о самом запросе.

Эта информация представляет собой небольшой набор символьных строк, описывающих атрибуты запрашиваемого объекта, включая его URI, имя файла, тип содержания и его кодировку[6].

Другими часто используемыми данными являются таблицы, использующие MIME-заголовки пользовательских запросов и формирующие заголовки MIME, колторые отправляются в ответ[7], и переменными окружения для всех порожденных процессов, обслуживающих запросы. Все операции над этими таблицами осуществляются с помощью процедур `table_get` и `table_set`.

И наконец, имеется два указателя на структуры данных, которые, в свою очередь, указывают на конфигурационные структуры конкретного модуля. В частности, они содержат ссылки на структуры, определяющие порядок управления модулем при работе в том или ином каталоге (посредством файлов `.htaccess` и секции `<Directory>`), тех или иных файлов, ориентированных на обслуживание различных модификаций запросов (в этом случае обработчики модуля для одной фазы могут пересылать сообщения обработчикам других фаз запроса). Имеется также еще один конфигурационный вектор в структуре данных `server_rec`, указывающий на конфигурационные данные виртуального сервера.

---

[6]Все эти поля заполняются на этапе трансляции и проверки типов документов соответствующими обработчиками.

[7]Эта возможность используется далеко не всеми модулями.

Ниже приведено сокращенное определение, содержащее наиболее часто используемые поля:

```
struct request_rec {

  pool *pool;
  conn_rec *connection;
  server_rec *server;

  /* Характеристика запрашиваемого объекта */

  char *uri;
  char *filename;
  char *path_info;
  char *args;           /* аргументы запроса,
   если они определены */
  struct stat finfo;    /* устанавливается ядром сервера;
 * если такого файла нет, то
 * st_mode устанавливается в нуль */

  char *content_type;
  char *content_encoding;

  /* Заголовки MIME, входные и выходные.
   * Кроме того, здесь же
   * хранится массив, содержащий переменные
   * среды, передаваемые
   * дочерним процессам, что позволяет создавать модули,
   * использующие переменные среды для передачи
   * информации в процессы.
   *
   * Различие между  headers_out и err_headers_out
   * состоит в том, что
   * последний печатается даже в случае ошибки и
   * сохраняется при
   * внутренних перенаправлениях запроса
   * ( поэтому они входят в состав заголовков, печатаемых
```

```
 * обработчиками ErrorDocument).
 */

table *headers_in;
table *headers_out;
table *err_headers_out;
table *subprocess_env;

/* Информация о самом запросе... */

int header_only;
/* Запрос HEAD, в противоположность GET */
char *protocol;
/* Согласованный с клиентом протокол или HTTP/0.9 */
char *method;        /* GET, HEAD, POST, и т.д. */
int method_number;   /* M_GET, M_POST, и т.д. */

/* Информация для журнала */

char *the_request;
int bytes_sent;

/* Флаг, который могут установить
 * модули для индикации того,
 * что данные возвращаются изменяемыми
 * и клиенты не должны
 * их кэшировать
 */

int no_cache;

/* Прочая информация о конфигурации,
 * которая может быть изменена
 * файлом .htaccess. Здесь помешаются
 * конфигурационные векторы
 * с одним указателем void* для каждого модуля
 * (содержимое ссылки определяется назначением модуля).
```

```
   */

  void *per_dir_config;   /* Опции, устанавливаемые в
     конфигурационных файлах  */
  void *request_config;   /* Дополнительные данные о
     текущем запросе */

};
```

9.2.2 Источник данных для request_rec

Большая часть структур `request_rec` формируется в результате считывания HTTP–запроса от клиента и заполнения соответствующих полей. В то же время имеется и ряд исключений:

- Если выполняется запрос к битовой карте, карте типов[8], или CGI-скриптам, которые возвращают локальный заголовок `Location:`, то ресурсы, запрошенные пользователем должны быть расположены под иным URI, чем тот, который запрашивал пользователь. В этом случае сервер осуществляет *внутреннее перенаправление*, создавая новую структуру `request_rec` для вновь созданного URI, и обрабатывая ее почти так же, как и при непосредственном запросе пользователем нового URI.

- Если какой-либо обработчик сообщил об ошибке и в области видимости находится `ErrorDocument`, в игру вступает тот же механизм внутреннего перенаправления запросов.

