Администрирование Web-сервера APACHE и руководство по электронной коммерции

- Конфигурация, управление и создание узлов электронной коммерции с помощью Web-сервера Apache
- Интеграция баз данных, доставка динамического содержимого, безопасность транзакций и многое другое
- Поддержка всех основных ОС: Unix, Linux, Windows и даже MacGSX
- Справочник команд популярного языка PHP

СКОТТ ХОКИНС

ЦИКЛ ПО ТЕХНОЛОГИЯМ 'ОТКРЫТЫЕ СИСТЕМЫ

АДМИНИСТРИРОВАНИЕ WEB-СЕРВЕРА APACHE И РУКОВОДСТВО ПО ЭЛЕКТРОН НОЙ КОММЕРЦИИ

СКОТТ ХОКИНС

Издательский дом "Вильяме"
Москва, Санкт-Петербург, Киев
2001

ББК 32.973.26-018.2.75
Х68
УДК 681.3.07

Издательский дом "Вильяме"

Зав. редакцией *С.Н. Тригуб*

Перевод с английского и редакция *Н.В. Воронина*

По общим вопросам обращайтесь в Издательский дом "Вильямс"
по адресу: info@williamspublishing.com, http://www.wiiliamspublishing.com

Хокинс, Скотт.

Х68 Администрирование Web-сервера Apache и руководство по электронной коммерции. : Пер. с англ. М. : Издательский дом "Вильяме", 2001. — 336 с. : ил. — Парал. тит. англ.

ISBN 5-8459-0212-6 (рус.)

Эта книга задумывалась как достаточно полное справочное руководство по Web-серверу Apache. Изложенный в ней материал предполагает определенный уровень компьютерной грамотности, но знания сетевых технологий при этом не требуется. Несмотря на то, что основная проблематика книги лежит в области электронной коммерции, в приложениях затронуты самые разнообразные проблемы и информация, необходимая для создания и функционирования Web-сервера. Это проблема соответствия имен и IP-адресов, детали протокола TCP/IP и синтаксис регулярных выражений. Кроме того, в перспективе Web-администрирования затронуты темы создания системы электронных платежей и взаимодействия с базами данных.

ББК 32.973.26-018.2.75

ISBN 5-8459-0212-6 (рус.)
ISBN 0-13-089873-2 (англ.)

Оглавление

Содержание

Предисловие

Резюме

Эта книга задумывалась как достаточно полное справочное руководство по Web-серверу Apache. Изложенный в ней материал предполагает определенный уровень компьютерной грамотности, но знания сетевых технологий при этом не требуется. Несмотря на то, что основная проблематика книги лежит в области электронной коммерции, в приложениях затронуты самые разнообразные проблемы и информация, необходимая для создания и функционирования Web-сервера. Это проблема соответствия имен и IP-адресов, детали протокола TCP/IP и синтаксис регулярных выражений. Кроме того, в перспективе Web-администрирования затронуты темы создания системы электронных платежей и взаимодействия с базами данных.

Apache

Web-сервер Apache называют самым главным сокровищем движения "открытые программные системы". Его можно получить совершенно бесплатно. Он имеет отличные рабочие характеристики и поэтому используется более широко, чем все остальные Web-серверы вместе взятые. В настоящий момент 61,5 процентов всех Web-узлов в мире созданы с использованием сервера Apache.

Распространение "открытых программных систем" во многом аналогично процессу естественного биологического отбора — все ОС Linux и утилиты sendmail мира постепенно заполняют обложку журнала "Time" и перемалываются рекламной машиной в то время как легионы DOS-утилит медленно, но неотвратимо приближаются к устройству /dev/null истории. Сервер Apache никогда не был бы настолько популярен, если бы он не работал надежно.

Сервер Apache имеет еще одно преимущество не присущее остальным открытым системам: он так прост, что любой достаточно квалифицированный пользователь может овладеть им во всей его полноте. Видит Бог, я не являюсь большим поклонником Microsoft, но если передо мной поставить задачу выбора между ОС Linux и ОС Windows 2000, я бы мгновенно выбрал Windows — не успели бы вы и глазом моргнуть. И, между прочим, это нельзя рассматривать как неуважение к Linux: операционные системы, в частности многопользовательские, очень сложны. Единственный способ сделать их доступными для среднего пользователя — это упрощение.

К счастью, набор задач, которые можно решить с помощью сервера Apache, не настолько широк. Те из вас, кто начинает изучение сервера с неуверенностью и чувством страха перед новым, сможет с облегчением узнать, что сами по себе процедуры конфигурирования и обслуживания сервера не являются очень сложными. Суть освоения сервера, в зависимости от уровня вашего опыта, заключается в освоении основных концепций операционной системы, освоении команд, которые помогут заставить машину делать то, что вам нужно, и освоении жаргона. Если вы являетесь докой в одной из указанных проблематик, вас ожидает *действительно* приятный сюрприз.

Краткая история

Сервер Apache берет свое начало от сервера httpd, созданного **Робом Макколом (Rob McCool)** в Национальном центре по применению суперкомпьютеров (National Center for Supercomputing Applications — NCSA). В 1995 году сервер httpd был самым популярным из существовавших тогда Web-серверов, но когда в 1994 **Маккол** покинул

NCSA, развитие программы замерло. Поэтому для его поддержки и развития небольшая группа Web-администраторов сплотилась и образовала ядро организации, которая теперь хорошо известна как "Apache group". Ее членами являются

Брайан Белендорф (BrianBehlendorf)
Рой Т. Филдинг (Roy T. Fielding)
Роб Хартилл (Rob Hartill)
Дэвид Робинсон (David Robinson)
Клиф Скольник (Cliff Skolnick)
Рэнди Тербуш (Randy Terbush)
Роберт Тау (Robert S. Thau)
Эндрю Вильсон (Andrew Wilson)

При тесном сотрудничестве с Эриком Хагбергом (Eric Hagberg), Фрэнком Петерсом (Frank Peters) и Николасом Пиочем (Nicolas Pioch), "Apache group incorporated" опубликовала исправления ошибок для httpd 1.3, добавила несколько новых возможностей и в апреле 1995 выпустили очередную версию сервера под именем Apache О.6.2.

С тех пор "Apache group", так они вскоре стали известны, посвятила себя настройке и усовершенствованию программного обеспечения. Сейчас имеются версии практически для всех основных операционных систем, хотя платформа Unix среди них является бесспорным лидером.

По своей сути Web-сервер Apache является конечным результатом грандиозного совместного труда группы программистов самой высокой квалификации. Возникает естественный вопрос, что их подвигло на работу над Apache вместо того, чтобы за хорошие деньги разрабатывать обычное коммерческое программное обеспечение. Здесь можно процитировать Web-узел *www.apache.org:*

"Сервер Apache существует для того, чтобы обеспечить надежные решения на коммерческом уровне с использованием протокола HTTP. Он является платформой, на основании которой как частные лица, так и организации могут создавать надежные системы и для экспериментальных, и критическихзадач. Мы верим, что публикуемый в Internet инструментарий, попадая в руки любому желающему или компаниям, занимающимся разработкой программных продуктов, смогут помочь сделать деньги, создавая дополнительные услуги по созданию специализированных модулей и оказывая услуги по технической поддержке. Мы понимаем, что "владение" рынком является экономическим преимуществом, а в программной индустрии это означает, что достаточно контролировать поступления платежей от пользователей программного продукта. Обычно такого положения вещей можно достичь "владением" протоколом, с помощью которого компании ведут свой бизнес. По той причине, что протоколы, использующиеся в Internet, являются "ничейными", он все ещеостается полем действия больших и маленьких компаний. Таким образом, представляется возможным предотвратить "частное владение" протоколом и обеспечить существование надежного программного продукта, использующего протокол, доступного абсолютно бесплатно для всех компаний, а это недооценить невозможно."

Открытые программные средства

Web-сервер Apache можно смело отнести к плеяде так называемых открытых программных продуктов. Традиционно поставляемые архивированные прикладные программы обычно содержат только исполняемый объектный код, а не исходный код, из которого программа была скомпилирована. Apache и другие подобные ему продукты включают в свои дистрибутивы не только исполняемый объектный код, но и исходный код, из которого был создан этот объектный код.

С точки зрения конечного пользователя это имеет смысл. Например во время работы над этой книгой у нас в офисе возникла масса проблем. Большое коммерческое про-

граммное обеспечение, которое работало на большой коммерческой же операционной системе перешло в безответное состояние. Оно перестало реагировать на ввод. Мы попытались отследить стек и некоторые другие вещи, но, за неимением на руках исходного текста, не смогли предпринять никаких существенных шагов. Поэтому была снята вся возможная диагностика и вместе с программным обеспечением передана на анализ поставщику. Совершенно естественно, проблема была решена, но это заняло две недели.

Следовательно, это очень сложная процедура. Если бы мы работали с открытым программным средством, то решение обязательно бы было найдено значительно быстрее. А в этом конкретном случае не было понятно, в чем, собственно, заключается проблема, и даже если бы мы знали причину, то определенно не смогли бы ее исправить. Мы полностью находились во власти нашего поставщика.

Конечно, работа с "открытыми системами" несколько необычна. Ведь в случае работы с открытым программным обеспечением нет никого, кто бы непосредственно отвечал за сопровождение открытой версии сервера Apache, нет бесплатного номера, по которому можно позвонить в 2 часа ночи, если что-то произошло с сервером[1] и неизвестно, что делать. Поддержка существует в виде группы новостей и Web-узлов, но она приходит тогда, когда это удобно тому, кто осуществляет поддержку, а не тому, кому она нужна.

Web-сервер Apache, как и другие программные продукты класса "открытые программные продукты", только выигрывают от их постоянной доступности для программистского сообщества. Вследствие того, что над каждым "открытым" проектом работает гораздо больше разработчиков, чем могла бы нанять даже самая богатая корпорация, ошибочный код обнаруживается и исправляется значительно быстрее.