- И наконец, в тех случаях, когда обработчику потребовалось выяснить, "что произойдет, если...", могут быть запущены вспомогательные запросы. Например, модуль индексации каталогов должен знать, какой MIME-тип, назначается каждой записи в каталоге, чтобы подобрать соответствующую случаю иконку.

[8]то есть файлам типа `*.var` file

Такие обработчики могут создавать *вторичные запросы* с помощью функций сервера `sub_req_lookup_file` и `sub_req_lookup_uri`, которые порождают новую структуру `request_rec` и обрабатывают ее в соответствии с вашими ожиданиями, не включая лишь реальную отправку выходного документа[9].

Обработка запросов на стороне сервера включает операции по построению вторичных запросов и последующему вызову обработчиков для них посредством функции `run_sub_request`).

9.2.3 Обслуживание запросов, отказ в обслуживание и возвращение кодов ошибок

Как мы уже говорили выше, любой обработчик при вызове для обработки конкретной структуры `request_rec`, должен вернуть переменную типа `int`, объясняющую, чем закончилась работа. При этом возвращается одно из следующих значений:

- OK — запрос обслужен без ошибок. Этот код может привести к завершению фазы, а может потребовать и дальнейшей обработки.

- DECLINED — никаких ошибочных ситуаций не возникло, но модуль отказался обработать текущую фазу; сервер предпринимает попытки найти другой обработчик.

- Код ошибки протокола HTTP, который прерывает обработку запроса.

Обратите внимание, что если возвращается код ошибки `REDIRECT`, то модуль должен поместить в запись `headers_out` поле `Location`, чтобы указать, куда должен быть перемещен клиент.

[9] Кстати, эти функции пропускают проверки прав доступа вторичного запроса, если он выполняется для файла, находящегося в том же каталоге, что и первичный запрос.

9.2.4 Генерация выходных документов

Для большинства фаз обработки запросов функция обработчиков состоит в установке нескольких полей структуры request_rec, или, в случае средств проверки прав доступа, в простом возврате кода ошибки, если условие доступа не выполнено. Но обработчики, отвечающие за генерацию выходных документов, должны на самом деле вернуть клиенту результаты запроса.

Эти процедуры начинают работу с отправки HTTP–заголовка ответа, используя функцию send_http_header[10]. Если запрос имеет маркер header_only, то больше никаких операций выполнять не нужно и формирование выходного документа завершается без каких-либо дополнительных попыток вывода данных.

В противном случае формируется тело документа, которое отправляется клиенту соответствующим образом. Примитивами, использующимися для решения этой задачи, являются функции (для генерируемых в сервере выходных данных) rputc и rprintf, и функция send_fd для копирования файлов, определенных с помощью FILE * непосредственно клиенту.

К настоящему моменту вы в большей или меньшей степени должны понять приведенный ниже фрагмент кода, представляющий собой обработчик запросов GET, для которых не реализованы более детальные обработчики. Кроме того, пример показывает каким образом могут обрабатываться условные запросы GET, в данном случае сопоставляется значение set_last_modified с If-modified-since, переданным клиентом и возвращает соответствующий код[11]. Примерно так же устроен обработчик set_content_length, с тем лишь исключением, что возвращает другой код ошибки.

```
int default_handler (request_rec *r)
{
    int errstatus;
```

[10] Вам не нужно предпринимать никаких специальных усилий, чтобы пропустить отправку заголовков для запросов HTTP/0.9; функция самостоятельно распознает такую ситуацию и корректно ее обрабатывает

[11] Который, в том случае когда отличен от нуля, должен иметь значение USE_LOCAL_COPY

```
    FILE *f;

    if (r->method_number != M_GET)
return DECLINED;
    if (r->finfo.st_mode == 0)
return NOT_FOUND;

    if ((errstatus =
    set_content_length(r, r->finfo.st_size))
        || (errstatus =
    set_last_modified (r, r->finfo.st_mtime)))
        return errstatus;

    f = fopen (r->filename, "r");

    if (f == NULL) {
        log_reason("file permissions deny server access",
      r->filename, r);
        return FORBIDDEN;
    }

    register_timeout ("send", r);
    send_http_header (r);

    if (!r->header_only) send_fd (f, r);
    pfclose (r->pool, f);
    return OK;
}
```

В конце концов, если этот фрагмент требует чрезмерного напряжения усилий, существует и несколько более простых путей. Во-первых, как это показано выше, генератор выходных документов, который еще не сформировал никаких выходных данных, просто возвращает код ошибки, и в этом случае сервер автоматически выдает сообщение об ошибке.

Во-вторых, обработка может быть передана какому–либо другому обработчику путем вызова `internal_redirect`, использующей рассмотренный выше механизм внутреннего перенаправления запросов. Обработчик, который перенаправил запрос внутри сервера на другую процедуру обработки всегда возвращает вызвавшей функции код `OK`.

> **Внимание!** Вызов `internal_redirect` из обработчиков, *не являющихся* генераторами выходных документов может привести к непредсказуемым результатам.

9.2.5 Особенности аутентификации

Процесс аутентификации пользователей и прав доступа к ресурсам отличается в серевере Apache некоторыми особенностями, которые перечислены ниже:

- Обработчики этапа аутентификации не вызываются до тех пор, пока для данного каталога не установлен режим аутентификации.
- Глобальная аутентификационная информация сохраняется в конфигурации ядра (для каждого каталога); она использует структуры `auth_type`, `auth_name`, и `requires`.
- Общие процедуры, используемые для обслуживания протокола, в чатсности, для базовой схемы аутентификации протокола HTTP (`get_basic_auth_pw`, автоматически устанавливающие поля структуры `connection->user` и `note_basic_auth_failure`, значения для заголовка `WWW-Authenticate:` возвращаемого клиенту.

9.2.6 Особенности формирования журнала

При внутреннем перенаправлении запроса возникает проблема — какую именно информацию необходимо поместить в журнал.
Apache решает эту задачу путем помещения всей цепочки перенаправлений в список `request_rec`, которые увязаны в список с помощью указателей `r->prev` и `r->next`.

Структура `request_rec`, которая передается обработчикам журнала в таких случаях, содержит данные, сформированные на основе первоначального запроса клиента; обратите внимание, что поле `bytes_sent` будет содержать корректное значение только для последнего запроса в цепи[12].

9.3 Выделение ресурсов и пулы ресурсов

Одной из проблем при разработке и отладке многопоточного сервера является предотвращение утечки ресурсов, то есть потерь областей оперативной памяти, открытых файлов и так далее, выделенных задаче и должным образом не освобожденных. Для решения этой задачи в Apache используется механизм пула ресурсов, позволяющий выделять ресурсы такми образом, что они *автоматически* освообождаются после того, как их использование завершено.

Работает этот механизм следующим образом: выделяемая память, открываемые файлы и пр., предназначенные для обслуживания конкретного запроса, привязываются к *пулу ресурсов*, который связан с запросом. Пул представляет собой структуру данных, которая отслеживает использование ресурсов.

После того, как обработка запроса завершена, пул *очищается*. В этот момент вся ассоциируемая с ним память освобождается для повторного использования, все файлы закрываются, и выполняются любые другие функции очистки, которые ассоциируются с текущим пулом ресурсов. После того, как эта операция закончена, мы можем быть уверены в том, что ресурсы, связанные с пулом освобождены и ни один из них не утерян.

При перезапуске сервера и назначении памяти и других ресурсов виртуальным серверам, контроль за утечкой осуществляется точно так же. В Apache реализован *конфигурационный пул*, который осуществляет слежение за ресурсами, выделяемыми при считывании конфигурационных файлов сервера и обработке содержащихся в

[12]то есть для того, от которого и направлен в конце концов выходной документ

них команд (например, память, которая была выделена конфигурации модулей каждого из виртуальных серверов, регистрационные журналы, другие открытые файлы и т.д.). При перезапуске сервера, когда осуществляется повторное считывание файлов конфигурации, пул очищается и вся выделенная ранее память и каналы ввода/вывода файлов вновь становятся доступными для использования.

Необходимо отметить, что использование механизма пула ресурсов не является обязательным, за исключением ситуаций, подобных обработчикам журнала, в которых вам действительно необходимо регистрировать процесс очистки памяти, чтобы убедиться в закрытии файлов журнала при перезапуске сервера[13], а также закрытии файловых дескрипторов до прочих дочерних процессов, например CGI-скриптов, или же с помощью механизма таймаута.

Впрочем, существует два серьезных преимущества этой технологии: выделенные с помощью пула ресурсы никогда не теряются (даже если вы выделили пустую строку и забыли о ней!); кроме того выделение памяти с помощью `palloc` обычно работает быстрее, чем `malloc`.