Я позволю себе предположить, что качество открытого текста обычно выше, чем у коммерческих продуктов. Ведь основной мотивацией разработчиков, работающих над "открытыми" продуктами, является любовь к программированию как к творческому процессу. Таким образом, при работе с "открытыми продуктами" можно получить самые лучшие образцы программ. Это разительно контрастирует с коммерческим продуктом, при работе с которым большая часть рабочего времени тратится на встречи, телефонные разговоры и, в конечном счете, инвентаризацию остатков.

Структура этой книги

Книга рассчитана на администраторов сервера Apache самой различной квалификации. Предполагается знакомство с основными концепциями, используемыми в компьютерной индустрии. Специальной подготовки для Web-администрирования не требуется. Для тех, кто совершенно не знаком с сервером Apache, имеется достаточно большой вводный раздел, в котором можно найти все, что необходимо для того, чтобы как можно скорее войти в курс дела.

После ознакомления с основными концепциями у вас может возникнуть желание побыстрее реализовать одну или две возможности сервера (например виртуальный хостинг). Главы этой книги скомпонованы в независимые отдельные очерки, посвященные самым различным темам. И с ними можно ознакомиться в произвольном порядке.

Вследствие своей открытости миру сервер Apache постоянно совершенствуется. Каждый день предлагаются и добавляются новые возможности. Иногда это усовершенствование базовых функций сервера, но чаще всего — новые или усовершенствованные модули. По этой причине, даже опытные администраторы время от времени должны расширять область своей компетенции. Надеюсь, что эта книга может предложить что-нибудь существенное вниманию даже опытных администраторов Apache.

[1] В обшем это так, но здесь хочется сделать некоторые уточнения. В соответствии с лицензией Apache в исходный текст можно вносить любые изменения, можно даже его перепродать.

Наконец, в приложениях обсуждается масса проблем, посвященных работе в сетях и программированию. Многие из них непосредственно с сервером Apache не связаны, но так как серверу Apache необходима работающая сеть, ваша карьера как Web-администратора скорее всего потребует определенных знаний по общему администрированию сетей и программированию. Приложения нельзя назвать всеобъемлющими, но они могут оказаться весьма полезными.

Часть I, "Основы"

Часть I посвящена основным концепциям и методам администрирования сервера Apache. Она проведет вас через процедуру получения и инсталляции сервера Apache, настроенного в соответствии с вашими потребностями. Если вы не были знакомы с сервером Apache ранее, то, скорее всего, придется последовательно прочесть все четыре главы, а особенно внимательноглаву 1, "Основные концепции". Информации, содержащейся в главах с 1 по 4, вместе с приложениями, будет вполне достаточно, чтобы смог начать работу любой новичок.

Часть II, "Администрирование Web-сервера"

В части II описаны более сложные функции сервера Apache. И если вы освоили основные концепции сервера, то можете пропустить главы, содержащиеся в этой части. Каждая глава задумывалась как самодостаточный очерк на тему, которая видна из самого названия главы.

Часть III, "Электронная коммерция"

В части III обсуждаются проблемы, имеющие непосредственное отношение к электронной коммерции, взаимодействию с базами данных и ме ханизмами заработка денег с помощью Internet. Последняя глава этой части посвящена конкретному исследованию создания инфраструктуры коммерческого узла.

Часть IV, "Приложения"

В приложениях имеется информация, которая слишком узкоспециальна для того, чтобы быть включенной в остальные части книги (например синтаксис директив), или которую можно охарактеризовать как сопутствующую информацию (например приложение Е, "Концепция Unix"). Некоторые из них представлены в виде кратких обучающих очерков, другие — как простой справочный материал.

Как работать с этой книгой

В материале, изложенном в книге, можно встретить примеры команд и конфигурационных директив, обычно сопровождающиеся объяснениями, а иногда распечаткой. Подробной информации о синтаксисе директив и системных команд в основных главах нет. Такую информацию можно найти в приложениях, в частности в приложении А, "Основные директивы", и Б, '"Прочие директивы". Надо надеяться, что по виду незнакомой вам команды можно определить ее назначение.

Успех или неудача любой конкретной транзакции сервера Apache зависит от внутренней конфигурации сервера, передаваемого содержимого, конфигурации операционной системы и капризов сервисов поддержки сети. Поэтому нельзя сказать со всей уверенностью будут ли примеры, представленные здесь, работать на конкретной машине. Автор только может со всей ответственностью поклясться в том, что у него все ониработали.

В некоторых случаях начальная тематика определенной главы была заложена в предыдущей главе. Например виртуальные узлы, о безопасности которых говорится в главе 8, "Безопасность", были рассмотрены в главе 4, "Запуск, перезапуск и остановка". Если

смысл определенной конфигурационной директивы неясен, можно посоветовать обратиться к предметному указателю или прочесть о ее назначении в предыдущих главах.

Следует подчеркнуть, что в примерах команд операционной системы предпочтение делается в пользу команд ОС Unix в целом и команд ОС Linux в частности. В этом нет никакого совпадения, ведь при написании этой книги в качестве тестового сервера использовалась именно Linux-машина, некоторая часть работ проводилась на ОС Windows NT, Windows 95 и Mac OS X.

Одно из соглашений для ОС Unix, которым я очень горжусь, но которого нельзя встретить нигде в документации по серверу Apache, это практика объявления переменной окружения. В ней может храниться длинный путь к каталогу. В книге можно встретить системную переменную ОС Unix $APACHE, хранящую имя каталога, в котором установлен Web-сервер Apache. (В конфигурационных файлах этот каталог упоминается под именем ServerRoot.)

Типографские соглашения

Для отображения команд, директив, имен и сообщений в системе Unix используется специальный шрифт:

Пример директивы с параметрами

Большие примеры конфигурационных файлов обычно выделяются отдельно.

```
ДирективаA
ДирективаB
ДирективаC
ДирективаD
```

Наконец, я попытался подразделить описание директив с применением заголовков, имеющих определенный смысл. В то время, как большинство директив сервера Apache имеют имена, которые дают достаточно опытному пользователю подсказку об их функциях (например, AddModule и Port), другие относятся к возможностям, присущим только Web-серверу Apache, и описать их с помощью двенадцати символов представляется достаточно затруднительным (например DocumentRoot).

Все вопросы, комментарии, исправления или предложения по усовершенствованию можно посылать по адресу *s_hawkins@mindspring.com*.

Гидра

Вместо послесловия, примите мои соображения о значении создания, изображенного на обложке этой книги. Это гидра, она была нарисована парнем по имени **Том Пост (Tom Post).** Мифологические звери стали традиционной тематикой оформления обложек книг, посвященных открытым системам, издаваемых издательством "Prentice Hall". Таким образом, гидра является довольно подходящим олицетворением Apache. Ведь многочисленные экземпляры Apache, работающие одновременно на одном компьютере, очень напоминают многочисленные головы гидры. Разве это не так? За столь удачный выбор мифологического образа с острыми зубами хочется поблагодарить моего редактора **Майлса Уильямса** (Miles **Williams),** а также за то, что он не надоедал мне всякими единорогами, нимфами и прочей нечистью.

Часть I

Основы

В этой части...

1. Основные концепции
2. Инсталляция Web-сервера Apache
3. Конфигурирование Web-сервера Apache
4. Запуск, перезапуск и остановка

Глава

1

ОСНОВНЫЕ КОНЦЕПЦИИ

В этой главе...

1.1. Введение

В этой главе читатель получит представление об основных концепциях, на которых базируется Web-сервер Apache. Другие главы данной книги в основном опираются на терминологию, которая здесь поясняется, и поэтому, если вы незнакомы с такими терминами, как *директива, модуль* и т.д., то, вероятно, вам следует прочитать эту главу.

Большинство методов, примененных в Web-сервере Apache, заимствованы из стандартной доктрины программирования. Однако, если вы не программист или совершенно ничего не слышали о Web-сервере Apache, ознакомление с этой главой значительно ускорит усвоение материала.

1.2. Конфигурационные файлы

Web-сервер Apache — хорошо конфигурируемая программа. Есть тысячи возможных комбинаций значений для сотни конфигурационных переменных. Без сомнения тот, кто работал с ОС Unix или ОС DOS, знаком с концепцией опций, задаваемых в командной строке. Основной принцип, заложенный в концепцию директивы, аналогичен концепции опции, задаваемой в командной строке. Но здесь, вместо того чтобы задавать при каждом запуске Web-сервера дюжину или сотню конфигурационных переменных Apache, параметры собраны в одном конфигурационном файле, который автоматически считывается сервером во время запуска. Первоначально все переменные, которыми задавались режимы работы Web-сервера, содержались в трех файлах.

httpd.conf	Основной конфигурационный файл, содержащий переменные, задающие конфигурационную информацию сервера.

srm.conf	Управление ресурсами сервера — исторически сложилось так, что данный файл содержит переменные, определяющие, каким образом ресурсы сервера будут использоваться.
access.conf	Исторически сложилось так, что данный файл содержит переменные, имеющие отношение к управлению доступом.

В настоящее время практическое использование всех трех файлов считается анахронизмом. Однако такая возможность все еще поддерживается для совместимости с ранними версиями Web-сервера Apache. *Сейчас все конфигурационные директивы можно поместить в конфигурационном файле* httpd.conf.

По умолчанию конфигурационные файлы хранятся в каталоге conf, который находится в главном каталоге сервера Apache. Как и все остальное окружение Web-сервера Apache, размещение и имена этих файлов легко меняется, но для простоты изложения материала здесь и далее воспользуемся стандартными именами файлов и каталогов. В главе 3, "Конфигурирование Web-сервера Apache" читатель найдет полную справку по методам размещения и определения главного каталога Web-сервера Apache.