We begin here by describing how memory is allocated to pools, and then discuss how other resources are tracked by the resource pool machinery.

9.3.1 Выделение памяти в пулах

Память в пулах ресурсов выделяется путем обращения к функции `palloc`, которая имеет два аргумента: указатель на структуру пула ресурсов и количество выделяемой памяти (в `char`). При разработке кода для обработчиков запросов наиболее типовым способом получения структуры ресурсов пула является анализ слота `pool` в соответствующей структуре `request_rec` — поэтому не нужно удивляться часто повторяющимся фрагментам кода, образец которых приведен ниже:

```
int my_handler(request_rec *r)
```

[13] Наиболее просто это делается с помощью функции `pfopen`.

```
{
    struct my_structure *foo;
    ...

    foo = (foo *)palloc (r->pool, sizeof(my_structure));
}
```

Обратите внимание, что *функция* `pfree` *не используется* — память, выделенная с помощью `palloc` освобождается только при очистке ассоциированного с ней пула ресурсов. Это означает, что `palloc` не требует столь тщательного контроля как `malloc`; все что от вас требуется, так это округлить размер выделяемой зоны, создать указатель и проверить размер созданной области памяти.

В то же время существует вероятность того, что интенсивное использование `palloc` может привести к существенному увеличению потребления сервером процессорного времени. Существует несколько способов борьбы с этим явлением, которые могут быть сведены к следующему...

Вы можете на скорую руку заменить вызовы `palloc` на `malloc` и попытаться добиться явного освобождения всей выделенной памяти с помощью вызова функций `free`, или же вы можете создать суб-пул ресурсов, выделить необходимые ресурсы в нем и периодически очищать его содержимое. Этот прием мы детально рассмотрим ниже, а на практике он используется в модуле индексации каталогов, где позволяет избежать излишнего выделения памяти при просмотре каталогов, содержащих тысячи файлов.

9.3.2 Выделение инициализированной памяти

Довольно часто огромную пользу приносят функции, позволяющие не только выделить область памяти, но и проинициализировать ее. При работе с пулом ресурсов для этой цели используется функция `pcalloc`, которая имеет тот же синтаксис, что и `palloc`, очищающая содержимое области памяти, перед предоставлением ее в ваше распоряжение.

Функция `pstrdup` использует в качестве аргументов пул ресурсов и `char *`, и осуществляет выделение области памяти, содержащей копию строки, которая передана в качестве аргумента.

И наконец, функция pstrcat представляет собой функцию с переменным количеством аргументов, в число которых входит обязательный указатель на пул ресурсов и по крайней мере два аргумента типа char *, последний из которых должен иметь значение NULL. Эта функция осуществляет выделение памяти, достаточной для размещения копий каждой из строки-аргумента в непрерывной области, например:

```
pstrcat (r->pool, "foo", "/", "bar", NULL);
```

возвращает указатель на 8-байтную область, памяти, проинициализированную значением: "foo/bar".

9.3.3 Контроль открытых файлов

Как уже упоминалось ранее, пул ресурсов используется для слежения не только за оперативной памятью, но и за другими типами ресурсов. Наиболее часто такими ресурсами являются открытые каналы доступа к файлам. Для решения этой задачи обычно применяется функция pfopen, имеющая в качестве аргументов указатель на структуру пула ресурсов и две строки, которые в точности соответствуют аргументам стандартной функции fopen, например:

```
...
FILE *f = pfopen (r->pool, r->filename, "r");

if (f == NULL) { ... } else { ... }
```

Вы можете также использовать функцию popenf, которая основана на системном вызове open. Обе эти функции обеспечивают автоматическое закрытие файла при очистке содержимого пула.

В отличие от работы с областями оперативной памяти для файлов реализованы функции закрытия каналов: pfclose и pclosef, соответствующие вызовам pfopen и popenf.

Необходимость в этих функциях возникает в тех случаях, когда операционная система ограничивает процессы в открытии достаточного числа файлов. Эти функции должны использоваться только для файлов, открытых с помощью `pfopen` и `popenf`, так как в противном случае возможно возникновение фатальных ошибок в таких операционных системах как Linux, которые очень болезненно реагируют на попытку закрытия уже закрытых файлов.