1.3. Директивы

Как подчеркивалось выше, значения конфигурационных переменных хранятся в конфигурационных файлах. Эти переменные и есть *директивы.* Основное время администрирования «сервера Apache (90 %) затрачивается на определение того, значения каких директив будут изменяться, и значения, которые эти директивы должны задавать[1].

Следует обратить внимание на то, что далеко *не все директивы автоматически распознаются сервером Apache.* Существует достаточно большое подмножество директив, которые называются *основными.* Эти директивы устанавливаются по умолчанию. Другие директивы распознаются в зависимости от того, какой набор модулей скомпилирован при построении данного конкретного сервера Apache. Поэтому указание директивы в конфигурационном файле совсем не означает, что она возымеет какое-либо действие: работающий вариант сервера вполне может не иметь модуля, который взаимодействует с данной директивой. Чтобы получить перечень скомпилированных модулей, достаточно ввести команду:

```
httpd -l
```

Обратите внимание на то, что разделяемые объектные модули 2, "Инсталляция Web-сервера Apache" и 4, "Запуск, перезапуск и остановка" во время работы сервера будут загружены и не будут отображаться данной командой. Более полную информацию по этой проблеме можно найти в разделе "Модули".

1.4. Ограничение диапазона действиядирективов

Далеко не все директивы применяются одновременно. Иногда очень полезно иметь один набор директив для одного варианта сервера, а другой набор директив — для другого варианта.

Наглядным примером может служить случай, когда один и тот же сервер обслуживает более одного Web-узла. Такая практика поясняется в главе 5, "Хостинг нескольких Web-узлов" Сейчас можно только предложить принять на веру, что два узла (например ***www.christiffnsite.org*** **и** *www.muslimsite.org)* могут обслуживаться одной и той же программой на одном и том же компьютере. В таком случае для каждого узла задаются различные директивы, и, как минимум, у них должны быть разные имена.

[1] Другие девяносто процентов рабочего времени тратится на перекомпиляцию сервера

За исключением виртуальных узлов (они обсуждаются позже) диапазон конкретной директивы может быть ограничен тремя различными способами:

- По каталогу с помощью директив <Directory>, <DirectoryMatch> или файла .htaccess.
- По URL (Unified Resource Locator) с помощью директив <Location> и <LocationMatch>.
- По файлу с помощью директив <Files> и <FilesMatch>.

1.4.1. Ограничение диапазона действия директив скобками <Directory> И <DirectoryMatch>

Предположим, диапазон действия некоторой директивы необходимо ограничить каталогом /home/ site2 и его подкаталогами. С точки зрения этого примера совсем не важно, какую функциональную нагрузку несет эта директива, важно то, что требуется ограничить ее действие деревом, корнем которого является каталог /home/site2, для того, чтобы она не действовала на другие структуры рассматриваемого узла. Воспользуемся для этого условной директивой DirectiveA. Чтобы ограничить зону действия DirectiveA каталогом /home/site2 и всеми подкаталогами, содержащимися в нем, достаточно эту директиву взять в скобки <Directory>:

```
<Directory  /home/site2>
        DirectiveA
</Directory>
```

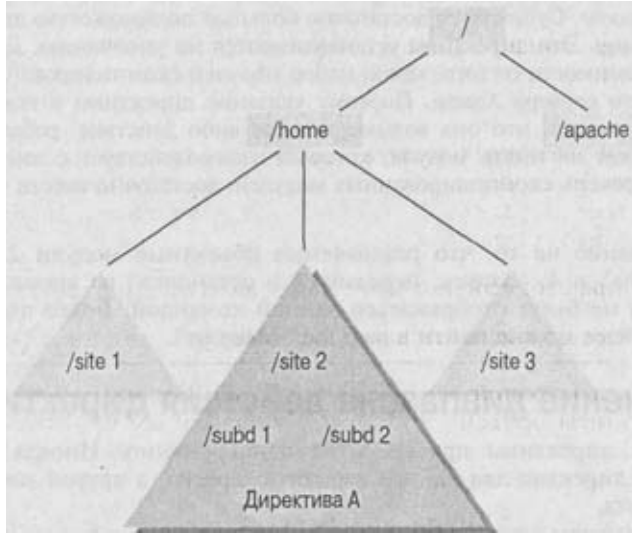

Рис. 1.1. Диапазон действия директивы ***DirectiveA***

Директива <DirectoryMatch> действует во многом аналогично директиве <Directory> за исключением того, что в ней в качестве аргумента уже указывается выражение, а не конкретное имя каталога. Например команде

```
<DirectoryMatch    "/home/site[1-3]">
        DirectiveA
</DirectoryMatch>
```

будут удовлетворять Три Каталога /home/sitel, /home/site2 И /home/site3.

1.4.2. Ограничение диапазона действия директив с помощью файлов .htaccess

Другой способ выполнить аналогичную задачу заключается в создании специального файла с конфигурационными директивами и хранении этого файла в каталоге, по отношению к которому эта информация будет применяться. По умолчанию такие файлы имеют имя .htaccess. Если такое имя по каким-либо причинам не подходит, его можно заменить на любое другое директивой AccessFileName.

Чтобы директивы, указанные в файле .htaccess, заработали, сервер Apache должен знать, что такие файлы существуют и что он должен их найти. Это можно осуществить с помощью директивы AllowOverride. Однако функциональная нагрузка директивы AllowOverride заключается не только во включении—отключении. С ее помощью можно определить, *какие типы* директив могут быть указаны в файле .htaccess. Возможные варианты указаны в табл. 1.1.

Таблица 1.1. Директива AllowOverride

Опции	Действие
All	Разрешает использование всех директив
None	Запрещает использование всех директив. Поиск файлов невозможен
AuthConfig	Разрешает использование директив AuthDBMGroupFile, AuthDBMUserFile, AuthGroupFile, AuthName, AuthType, AuthUserFile, require
FileInfo	Разрешает использование директив AddEncoding, AddLanguage, AddType, DefaulType, ErrorDocument, LanguagePriorty
Indexes	Разрешает использование директив AddDescription, AddIcon, AddIconByEncoding, AddIconByType, DefaultIcon, DirectoryIndex, FancyIndexing, HeaderName, IndexIgnore, IndexOptions, ReadMeName
Limit	Разрешает использование директив allow, deny и order
Options	Разрешает использование директив options и XBitHack

Можно задать больше одной опции одновременно. Например, для того чтобы проигнорировать директивы Options и FileInfo содержимым файла .htaccess, необходимо указать:

```
AllowOverride Options FileInfo
```

То, что файлы .htaccess влияют на подкаталоги так же, как и на каталоги, в которых они размещаются, означает, что, когда кто-либо пытается получить доступ к файлам в каталоге /home/site2/cgi-bin, сервер Apache сначала ищет файл с именем .htaccess (или как он был вами назван) в упомянутом каталоге *и во всех каталогах, находящихся в иерархическом дереве над данным каталогом.* Как можно догадаться, такая процедура сильно снижает производительность сервера. Поэтому, если нет настоятельной необходимости в файлах .htaccess, их использование следует запретить следующим образом:

```
AllowOverride none
```

1.4.3. Ограничение диапазона действия директив по URL: <Location> и <LocationMatch>

Аналогично тому как директива <Directory> ограничивает диапазон директив в пределах файловой системы, директива <Location> ограничивает диапазон директив в пределах Web-адресов. В действительности директивы <Location> и <Directory> могут иметь аналогичное действие. Но существенное различие между этими директивами заключается в том, что директива <Location> не обязательно указывает на конкретное физическое положение в файловойсистеме.

Обычно директивы размещения задают подадрес с допущением, что он находится на данном сервере. Предположим, например, что мы проводим настройку сервера с именем *www.example.com* с помощью следующей директивы:

```
<Location /status>
    SetHandler server-status
</Location>
```

Сервер Apache активизирует обработку состояния сервера в случае, когда запрашивается следующий адрес:

```
http://www.example.com/status
```

Не следует переживать, если вы никогда ничего не слышали о дескрипторах. Это просто способ сообщить серверу Apache, что нужно делать, когда во время доступа к определенным узлам, операций, выполняемых по умолчанию, окажется уже недостаточно. Эта проблема обсуждается детально вданной главе. Главное заключается в том, что, если имя сервера не задано, сервер будет полагать, что любой указанный адрес имеет отношение к локальному серверу.

1.4.4. Ограничение диапазона действия директив с помощью виртуального узла

Для полноты картины нужно упомянуть и возможность создания виртуального узла на локальном сервере. Термин *виртуальный* здесь относится к серверу Apache, который нужно сконфигурировать так, чтобы он был в состоянии реагировать на запросы, поступающие более чем к одному серверу. Например небольшая компания, предоставляющая услуги по размещению Web-узлов, может иметь десятки или сотни Web-узлов, работающих на одном и том же оборудовании и обслуживаемых одним экземпляром сервера Apache. При правильном подходе (конфигурировании) сервер успешно справляется с управлением множества виртуальных узлов, создавая при этом видимость того, что каждый из них работает совершенно самостоятельно. Как это можно сделать, читатель узнает в главе 5, "Хостинг нескольких Web-узлов".

1.4.5. Ограничение диапазона действия директив С ПОМОЩЬЮ директив <Files> И <FilesMatch>

Директивы <Files> и <FilesMatch> концептуально тоже подобны директивам <Directory> и <DirectoryMatch>. Какнетрудно догадаться, различие заключается в том, что они применяются котдельным файлам, или, как в случае с директивой <FilesMatch>, только к отдельным файлам, удовлетворяющим указанным выражениям. Вот, например, директива, которая делает конфигурационный файл .htaccess недоступным для всех:

```
<Files .htaccess>
    Order deny, allow
```

```
    Deny from all
</Files>
```

Следует отметить, что в отличие от прочих вышеперечисленных директив, которыми задаются диапазоны, директивы <Files> и <FilesMatch> можно включить в файл .htaccess.

1.5. Модули

Как уже указывалось, сервер Apache имеет ядро, гарантирующее выполнение основных функций. Ядро обеспечивает работу директив, возможность чтения конфигурационных файлов, усеченную возможность управления доступом, возможность дополнения функциональных возможностей и полдесятка других основных функциональных возможностей. В частности, директивы, которые перечислены в приложении А, "Основные директивы", всегда присутствуют в стандартном дистрибутиве сервера Apache.

Кроме того, сервер Apache разработан таким образом, что всегда существует возможность варьирования основных функциональных возможностей. Функциональные части могут быть и не подключены к первоначальной исполняемой программе. Эти субсекции называются *модулями,* причем достаточно значительная их часть поставляется по умолчанию в стандартном дистрибутиве. Чтобы получить перечень подключенных модулей, можно воспользоваться опцией -l:

```
httpd -l
```

Включать или не включать определенный модуль в работающий исполняемый код сервера в процессе компиляции или с помощью директив AddModule и ClearModuleList, решает администратор сервера (предположительно наш читатель). В случае разделяемых объектных файлов модулями можно динамически управлять с помощью директивы LoadModule (см. раздел "Динамически разделяемые объекты" в этой главе).

Большое значение имеет порядок загрузки модулей. Возможность управлять порядком загрузки модулей (и, вероятно, исключить некоторые из модулей, загружаемых по умолчанию) реализуется с помощью директивы ClearModuleList :

```
ClearModuleList
```

После этого необходимо немедленно перебрать список загружаемых модулей последовательностью директив AddModule:

```
AddModule mod_access.c
```

Если модуль включен, он становится составной частью исполняемого процесса httpd с тем же идентификатором процесса и доступом к тем же системным ресурсам. CGI-программы, которые могут быть на Web-узле, отличаются от программы httpd, их вызвавшей, и общаются с ним через ресурсы обмена данными между процессами, а это *значительно более* медленный метод. В том случае, если скорость является решающим фактором, нужно создавать новый модуль.

Сервер Apache создавался с применением модульной идеологии. На различных стадиях его работы ядро httpd опрашивает все программы, которые были включены, на предмет того, что они "знают" об обработке конфигурационных файлов, исходных текстов http на локальных узлах и запросов пользователей. Именно в таком порядке.

Методы создания собственных модулей достаточно сложны, но вполне доступны. Свой Apache-модуль может создать любой желающий. Более того, большинство самых полезных модулей было разработано сторонними разработчиками для решения своих насущных задач, и только позднее включено в состав дистрибутива Web-сервера Apache. Полное описание процесса создания нового модуля можно найти в главе 12, "Состав модуля"

1.6. Динамически разделяемые объекты

Многие приложения Unix имеют возможность определения того, какие части будут включены, а какие отключены во время исполнения. Вот термин, которым обозначаются такие части — *разделяемые объекты*. Метод разделяемых объектов широко используется сервером Apache.

Модуль, загружаемый в качестве разделяемого объекта, компилируется с помощью программы apxs (APache eXtenSion). Модули сторонних разработчиков (mod_perl, mod_php) обычно поставляются со своими собственными инструкциями по компиляции. На некоторых платформах среди основных возможностей сервера Apache требуется наличие возможности динамических разделяемых объектов (DSO). Это необходимо для того, чтобы редактор связей экспортировал таблицу перекрестных ссылок для дальнейшего использования сторонними разработчиками. Чтобы активизировать возможность работы с динамическими разделяемыми объектами, достаточно указать опцию в конфигурационном сценарии:

```
--enable-rule=SHARED_CORE
```

Затем требуется перекомпилировать, перестроить и переустановить программу httpd, как указано в главе 2, "Инсталляция Web-сервера Apache".

Модуль mod_so позволяет во время исполнения задавать, какие модули будут загружены с помощью директивы LoadModule. В этой директивезадается имя модуля и путь к файлу разделяемого объекта:

```
LoadModule perl_module libexec/libperl.so
```

Обычно (и по умолчанию) такие файлы можно найти в библиотечном каталоге libexec, который находится в каталоге ServerRoot.

1.7. Дескри пторы

Иногда модули представляют в распоряжение специальные дескрипторы, которые являются методами обработки файлов или запросов каким-то специальным способом. Иногда дескрипторы именуются таким образом, что к ним можно обращаться непосредственно с помощью конфигурационных директив. Дескрипторы и связанные с ними модули перечислены в табл. 1.2.

Таблица 1.2. Дескрипторы и соответствующие модули

Дескриптор	*Модуль*	*Действие*
send-as-is	mod_asis	Обслуживать файлы и заголовки в их текущем состоянии
cgi-script	mod cgi	Выполнение CGI-сценариев
imap-file	mod imap	Файл правил обработки изображений
server-info	mod info	Отображение конфигурационной информации сервера
server-parsed	mod include	Найти и заместить все вставленные на стороне сервера модули
server-status	mod status	Отображение информации о статусе сервера
type-map	mod negotiation	Анализировать как файл карты типа

Как показано в столбце "Модуль" в табл. 1.2, для получения доступа к определенному дескриптору необходимо, чтобы в текущем httpd был включен соответствующий модуль.

Дескрипторы активизируются с помощью директивы AddHandler. Например, с помощью директивы

```
AddHandler cgi-script .pl
```

можно задать передачу всех файлов с расширением .pl на обработку дескриптору cgi-script.

Другим способом активизации определенного дескриптора может служить директива SetHandler. Директива SetHandler предназначена для использования внутри скобок <Directory> или <Location>. Например, для того, чтобы все файлы, содержащиеся в каталоге /images обрабатывались как файлы правил обработки изображений, можно воспользоваться следующей директивой:

```
<Location /images>
      SetHandler imap-file
</Location>
```

1.8. MIME-типы

Термин MIME является аббревиатурой термина Multimedia Internet Mail Extensions (Расширения Мультимедиа дляпочты в Internet). Смысл, заложенный в концепции MIME-типов, заключается в том, чтобы дать программам возможность определить тип данных, содержащихся в файле, по расширению файла. По умолчанию MIME-тип сервера Apache и соответствующие им расширения можно найти в файле mime.types, который находится в каталоге conf. Конечно расположение этого файла можно изменить. Для этих целей воспользуемся директивой TypesConfig. Например

```
TypesConfig /etc/mime.types
```

позволит разместить файл mime . types в каталоге /etc.

Для ассоциации определенного MIME-типа с определенным расширением файла обратимся к директиве AddType. Например, директива

```
AddType application/x-httpd-php .php
```

сообщает серверу, что файлы, с расширением .php, содержат данные в формате HTML с кодом PHP4. С помощью MIME-типов сервер Apache определяет, какой тип предварительной обработки файлов требуется перед их доставкой пользователям. При работе с MIME-типами рекомендуется пользоваться директивой AddType и воздерживаться от прямого внесения изменений в файлmime.types.

Еще одной весьма полезной директивой является директива DefaulType. Предполагается, что сервер должен информировать клиентов о типе передаваемых им данных. Если ничего другого не остается, задайте с помощью директивы DefaultType тип передаваемых данных:

```
DefaultType text/html
```

Глава

2

ИНСТАЛЛЯЦИЯ WEB-СЕРВЕРА APACHE

В этой главе...

21. Введение

В этой главе детально рассмотрена процедура получения, компиляции и инсталляции Web-сервера Apache. В зависимости от того, под управлением какой операционной системы сервер будет работать, некоторые составные описанной здесь процедуры опущены. Например, в ОС Red Hat Linux сервер инсталлируется во время инсталляции самой операционной системы. У вас, возможно, возникнет необходимость оптимизировать уже работающий сервер или просто потренироваться в компилировании и настройке сервера.

2.2. Выбор аппаратной части

Выбор компьютера в большей или меньшей степени зависит от того, какие задачи вы собираетесь решать с помощью вашего Web-сервера. Например, если круг задач ограничивается желанием приобрести опыт или необходимостью отработать какую-либо рабочую концепцию, для этих целей вполне подойдет старый компьютер. Сервер Apache достаточно надежно работает на любом "железе" с процессором до 486 включительно. При более честолюбивых целях рекомендуем обратиться к Web-узлу www.apache.org, где можно найти дистрибутивы для ОС HP-UX, AIX и других "крутых" серверов.

2.2.1. Оперативная память

Самым узким местом, влияющим на производительность сервера, является оперативная память. Сервер Apache, работающий под управлением ОС Unix, при

запросе одного клиента порождает один процесс[1]. Ваша задача заключается в том, чтобы обеспечить сервер оперативной памятью такого объема, чтобы все порожденные сервером процессы могли бы одновременно там уместиться. Если оперативной памяти при этом не хватает, ОС будет вынуждена выгружать часть порожденных процессов в виртуальную память (на жесткий диск) и периодически выгружать/загружать их из/в физической памяти в очень медленном режиме, известном как "подкачка".

Начинать планирование профессиональной системы необходимо с выбора модулей, которые включает в себя сервер. Скомпилируйте сервер с этими модулями и определите размер полученного файла httpd. Умножьте полученный результат на число клиентских запросов, которые планируется обрабатывать одновременно, а затем для получения приблизительной оценки объема необходимой памяти умножьте полученный результат на 1,5. Следует помнить, что данная формула необходима только для того, чтобы избежать вероятности недооценки объема памяти, необходимого для эффективной работы сервера. Если есть возможность позволить себе покупку сервера с большим объемом оперативной памяти, обязательно сделайте это.

2.2.2. Диски

Существует два основных типа жестких дисков — IDE и SCSI. Интерфейс IDE самый распространенный и недорогой. Диски с этим интерфейсом установлены на большинстве функционирующих персональных компьютеров. Операции чтения/записи на них управляются центральным процессором, что является основной причиной их относительно низкой производительности. Это особенно сильно проявляется на многодисковых системах.