Использование функции `close` не обязательно, поскольку файл будет закрыт в любом случае, но можно подумать о ее использовании в тех случаях, когда ваш модуль открывает, или может открыть, большое количество файлов.

9.3.4 Тонкая настройка — создание и обслуживание суб-пулов и суб-запросов

В некоторых, достаточно редких случаях слишком свободное использование `palloc()` может привести к нежелательному росту временных затрат на выделение ресурсов. Вы можете справиться с этой ситуацией за счет создания *суб-пула*, и распределения ресурсов не в основном пуле, а в созданной вами структуре, а также выполнения над ним всех функций по очистке содержимого или даже его удаления, что автоматически приведет к освобождению ассоциируемых с этой структурой данных ресурсов.

Следует отметить, что подобная ситуация *действительно встречается очень редко*; единственным модулем, в котором такой подход реализован сегодня, является модуль формирования листингов каталогов, ориентированный на обслуживание *очень* больших каталогов. Использование обсуждаемых в этом параграфе примитивов без достаточной на то необходимости приведет лишь к запутыванию кода без достижения сколь-нибудь значимого выигрыша.

Основой для создания суб-пула является функция `make_sub_pool`, имеющая единственный аргумент — указатель на родительский пул ресурсов. При очистке основного пула дочерний суб-пул уничтожается. Эта операция может выполняться и вручную в любой момент времени путем вызова функций `clear_pool` или `destroy_pool`, которые очищают или уничтожают содержимое пула[14].

[14] Отличие между этими функциями состоит в том, что `clear_pool` освобо-

И напоследок еще одно замечание — суб-запросы имеют свои собственные пулы ресурсов, которые являются суб-пулами для пула ресурсов, выделенного основному запросу.Корректный способ освобождения ресурсов, выделенных субзапросу[15] состоит в использовании вызова `destroy_sub_request`, который освобождает пул ресурсов.

Однако перед вызовом этой функции вы должны позаботиться о копировании из суб-пула всех данных, которые необходимы вам для обслуживания суб-запроса[16].

Во многих случаях вы вовсе не обязаны использовать эту функцию; как правило для типичного субзапроса выделяется около 2 КБ памяти, которая в любом случае будет освобождена при очистке основного пула ресурсов. Поэтому использовать функции `destroy...` вам нужно только в тех случаях, когда вам приходится формировать очень много субзапросов для обслуживания основного запроса пользователя.

9.4 Конфигурация, команды и т.д.

Одна из целей, которые ставились при разработке сервера Apache состояла в обеспечении внешней совместимости с сервером NCSA 1.3 — то есть в использовании тех же конфигурационных файлов, корректной обработке всех директив, и вообще говоря, в использовании Apache как альтернативы серверу NCSA. С другой стороны, разработчики стремились как можно больше функций сервера реализовать в виде подгружаемых модулей, чтобы как можно меньше забот было связано с обслуживанием монолитного ядра сервера. Единственным способом одновременного решения этих обеих задач оказалось делегирование прав обработки большей части команд от центрального сервера отдельным модулям.

ждает ресурсы, ассоциированные с пулом, а `destroy_pool` уничтожает сам пул. В первом случае вы можете назначить пулу новые ресурсы, освободить их и снова назначить...а во втором ресурсы освобождаются вместе с пулом.

[15]с помощью функций `sub_req_lookup_...`

[16]К числу таких данных относится, например, имя файла в структуре `request_rec`.

В тоже время простого создания таблиц команд в модулях оказывается недостаточно для полной разгрузки от этих проблем ядра сервера. Сервер должен иметь понятие о командах для того, чтобы реагировать на них соответствующим образом. Это приводит к необходимость поддержания внутренних структур данных модулей, которые могут быть привязаны к конкретному серверу[17] или каталогу. Как правило, директивы ориентированы на обслуживание отдельных каталогов, включая контроль доступа и авторизацию пользователей, механизмы декодирования типов файлов по суффиксам их имен и так далее. Вообще говоря, основное правило при формировании обработчиков команд можно сформулировать следующим образом:

> Если обработка команды может быть ориентирована на обслуживание одного конкретного каталога, она должна быть реализована именно таким образом.

Информация, ориентированная на серверы обычно используется в стандартных наборах модулей для директив[18], которые вступают в игру до того, как запрос оказывается привязан к конкретному файлу или каталогу в файловой системе.