Для серьезных серверов единственно правильным будет аппаратное решение с интерфейсом SCSI. Интерфейс SCSI (произносится как "скази") намного "умнее" интерфейса IDE. В частности, он и его контроллер берут на себя выполнение всех рудиментарных задач поиска (операции чтения диска, операции позиционирования), не обращаясь при этом для выполнения каждой команды к центральному процессору. В результате диски с этим интерфейсом работают значительно быстрее, чем с интерфейсом IDE, но стоимость их естественно, значительно дороже.

Рекомендуется устанавливать на сервер как минимум два диска SCSI. Один — под операционную систему, другой — под сервер Apache. Это позволяет осуществлять доступ к файлам сервера и файлам операционной системы одновременно.

В обшем, лучше установить несколько дисков поменьше, чем один или два больших. Независимо от типа интерфейса отдельный диск может осуществлять только одну операцию чтения/записи за один период времени. Распределение операций чтения/записи между несколькими дисками позволит увеличить производительность системы в целом.

В зависимости от вашего отношения ко времени простоя сервера можно добавить еще два диска и попробовать построить массив RAID (Redundant Array of Inexpensive Disks — избыточный массив недорогих дисков). Конфигурация RAID 2, или зазеркаливание, создает и поддерживает две одинаковые копии данных на двух различных дисках. Копией можно воспользоваться в случае сбоя в работе одного из дисков. Конфигурация RAID 5 размешает копии данных на трех дисках таким образом, что сбой в работе одного из них никак не может разрушить данные. К слову, далеко не все операционные системы поддерживают работу массивов RAID.

[1] В пределах возможностей конфигурации (см. главы 3 и 10)

2.3. Подготовка системы

Во-первых, следует сделать выбор системной платформы. Для большинства клонов ОС Unix и Linux можно получить скомпилированные двоичные коды. ОС Unix является "родной" средой сервера Apache, и он должен лучше всего работать на Unix-платформах. Однако имеются достаточно надежные версии сервера, работающие с ОС Windows 95, Windows 98, Windows 2000, и Windows NT, а совсем недавно и у Apple появилась своя версия сервера.

Обычно пользователи останавливают выбор на какой-то одной операционной системе. Ибо если пользователь имеет опыт работы с одной ОС, он всегда предпочтет ее любой другой операционной системе. Если же вы одинаково хорошо (или плохо) знакомы со всеми тремя вышеперечисленными операционными системами, то лучше выбрать платформу Unix. Я не хочу этим сказать, что версии под управлением платформ Win32, работают недостаточно надежно. Но сказывается тот факт, что сервер изначально был создан для работы под управлением ОС Unix. Беглый взгляд на конфигурационные файлы обнаруживает Unix-происхождение сервера (множество порожденных процессов, ограничители "/" в спецификациях каталогов).

Кроме того, новые доработки производятся на версиях, работающих под ОС Unix, и только после этого переносятся в другие операционные среды. Многие из возможностей первоначально создавались для работы под Unix и только потом использовались для работы под ОС Windows и ОС Mac. И, наконец, ОС Unix - самая дешевая из трех операционных систем. Во время написания этой книги цена ОС OS X и Windows 2000 составляла от 750 до 1000 долларов, а ОС Linux (Unix для персональных компьютеров) можно получить совершенно бесплатно.

2.3.1. Подготовка ОС Unix

Конечно же для сервера Apache потребуется определенное место на диске, будь это отдельная файловая система или просто каталог в уже существующей файловой системе. Но это должно быть место, отведенное только для него. С этого и следует начать.

Придумывать идентификаторы пользователя и групп пользователей для процессов, создаваемых сервером Apache, совсем не обязательно. Программа отлично работает под стандартным идентификатором (например nobody). Но все же это рекомендуется сделать. Специфический, присущий только этому серверу идентификатор пользователя увеличивает его безопасность и устойчивость, а также создает впечатление, что вы знаете, что делаете.

2.3.2. Корневой каталог Unix

Пользователь Apache должен иметь дисковое пространство (сейчас мы его подготовим), предназначенное только для него. Для этого с помощью команды mkdir создадим подкаталог в одном из уже существующих каталогов (например /usr/local, /opt).

```
cd /opt; mkdir apache
```

Вероятно, потребуется создать и новую файловую систему. Все зависит от максимальных объемов планируемых Web-услуг. Для этого необходимо выделить дисковое пространство (в ОС Linux воспользуемся командой cfdisk), затем создать файловую систему (с помощью команды mkfs) и смонтировать ее командой mount.

Для создания подкаталогов в корневом каталоге Apache, указанных в табл. 2.1, воспользуемся командой mkdir.

Таблица 21. Подкаталоги пользователя Apache

Подкаталог	*Назначение*
bin	Содержит исполняемый файл сервера Apache (httpd) и все другие программы.
logs	Содержит все регистрационные файлы сервера Apache.
conf	Содержит все конфигурационные файлы сервера Apache.

Программы Apache должны работать под управлением пользователя root. По этой причине нельзя никому предоставлять права записи и владения в вышеперечисленных каталогах. Это создаст брешь в системе безопасности. Обезопасить каталоги можно с помощью команд chown, chgrp и chmod.

2.3.3. Идентификаторы пользователей и групп пользователей Unix

Команды, предназначенные для создания новых пользователей и групп пользователей, варьируются в разных клонах ОС Unix. Команды, приведенные в следующих примерах, рассчитаны на ОС Linux. Если вы работаете не с ОС Linux, новых пользователей можно создать, изменив содержимое файлов /etc/passwd и /etc/group. Вот формат файла /etc/group, структура которого приведена в табл. 2.2:

```
group:password:GID:user_list
```

Таблица 2.2. Элементы файла /etc/group

Поле	*Назначение*
group	Наименование группы (рекомендуется Apache).
password	Рекомендуется оставлять пустым.
GID	Уникальный числовой идентификатор группы.
user_list	Перечень членов группы, разделенные запятыми, имена должны соответствовать именам пользователей из файла /etc/passwd file.

Вот формат файла /etc/passwd, структура которого приведена в табл. 2.3:

```
user:encrypted password:UID:GID:user name:home directory:shell
```

Таблица 2.3. Элементы записи пользователя Apache изфайла /etc/passwd

Поле	***Назначение***
user	Имя пользователя (вероятно Apache).
encrypted_password	В целях безопасности нужно задать звездочку. Это означает, что пароль отсутствует, но никто не может зарегистрироваться в качестве пользователя Apache. Если же возникла такая необходимость, сначала следует зарегистрироваться как пользователь root, а затем заменить активного пользователя с помощью команды su - Apache. Таким образом можно обойти парольную защиту.

Окончание табл. 2.3

Поле	*Назначение*
UID	Уникальное числовое значение, необходимое системе для обозначения всех файлов и каталогов, которыми владеет пользователь Apache.
GID	Уникальный числовой идентификатор группы, аналогичный идентификатору, имеющемуся в файле /etc/group.
user_name	Реальное имя пользователя, например, Sue Smith или Bob Wilson. В нашем случае это Apache.
home_directory	Каталог или файловая система, созданная специально в качестве корневого каталога сервера Apache (/opt/apache для вышеприведенного примера).
shell	/bin/false

Нужно быть особенно осторожным при корректировке и добавлении значений в файлах /etc/passwd и /etc/group, тщательно соблюдать последовательность столбцов даже если они пустые.

Кроме того, для внесения изменений в эти важные системные файлы можно обратиться к услугам специальной программы или сценария. Работая в ОС Linux, введите команду:

```
groupadd apache
```

Затем создайте пользователя по имени Apache, входящего в группу Apache:

```
useradd -g apache apache
```

Теперь убедитесь в том, что владельцем всех ранее созданных подкаталогов является пользователь root:

```
chown 0:0 -R /opt/apache
```

2.3.4. Подготовка ОС Windows

Инсталляция сервера, работающего под управлением ОС Windows, не требует никакой специальной подготовки, так как специальных мероприятий по обеспечению безопасности проводить не требуется.

2.3.5. Подготовка ОС Mac OS X

ОС Mac OS X поставляется с уже установленным сервером Apache.

2.4. Как получить Web-сервер Apache

Web-сервер Apache можно получить совершенно бесплатно как в откомпилированных двоичных кодах, так и в исходных текстах. На CD-ROM, приложенном к данной книге, можно найти самые последние версии (на момент издания этой книги) как того, так и другого. Возможно, когда вы будете читать эти строки, у вас в распоряжении уже появится все необходимое. Если содержимое CD-ROM вас удовлетворяет, то щелкните на пиктограмме CD-ROM (для ОС Mac и Windows) или смонтируйте устройство (в ОС Unix) и скопируйте все, что необходимо.

В противном случае можно обратиться к первоисточнику. Последнюю версию Web-сервера Apache для любой операционной системы всегда можно найти по адресу http://www.apache.org/dist. Исходные тексты (которые затем следует откомпилиро-

вать) тоже можно найти в каталоге /dist. Откомпилированные двоичные коды для различных платформ размещаются в каталоге в /dist/binaries. При копировании двоичных кодов следует обязательно убедиться в том, что были выбраны коды, соответствующие вашей платформе (например Win32 для Windows 95/98/NT, Solaris для ОС Solaris).

Для упрощения и ускорения передачи дистрибутивы исходных текстов объединены в один загружаемый файл, сжатый с помощью программы-архиватора. На ОС Unix необходимо разархивировать и распаковать дистрибутив перед его использованием. Дистрибутивы в ОС Win32 являются саморазархивирующимися файлами, и все, что требуется — это только дважды щелкнуть по ним.

2.4.1. Unix

Первое, что нужно сделать — скопировать загружаемый файл в файловую систему, имеющую достаточно места для разархивирования и распаковки файла (сейчас это около 1 Мбайта).

```
mv apache_X.Y.Z.tar.Z /
```

Затем разархивировать файлы.

```
tar -xvzf apache_X.Y.Z.tar.gz
```

Наконец, скопировать их в предварительно созданную файловую систему.

```
mv apache_X.Y.Z /opt/apache
```

2.4.2. ОС Windows

В случае с ОС Windows NT необходимо убедиться, что установлен Service Pack 3 или выше. Инсталляция на всех платформах Win32 заключается только в двойном щелчке по загружаемому файлу.