Еще одно важное требование к серверу, необходимое для эмуляции поведения сервера NCSA состоит в обеспечении способности обслуживать конфигурационные файлы отдельных каталогов[19]. Поэтому, после завершения трансляции *URI -> имя_файла*, но до начала выполнения любой другой фазы обработки сервер просматривает все поддерево каталогов в поисках файлов `.htaccess`. Информация, которая считывается из этих файлов *объединяется* с соответствующими данными из собственных конфигурационных файлов сервера[20].

И наконец, после обслуживания запроса, который потребовал исследования файлов `.htaccess` нам потребуется освободить память,

---

[17]В том числе и виртуальному

[18]Характерными прмерами могут служить `Alias` или `Redirect`

[19]Обычно эти файлы имеют название `.htaccess`, хотя в NCSA-сервере они часто содержат директивы, не имеющие ничего общего с контролем доступа.

[20]либо из секций `<Directory>` в файле `access.conf`, либо из настроек `srm.conf`, которые по своим функциям практически полностью повторяют `<Directory />`.

выделенную для обслуживания. Это решается так же, как и раньше за счет привязывания временных структур данных к пулу ресурсов, выделяемому на период обслуживания транзакций.

9.4.1 Конфигурационные структуры каталогов

Давайте рассмотрим, каким образом изложенные выше соображения реализуются на практике на примере модуля `mod_mime.c`, который реализует обработчик типов файлов, эмулирующий поведение сервера NCSA, определяющего тип файлов по их суффиксам. В настоящий момент нас будут интересовать фрагменты кода, которые реализуют команды `AddType` и `AddEncoding`. Эти команды могут появляться в файлах `.htaccess`, а поэтому они должны обслуживаться на уровне модуля в числе данных уровня отдельных каталогов, что на практике приводит к появлению двух отдельных таблиц: для типов MIME и способа кодирования данных. Описание этих структур приведено ниже.

```
typedef struct {
    table *forced_types;
    /* дополнительные записи AddTyped  */
    table *encoding_types;
    /* расширение AddEncoding... */
} mime_dir_config;
```

Когда сервер загружает конфигурационный файл или раздел `<Directory>`, который включает одну или несколько команд MIME–модулей, появляется необходимость в создании структуры `mime_dir_config`, над которой эти команды и осуществляют все операции.

Эта операция выполняется путем вызова функции, которая ищет необходимую информацию в конфигурационном слоте описания каталога, используя для ориентирования два аргумента: имя каталога, для которого необходимо найти данные (или `NULL`, если используется информация из `srm.conf`) и указатель на пул ресурсов в который помещается ссылка на выделяемые ресурсы.

В случае, когда осуществляется загрузка параметров из файла .htaccess в качестве пула используется ресурсный пул, создаваемый для обслуживания данного запроса; в остальных случаях — пул, который используется для хранения конфигурационных данных и очищается только при рестарте программы. В любом случае важно отметить, что создаваемые структуры уничтожаются при очистке пула.

Для рассматриваемого нами MIME-модуля создание описанной выше структуры данных осуществляется с помощью функции palloc, а кроме того, производится создание и заполнение нескольких таблиц. Эта процедура выглядит следующим образом:

```
void *create_mime_dir_config (pool *p, char *dummy)
{
    mime_dir_config *new =
      (mime_dir_config *) palloc( p,
sizeof(mime_dir_config));

    new->forced_types = make_table (p, 4);
    new->encoding_types = make_table (p, 4);

    return new;
}
```

Теперь предположим, что мы прочитали файл .htaccess. В результате мы сформировали конфигурационную структуру для следующего по иерархии каталога в дереве путей доступа. Если прочитанный нами только что файл .htaccess не содержит никаких директив AddType или AddEncoding, его структура с точки зрения модуля MIME остается верной и мы можем использовать ее без внесения каких-либо изменений. В противном случае необходимо выполнить ряд процедур по слиянию уже имеющихся данных с вновь введенными.

Для этого сервер вызывает функцию слияния конфигурационной информации модуля, если, конечно, она присутствует. Эта функция должна иметь три аргумента: две структуры, данные из которых подлежат слиянию и пул ресурсов, в который должен быть помещен результат. В случае модуля MIME все что нам необходимо сделать, так это наложить таблицы текущей конфигурации каталога на родительские:

```
void *merge_mime_dir_configs (pool *p,
                      void *parent_dirv,
      void *subdirv)
{
    mime_dir_config *parent_dir =
               (mime_dir_config *)parent_dirv;
    mime_dir_config *subdir =
               (mime_dir_config *)subdirv;
    mime_dir_config *new =
      (mime_dir_config *) palloc (p,
                          sizeof(mime_dir_config));

    new->forced_types =
         overlay_tables (p, subdir->forced_types,
                        parent_dir->forced_types);
    new->encoding_types =
         overlay_tables (p, subdir->encoding_types,
                        parent_dir->encoding_types);

    return new;
}
```

В порядке замечания — если в модуле функция слияния не определена, сервер должен просто использовать конфигурационную информацию подкаталога и игнорировать настройки родительского каталгоа. В некоторых модулях такой подход себя вполне оправдывает, например в тех случаях, когда конфигурационное состояние каталога определяется исключительно состоянием `XBITHACK`, а также в тех модулях, в которых вы не создаете собственные конфигурационные структуры, оставляя значением соответствующего слота константу `NULL`.

9.4.2 Обработка команд

После того, как мы выяснили какие структуры данных используются сервером, пора разобраться, как мы можем заполнить их полезной информацией. На практике (а мы продолжаем рассматривать модуль MIME) это означает реализацию обработки команд `AddType` и `AddEncoding`. Чтобы выявить, кто отвечает за обслуживание той или иной директивы, сервер просматривает *таблицу команд* модуля. В этой таблице хранится информация, достаточная для того, чтобы сервер мог вызвать подавляющее большинство обработчиков директив, аргументы которых уже проанализированы и помещены в соответствующие структуры.

Впрочем, перейдем к конкретному примеру и взглянем на обработчик `AddType`[21], который выглядит следующим образом:

```
char *add_type(cmd_parms *cmd, mime_dir_config *m,
               char *ct, char *ext)
{
    if (*ext == '.') ++ext;
    table_set (m->forced_types, ext, ct);
    return NULL;
}
```

Конечно, этот обработчик прост до неприличия. Как вы видите, функция использует четыре аргумента, два из которых представляют собой ранее разобранные структуры данных, третий является конфигурационной структурой текущего каталога, а четвертый является указателем на структуру `cmd_parms`. Эта структруа содержит набор полей, которые часто используются в различных командах и включает пул ресурсов, используемый обработчиком и конфигурационные данные виртуального сервера[22].

Важной особенностью приведенного выше обработчика, обусловившую его простоту, является отсутствие ошибочных ситуаций, которые эта функция могда бы обработать. Если бы нам пришлось

[21]Обработчик команды `AddEncoding` от этого фрагмента радикально не отличается.

[22]Естественно, эти данные используются только в том случае, когда в них возникает необходимость

все-таки вводить обработку ошибок, то мы поместили бы код ошибки в возвращаемое значение (вместо NULL, который используется в этой функции). Реакция сервера на ненулевой код возврата может быть достаточно разнообразна:

- Если ошибочная ситуация возникла при обработке основных конфигурационных файлов, на устройство /dev/stderr сервера выводится сообщение об ошибке, после которого работа программы httpd прекращается.

- При ошибке, обнаруженной во время загрузки файла .htaccess, формируется запись в регистрационном журнале сервера[23].

- Запрос отбрасывается с формированием серверного кода ошибки (HTTP error status, code 500).

Таблица команд модуля MIME содержит записи для этих команд, которые выглядят следующим образом:

```
command_rec mime_cmds[] = {

{ "AddType", add_type, NULL, OR_FILEINFO, TAKE2,
"a mime type followed by a file extension"},

{ "AddEncoding", add_encoding, NULL, OR_FILEINFO, TAKE2,
"an encoding (e.g. gzip), followed by a file extension"},

{ NULL }
};
```

Значения полей в этих таблицах следующее:

- Имя команды

- Функция, которая обрабатывает данную команду

[23]Одновременно с фиксацией места, где произошла ошибка

- Указатель `(void *)`, который передается в структуру `cmd_parms` обработчика команды — это оказывается полезным в тех случаях, когда один обработчик используется для обслуживания сразу нескольких команд.