2.5. Компиляция Apache

Компиляцией называется процесс преобразования понятного для человека исходного кода в исполняемый компьютером машинный код. Чтобы извлечь пользу из загруженного исходного текста, его необходимо откомпилировать и поместить туда, где система сможет его найти и запустить.

2.5.1. Дополнительная информация: команда make

- Unix-файлы почти всегда компилируются с помощью популярной утилиты make. Эта утилита имеет двойственное предназначение: исходный код и библиотеки собираются в единое целое. В случае с сервером Apache, они могут быть получены с Web-узла или (хвала Господу) уже находятся на вашей машине.
- Набор правил, необходимых для сборки исходных текстов. Эти правила приобретают форму текстового файла Makefile. Он определяет такие аспекты, как компилятор, файлы с исходным текстом и их взаимосвязи, поставленную конечную цель (откомпилированный двоичный исполняемый файл, чистый каталог или установленный исполняемый файл).

В процессе создания и конфигурирования сервера Apache время от времени мы будем просматривать Makefiles. В зависимости от того, какой тип сервера будет построен, его, вероятно, понадобится редактировать вручную. Подробнее с характеристиками утилиты make можно ознакомиться в системной документации.

2.5.2. Сценарий APACI

Прежде процесс компиляции рабочей версии сервера Apache заключался в модификации конфигурационных файлов, находящихся в каталоге src, с тем, чтобы задать модули, включаемые в двоичный код. Всовременных дистрибутивах этого не требуется, по крайней мере без этого вполне можно обойтись. Сейчас появилась возможность использовать конфигурационный сценарий Apaci для этих целей. Он позволяет задавать перечень модулей, которые будут включены в двоичный файл httpd, прямо из командной строки.

Конечно, если очень хочется, никто не может запретить вам по-старому корректировать файлы, находящихся в каталоге src. Безусловно, этот метод позволяет более полноценно управлять процедурой компиляции. Такие ситуации возникают тогда, когда компилируются модули сторонних поставщиков, и это приходится делать независимо от вашего желания. Дискуссия на тему редактирования конфигурационных сценариев будет продолжена в главе 12, "Состав модуля". А пока сфокусируем наше внимание на работе с конфигурационным сценарием Apaci как на простейшем методе.

Поменяйте рабочий каталог на каталог, в котором находится только что распакованный дистрибутив Apache. Там можно найти файл сценария configure. Начнем со стандартной инсталляции, значит сценарию configure будет передан только один параметр --prefix, содержащий имя каталога, в котором хранится вся информация о сервере Apache. В нашем примере таким каталогом является /opt/apache. Вызываем конфигурационный сценарий, полагая, что будет использоваться этот же каталог:

```
./configure -prefix = /opt/apache
```

Получаем следующий ответ:

```
Configuring for Apache, Version 1.3.12
+ using installation path layout: Apache (config.layout)
        Creating Makefile
Creating Configuration.apaci in src Creating Makefile in src
+ configured for Linux platform
+ setting C compiler to gcc
+ setting C pre-processor to gcc -E
+ checking for system header files
+ adding selected modules
+ checking size of various data types
+ doing sanity check on compiler and options
Creating Makefile in src/support
Creating Makefile in src/regex
Creating Makefile in src/os/unix
Creating Makefile in src/ap
Creating Makefile in src/main
```

В результате будет создан файл Makefile. Выше уже указывалось, что этот файл сообщает системной утилите make, каким образом сочетаются загруженный исходный код с системной информацией и утилитами при построении исполняемого двоичного кода. Чтобы запустить этот процесс, введите:

```
make
```

Если все пройдет нормально, это закончится появлением рабочей исполняемой версии httpd в этом каталоге[2]. Чтобы установить его в системе, введите команду:

```
make install
```

На экране появится сообщение:

```
You now have successfully built and installed the
Apache 1.3 HTTP server. To verify that Apache
actually works correctly you now should first check
the (initially created or preserved) configuration
files
/opt/apache/conf/httpd.conf
and then you should be able to immediately fire up
Apache the first time by running:
/opt/apache/bin/apachectl start
Thanks for using Apache.                The Apache Group
                                          http://www.apache.org/


Сейчас вы успешно построили и установили HTTP сервер
Apache 1.3. Для проверки его работоспособности
сначала проверьте (первоначально созданные или
полученные) конфигурационные файлы
/opt/apache/conf/httpd.conf
после этого можно запустить сервер Apache с помощью
команды:
/opt/apache/bin/apachectl start
Спасибо за использование Apache,      Группа Apache
                                          http://www.apache.org/
```

Наши поздравления. Теперь самое сложное позади. По умолчанию только что созданный файл **httpd** содержит модули, указанные в табл. 2.4.

Таблица 2.4. Стандартные модули файла httpd

Модуль	*Описание*
http_core.c	Основные функциональные возможности сервера.
mod_env.c	Обеспечивает передачу переменных окружения CGI-сценариям.
mod_log_config.c	Регистрация с возможностью конфигурирования пользователя.
mod_mime.c	Обеспечивает определение типов документов по расширениям.
mod_negotiation.c	Согласование содержимого.
mod_status.c	Отображает информацию о состоянии сервера в виде Web-страницы.

[2] Компилятор и библиотеки, имеющиеся на вашем компьютере, могут не соответствовать тем, которые были использованы при создании файла makefile. В таком случае при попытке компиляции это вызовет ошибку (или целую их последовательность). Тогда может потребоваться модифицировать файл makefile, или, возможно, некоторые файлы системной библиотеки. Этот процесс называется нетривиальным. В качестве средства поиска файлов в системе предлагается конструкция find / -type f -print | grep <missing file>, гдечасть <missing file> можно заменить на имя отсутствующего файла.

Окончание табл. 2.4

Модуль	*Описание*
mod_include.c	Позволяет создавать ограниченное динамическое содержание.
mod_autoindex.c	Автоматическое отображение каталогов.
mod_dir.c	Основное отображение информации о каталогах.
mod_cgi.c	Позволяет генерирование динамического содержимого.
mod_asis.c	Обеспечивает передачу файлов без HTTP-заголовков.
mod_imap.c	Обработка файлов распределения изображений.
mod_actions.c	Позволяет использовать CGI-сценарии для обработки определенных типов файлов.
mod_userdir.c	Позволяет обрабатывать страницы из пользовательских корневых каталогов.
mod_alias.c	Переопределение URL и перемещение в пределах одной файловой системы.
mod_access.c	Управление доступом.
mod_auth.c	Ограничение идентификации.
mod_setenvif.c	Обеспечивает возможность устанавливать переменные окружения сервера Apache на основании информации о пользователе.

Если такая конфигурация Вам не подходит, можно воспользоваться сценарием configure и указать в нем, какие модули оставить, а какие — убрать. Сценарий configure содержит много опций, и для того, чтобы получить их полный список, введите команду configure -h. Среди наиболее популярных опций можно назвать --enable-module И --disable-module. Например, чтобы создать httpd, который содержит все перечисленные выше модули плюс модуль mod_proxy, необходимо ввести команду:

```
./configure -prefix = /opt/apache \
--enable-module = proxy
```

А потом вызвать команду make и переустановить полученный сервер. Или, воспользовавшись опцией -disable, можно удалить стандартные модули. Вот, например, последовательность команд, удаляющая из состава сервера стандартный модуль mod_asis:

```
./configure --prefix=/opt/apache \
--disable-module=asis
```

Но каждый раз, когда в сценарий configure вносятся изменения, чтобы они возымели действие, сервер нужно перекомпилировать. После того как будет получен желаемый результат, приступаем к финальному шагу.

2.6. Инсталляция Web-сервера Apache

2.6.1. ОС Unix

Сервер Apache может работать в любом каталоге. Однако из соображений безопасности неплохо было бы скопировать скомпилированную (или загруженную) программу в безопасный каталог.

```
cp /opt/apache/src/httpd /usr/sbin
```

В таком случае в сценарий apachectl следует внести изменения. Сценарий можно найти в подкаталоге bin инсталляционного каталога. Измените с помощью любого редактора переменную HTTPD в сценарии. Переменная теперь хранит путь к новому каталогу, в котором будет находиться httpd.

```
HTTPD=/usr/sbin/httpd
```

2.6.2. ОС Windows

Процедура инсталляции в ОС Windows NT полностью автоматизирована. Дистрибутив, который можно получить с Web-узла www.apache.org, является саморазворачивающимся[3]. Для этого достаточно дважды щелкнуть по загружаемому файлу (см. рис. 2.1). Перед этим будет задано несколько вопросов, касающихся правил лицензирования и выбора каталога, в который устанавливается сервер Apache. По умолчанию выбирается каталог C:\Program Files\Apache Group\Apache. Если вы собираетесь запустить сервер Apache как службу Windows NT, этот путь необходимо изменить на C:\Apache. Если нет, можно оставить стандартный путь.

Рис. 2.1. Двоичный дистрибутив для ОС Windows

За исключением выбора корневого каталога и в зависимости от того, нужен ли исходный код вообще, весь остальной процесс инсталляции проходит более или менее автоматически. Теперь нажмите клавишу Finish. Все,инсталляция завершена.

При работе в ОС Windows NT существует возможность установки сервера Apache в качестве службы. Этот вариант предпочтительнее, ибо в таком случае сервер запускается автоматически во время загрузки операционной системы и продолжает работать даже после того, как пользователь, его вызвавший, закончит работу и выйдет из системы. Для этого необходимо выбрать опцию *Install Apache as Service* в каталоге Apache меню Start. Или ввести команду:

```
apache -i
```

Тогда программа Apache будет установлена как служба Apache. В версиях для Windows NT, более поздних, чем 1.3.7, имеется возможность менять имена служб. Более того, версия Apache для Windows NT позволяет задавать конфигурационный файл, используемый во время конфигурации.

```
apache -i -n "httpd" -f "\\Odin\Apache\conf\httpd.conf"
```

По этой команде в соответствии с информацией, содержащейся в файле \\Odin\Apache\conf\httpd.conf, устанавливается служба httpd.