- Битовая маска, указывающая, где может использоваться команда. Поддерживаются маски, соответствующие всем опциям — `AllowOverride`, дополнительные битовые маски, и `RSRC_CONF`, указывающая, что команда может использоваться в основных конфигурационных файлах сервера, но *не может* помещаться в файл `.htaccess`.

- Флаг, указывающий сколько аргументов обработчик команды рассчитывает втретить в уже разобранном виде, и каким образом они должны быть переданы для обработки. `TAKE2` означает, что будет использовано два заранее разобранных аргумента. Другими разрешенными значениями являются `TAKE1`, которое соответствует одному разобранному аргументу, `FLAG`, означающий, что аргумент может принимать значение `On` или `Off`, и передается как логическая переменная, `RAW_ARGS`, заставляющий сервер передать аргументы без анализа (вся строка, за исключением имени команды). Возможно также использование флага `ITERATE`, означающего, что обработчик выглядит аналогично `TAKE1`, но в случае, если ему передается несколько аргументов, он будет вызван несколько раз для последовательной обработки каждого из них. И наконец, `ITERATE2`, сообщающий, что обработчик имеет вид, аналогичный `TAKE2`, но при передаче ему большего количества аргументов вызывается несколько раз[24]

- Строка, описывающая аргументы. Если в реальном конфигурационном файле аргументы не нужны, эта строка используется для формирования более подробного (расширенного) сообщения об ошибочных ситуациях[25].

[24]При этом первый аргумент функции фиксируется, а при повторных вызовах изменяется значение второго.

[25]Вы можете без каких-либо угрызений совести присвоить этой строке значение `NULL`.

Теперь разберемся, как все эти данные можно использовать. Обычно все операции над этой информацией реализованы в обработчиках модулей и выглядят примерно так, как это показано в приведенном ниже фрагменте. Обратите внимание, что конфигурационные структуры каталогов извлекаются из вектора `request_rec` с помощью функции `get_module_config`.

```
int find_ct(request_rec *r)
{
    int i;
    char *fn = pstrdup (r->pool, r->filename);
    mime_dir_config *conf =
      (mime_dir_config *)
         get_module_config(r->per_dir_config, &mime_module);
    char *type;

    if (S_ISDIR(r->finfo.st_mode)) {
        r->content_type = DIR_MAGIC_TYPE;
        return OK;
    }

    if((i=rind(fn,'.')) < 0) return DECLINED;
    ++i;

    if ((type = table_get (conf->encoding_types, &fn[i])))
    {
        r->content_encoding = type;

        /* возвращаемся к предыдушему расширению и
   пытаемся использовать его как тип файла  */

        fn[i-1] = '\0';
        if((i=rind(fn,'.')) < 0) return OK;
        ++i;
    }

    if ((type = table_get (conf->forced_types, &fn[i])))
    {
```

```
        r->content_type = type;
    }

    return OK;
}
```

9.4.3 Замечания о виртуальных серверах, конфигурации и прочих экзотических режимах

Основная идея, положенная в конфигурирование модулей для работы с виртуальными серверами та же, что и при настройке сервера на обработку данных в конкретных каталогах — имеются функции генерации конфигурации и функции слияния, при этом последние вызываются в тех случаях, когда виртуальный сервер частично изменяет базовую конфигурацию и возникает необходимость в вычислении некоторой комбинированной структуры.

Так же, как и в случае настройки отдельных каталогов, если функция слияния не определена, а модуль должен быть сконфигурирован для работы с виртуальным сервером, базовая конфигурация просто игнорируется.

Единственным существенным отличием в данном случае является то, что если команда должна быть сконфигурирована с частными данными сервера, ей необходимо обратиться к данным в структуре `cmd_parms`. Ниже приведен пример, взятый из модуля обработки синонимов (alias), который также демонстрирует механизм фиксации синтаксических ошибок[26].

```
char *add_redirect(cmd_parms *cmd,
                   void *dummy, char *f, char *url)
{
    server_rec *s = cmd->server;
    alias_server_conf *conf =
        (alias_server_conf *)
```

[26] Обратите внимание, что аргумент, указывающий на конфигурацию каталога объявлен как пустой, поскольку реально модуль не имеет этой структуры и ее не обрабатывает.

```
  get_module_config(s->module_config,&alias_module);
    alias_entry *new = push_array (conf->redirects);

    if (!is_url (url)) return "Redirect to non-URL";

    new->fake = f; new->real = url;
    return NULL;
}
```