[3] Двоичные инсталляции предпочтительней. Причина этого кроется в возможности разрешения многих проблем при помощи системного реестра.

Работа Web-сервера Apache в среде ОС Windows NT в качестве службы считается более предпочтительной. Утилита управления работой служб в ОС Windows NT доступна по команде Start->Settings->Control Panel->Services (см. рис. 2.2).

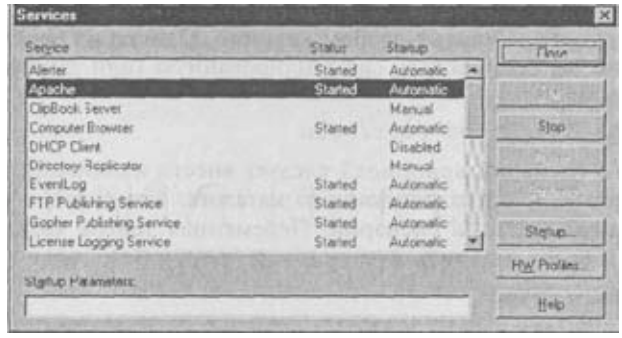

Рис. 2.2. Список служб в ОС Windows NT

Из этого окна запускается служба Apache. Доступ к окну служб можно получить, щелкнув на кнопке Startup. Затем выбирается метод запуска сервера. Существует три варианта запуска:

- Автоматический запуск. Сервер Apache запускается автоматически во время загрузки системы.
- Ручной запуск. Сервер Apache запускается и останавливается вручную.
- Отключено. Работающий экземпляр сервера Apache останавливается.

2.6.3. ОС Mac OS X

ОС Mac OS X поставляется с уже установленным портом Web-сервера Apache.

2.7. Заключение

На этом этапе все основные элементы сервера можно считать установленными. Но до реального запуска сервера Apache остается провести еще небольшую подготовительную работу. Детальную информацию о конфигурировании сервера вы найдете в главе 3, "Конфигурирование Web-сервера Apache", а с тем, как следует управлять им, можно ознакомиться в главе 4, "Запуск, перезапуск и остановка".

Глава

3

КОНФИГУРИРОВАНИЕ WEB-СЕРВЕРА APACHE

В этой главе…

3.1. Введение

Сервер Apache может получать информацию о настройках из четырех источников: конфигурационных файлов httpd.conf, access.conf, srm.conf и mime.types[1]. Изначальный замысел заключался в том, чтобы разделить директивы по функциям (функции защиты и контроля доступа возложить на файл acc ess.conf, управление ресурсами сервера — на файл srm.conf). Но на практике оказалось, что это не совсем удачная идея. Для улучшения читабельности директив рекомендуется хранить их всех в файле httpd.conf.

Подсказка

Не спешите начинать работу. Рекомендуем сначала удалить расширения .dist стандартных файлов, изменить пути к каталогам в файлах для приведения в соответствие всех значений и только после этого — запустить сервер. Перечисленные операции либо улучшат, либо ухудшат рабочие характеристики. Однако я настоятельно рекомендую закрыть доступ к серверу, пока вы не внесли все изменения в конфигурационные файлы и не протестировали работу сервера.

Вам предстоит немного настроить стандартный дистрибутив для его оптимальной работы. В этой главе описана буквально каждая строка конфигурационного файла, возможные значения, присваиваемые директивам, директивы, которые применяются в стандартных конфигурационных файлах, за исключением директив Proxy и virtual-Host (их очередь наступит в следующих главах). Цель данной главы — обеспечить читателя информацией, достаточной для принятия обоснованных решений при настройке сервера с использованием материалов этой книги.

[1]У модуля mod_perl, как и удругих модулей, могут быть свои сценарии загрузки.

3.2. Модули

Стандартный дистрибутив сервера Apache использует стандартный набор модулей и не требует применения директив AddModule и ClearModuleList, но автор все равно поместил информацию по этим директивам, поскольку они важны, особенно, если круг интересов читателя затрагивает электронную коммерцию. Более подробную информацию по этим директивам можно найти в главе 12, "Состав модуля".

3.2.1. Объединенный конфигурационный файл

Несмотря на то, что стандартное решение подразумевает использование трех раздельных конфигурационных файлов, абсолютно очевидно, что это создает больше проблем, чем удобств. Значительно проще хранить все конфигурационные директивы в одном файле — httpd.conf. К счастью, это несложно. Чтобы объединить все конфигурационные файлы в один, добавьте директивы AccessConfig и ResourceConfig в файл httpd.conf. Для сервера Apache это не будет ошибкой.

Разделение или объединение конфигурационных файлов не влияет на производительность. Здесь можно поступать, как вам удобно.

3.3. Файл httpd.conf

Файл httpd.conf — главный конфигурационный файл сервера, в нем задается большинство основных моментов спецификации web-страницы: имя сервера, сетевые установки, виртуальный хостинг и т.д.

3.3.1. Директива ServerType

Эта директива принимает только два значения: либо inetd, либо standalone. Чтобы сделать разумный выбор, нужно иметь представление о процессах ОС Unix. Процесс — программа, функционирующая в среде ОС Unix, созданная для работы в фоновом режиме (то есть, не подключенная к определенному терминалу). Процессы обеспечивают набор сервисов без вмешательства пользователя. В качестве примера такого процесса можно назвать сервис telnetd, который обеспечивает сеансы telnet или процесс регистрации системных сообщений syslogd.

Существует много программ, которые могут выполнять функции процессов. Впрочем, если они все будут вызваны одновременно, то даже самая мощная система ощутимо снизит производительность. Эта проблема была решена одним из светлых умов задолго до появления Internet.

Преобладающее большинство, если не все процессы, запущенные на вашей системе, полезны вам только время от времени, и лишь некоторые из них работают постоянно. В основном, они, однажды вызванные, просто "сидят" в системе, потребляя системные ресурсы и затормаживая работу, находясь в состоянии ожидания заданий. Это неразумно. Для решения проблемы была создана специальная программа, единственной задачей которой является выяснение наличия запросов системы к различным процессам, и запуск последних по мере надобности. Эта программа, как вы уже догадались, называется inetd.

После воспевания неоспоримых добродетелей inetd и достижений прошлого мне совестно признать, что все же *не стоит* работать с Apache таким образом. Проблема в том, что запуск и остановка программы дает нагрузку системе. В основном это происходит при чтении кода программы с винчестера и записи его в оперативную память. Доступ к диску требует определенного времени, и, как вам подтвердит любой, кто пытался зайти на Web-сайт при помощи модема со скоростью 28 Кбит, эти незначительные задержки имеют свойство быстро аккумулироваться.

Добиться большей производительности сервера Apache можно при работе в режиме standalone. Однако вам придется поступиться механизмами защиты, которые обеспечивает inetd. Если нет острой необходимости в защитных механизмах inetd (в частности, для получения дополнительной информации по этой теме загляните в документацию системы Unix по упаковщику tcpd), имеет смысл настроить сервер Apache на работу в режиме standalone.

```
ServerType standalone
```

3.3.2. Спецификация TCP-портов: директива Port

IP-соединение определяется двумя параметрами: IP-адресом сервера, к которому вы подключаетесь (например 192.168.1.10), и номером порта, который используется сервером для наблюдения за подключениями. На компьютере с ОС Unix порты соотносятся с именами сервисов в текстовом файле /etc/services. Очевидно, что для успешного соединения двух компьютеров необходимо согласовать, посредством каких портов будут переданы те или иные данные. В основном, номера портов уже давно четко определены и стандартизированы, сейчас их называют не иначе как "хорошо известные номера портов"[2]. Например порт соединений типа клиент-сервер (http или hypertext transfer protocol) имеет стандартный номер 80.

```
Port 80
```

Существует немного ситуаций, которые могут заставить изменить этот номер. Если у вас имеется локальная сеть, где осуществляется полный персональный контроль за всеми броузерами клиентов, вы можете посылать вашу информацию посредством любого порта. Proxy-серверы часто обслуживают запросы по нестандартным портам.

Однако, если кроме порта 80 вы планируете использовать какой-нибудь другой порт, то броузеры клиентов должны быть настроены соответствующим образом (см. главу 6, "Proxy-серверы и кэширование"). Необходимо, чтобы клиент и сервер были настроены на один и тот же номер порта, иначе соединения не произойдет. Поэтому сервер, предназначенный для использования широким кругом пользователей, должен использовать порт 80.

Кроме того, если коммуникация будет происходить строго в соответствии с протоколом Secure Socket Layer (SSL), вам потребуется включить TCP-порт номер 443.

```
Port 443
```

Протокол SSL описан в главе 8, "Безопасность".

Обратите также внимание на ограничение ОС Unix, в соответствии с которым все порты с номером до 1023 работают под управлением пользователя root, что делает их недоступными для остальных пользователей. Поэтому вы должны будете вызывать файл httpd.conf как пользователь root независимо от идентификатора пользователя, под которыми работают серверы клиента.

[2] Это кажется мне несколько неразумным. Более подходящим названием было бы "номера портов промышленного стандарта" или "нерушимо установленные номера портов". С другой стороны, это понятие было введено теми же людьми, которые решили собрать все действительно полезные протоколы, составляющие основу современных телекоммуникаций, и в шутку их назвать "Request For Comment" или RFC. Существует RFC для стандарта TCP (RFC 793), а также для telnet (RFC 854). При желании вы можете достать копию, перечитать ее и предложить изменения. Если вы придумаете что-то дельное, ваше предложение, возможно, даже будет принято к сведению,— большинство RFC не изменялись десятилетиями, но это возможно. В то же время, головокружительное количество "самозванных" промышленных стандартов буквально каждый день снова и снова предлагается компаниями-разработчиками ПО. Отсюда мораль: в компьютерной индустрии, как, впрочем, и в других начинаниях, самореклама вызывает подозрение (например, если что-либо называет себя "Точным руководством", то оно, вероятнее всего, таковым не является).

2.3.3. Директива ClearModuleList

Серверы Apache поставляются со стандартным набором активных модулей. Чтобы изменить этот набор, активизировав с помощью директивы AddModule лишь указанные вами модули, добавьте в конфигурационный файл директиву ClearModuleList.

```
ClearModuleList
```

3.3.4. Директива AddModule

Чтобы использовать готовый, но неактивный в данный момент модуль, используйте директиву AddModule. Например, чтобы активизировать модуль mod_auth_dbm, добавьте директиву:

```
AddModule    mod_auth_dbm.c
```

3.3.5. Читабельность против производительности: директива HostnameLookups

При регистрации пользователей, обращающихся к вашему серверу, можно ввести цифровой IP-адрес (например, 204.62.129.132), либо буквенное имя узла, связанное с этим адресом (например, *\www.apache.org)*. Естественно, буквенные имена сделают регистрационные файлы более читабельными. Однако, проверка в доменной системе имен имени каждого пользователя, обращающегося к вашему узлу, гарантирует вам значительное замедление производительности. Поэтому при отсутствии настоятельной необходимости в таких именах директиву HostnameLookups рекомендуется отключить.

```
HostnameLookups  off
```

3.3.6. Взаимодействие с системой защиты ОС Unix: директивы User И Group

Как говорилось в главе 2, "Инсталляция Web-сервера Apache", рекомендуется создать системного пользователя Unix специально для обеспечения работы сервера Apache. Это помогает избежать возникновения большого количества потенциальных проблем, которые имеют место при работе сервера Apache под управлением пользователя root.

В приведенном ниже примере пользователь обозначен читаемым буквенным именем (например apache), а группа обозначена номером, связанным с группой Apache. Для читабельности лучше использовать буквенные имена, хотя это не столь существенно.

```
User apache
Group 506
```

3.3.7. Согласование выводимой информации с типом броузера: директива BrowserMatch

Используйте упомянутую директиву для настройки переменных окружения, основанных на типе броузера клиента, который пытается получить доступ к вашему серверу, как показано в заголовке User-Agent запроса HTTP. Это необходимо для того, чтобы обеспечить механизм, позволяющий адаптировать узел к особенностям броузеров. Например, есть возможности, отлично зарекомендовавшие себя в броузерах Netscape, но не работающие в броузерах Internet Explorer. Эта директива позволяет настроить переменные окружения, которые будут использовать сценарии CGI для настроек вывода, основанных на типах броузеров, получающих к ним доступ. Например следующая директива устанавливает значение nokeepalive, когда клиент использует броузер Netscape 2.x:

```
BrowserMatch Mozilla/2 nokeepalive
```

3.3.8. Настройка контактного адреса: директива ServerAdmin

Это — адрес электронной почты, на который пользователи будут отсылать почту в случае возникновения проблем. Он будетотображаться в некоторых диагностических сообщениях.

```
ServerAdmin you@yousite.org
```

3.3.9. Корневой каталог сервера: директива ServerRoot

Директива ServerRoot задает имя корневого каталога, т.е. каталога, который станет корневым для сервера. Директива Server Root является ключевой, потому что большинство директив в качестве параметров принимают имена файлов или каталогов, и это позволяет определять в директивах либо абсолютные пути, либо путь относительно каталога ServerRoot. Заканчивая пример, начатый в главе 2, "Инсталляция Web-сервера Apache", можно определить директиву ServerRoot следующим образом:

```
ServerRoot /opt/apache
```

3.3.10. Выбор IP-адреса: директива BindAddress

Используйте эту директиву, чтобы сообщить серверу Apache IP-адрес на локальном компьютере. Если у вас имеется на этой машине только один сетевой интерфейс, то к упомянутой директиве прибегать не стоит. По умолчанию сервер Apache прослушивает все соединения.

Практически директива срабатывает лишь тогда, когда у вас несколько сетевых интерфейсов[3]. Например, чтобы сервер Apache поддерживал только один IP-адрес 192.168.1.10, используйте следующую директиву:

```
BindAddress 192.168.1.10
```

Подсказка

Чтобы сервер Apache игнорировал запросы, исходящие не от локального компьютера во время загрузки и запуска системы, вы можете в качестве вашего BindAdress задать loopback-адрес: BindAddress 127.0.0.1.

Значение BindAddress может быть задано звездочкой "*", обозначающей все сетевые интерфейсы на локальном компьютере как актуальные IP-адреса, либо как полное доменное имя. Использование доменного имени в конфигурационных файлах требует просмотра этого имени в базе DNS, что отрицательно сказывается на производительности. Для отображения всех возможных IP-соединений на вашем компьютере (стандартное поведение) используйте следующую директиву:

```
BindAddress *
```

При необходимости прослушивать два или более IP-адреса, но не все возможные на вашем компьютере адреса, вам потребуется директива Listen. Директива BindAddress не может использоваться в одном конфигурационном файле несколько раз.

[3] Для отображения имеющихся сетевых интерфейсов в Unix-системах воспользуйтесь командой netstat -i. Для ОС Windows зайдите в Пуск=>Настройка=>Панель Управления=>Сеть.

3.3.11. Определение файла регистрации сообщений об ошибках: директива ErrorLog

Эта директива указывает путь к регистрационному файлу, в который сервер Apache будет записывать диагностические сообщения об ошибках. Путь может быть как абсолютный, так и относительный. Если указанный путь не начинается с символа косой черты "/", задающей абсолютный путь из корневого каталога файловой системы[4], то будет рассмотрен путь относительно каталога ServerRoot. Например, предполагая, что ваш каталог ServerRoot — /opt/apache, директива

```
ErrorLog logs/error_log
```

показывает, что файл регистрации ошибок находится в /opt/ apache/logs/error_log. С другой стороны, директива

```
ErrorLog /var/logs/apache
```

приведет к тому, что все сообщения будут записываться в файл /var/logs/apache. Убедитесь, что пользователь, под управлением которого работает сервер, имеет право записи в файл регистрации сообщений об ошибках.

3.3.12. Определение файла записи данных об обмене данными: директива TransferLog

Эта директива задает файл, в котором будут храниться записи об обмене данными сервера с внешним миром. Указанный путь может быть как абсолютным, так и относительным.

```
TransferLog logs/access_log
```

3.3.13. Идентификатор процесса-родителя: директива PidFile

Эта директива указывает файл, в котором хранятся идентификаторы процессов ОС Unix. Идентификатор процесса — это уникальный номер, который использует ядро Unix для отслеживания процесса. Благодаря системному номеру процесса Apache (httpd), который был запущен в первую очередь, система сможет отключить его и его клоны. Путь к этому файлу может быть как абсолютным, так и относительным.

```
PidFile logs/httpd.pid
```

3.3.14. Обмен между процессами: директива ScoreBoardFile

Чтобы Apache мог обмениваться со своими клонами (порожденными процессами), которые создаются и удаляются в процессе работы, нужен некоторый объем оперативной памяти, отведенный только для этой операции. Обычно (и это предпочтительней) такая область находится где-то в оперативной памяти и потому целиком прозрачна для администратора. Однако некоторые Unix-системы требуют для этих целей отдельный файл. В таких случаях используйте директиву ScoreBoardFile. Путь к файлу может быть как абсолютным, так и относительным.

```
ScoreBoardFile logs/apache_status
```

[4] Это касается реализаций сервера как в ОС Windows, так и в Unix. Несмотря на то, что в качестве ограничителя в ОС Windows в спецификации пути используется символ обратной косой черты "\", в конфигурационных файлах используются ограничители, соответствующие стандарту Unix "/".

3.3.15. Имя сервера: директива ServerName

Используйте эту директиву для задания имени сервера, которое будет отсылаться клиентам вместо имени компьютера. Самый распространенный вид замены — это замена имени на www. Замеъте, чтобы пользователи могли получить доступ к вашему серверу, имя, указанное директивой ServerNam e, должно быть известно DNS-серверам.

```
ServerName www.example.com
```

3.3.16. Директива CacheNegotiatedDocs

Стандарт HTTP 1.1 определяет группу заголовков, которые сервер Apache отсылает клиентам и proxy-серверам (по крайней мере тем, которые обращают внимание на протоколы HTTP 1.1), чтобы дать им знать, какое содержание можно помещать в кэш, а какое — нет. По умолчанию все содержимое помечено как недоступное для кэширования. Эту директиву можно использовать, чтобы изменить стандартные установки и позволить proxy-серверам кэшировать такие файлы.

```
CacheNegotiatedDocs
```

3.3.17. Ограничение неактивных соединений по времени: директива Timeout

Эта директива указывает количество секунд, на протяжении которых данное соединение может быть неактивно, до того момента, как Apache завершит его. Определение неактивности достаточно приблизительное. Можно сказать, что это число секунд с того момента времени, как состоялось одно из событий:

- было установлено соединение или был получен запрос GET
- в случае неполных передач — с тех пор, как было получено последнее подтверждение получения
- пакет данных был получен по HTTP-запросам PUSH или PUT .

Заметьте, что 300 секунд, устанавливаемых по умолчанию — это ужасно много. Можно смело сократить это время, закрывая неработающие соединения (хотя бы самые медленные), что увеличит производительность. Например:

```
Timeout 150
```

3.3.18. Разрешение устойчивых соединений: директива KeepAlive

Эта директива дает возможность поддерживать устойчивые соединения. Устойчивое соединение позволяет клиенту запросить более одного блока данных одновременно. *Включение этой опции полезно всем,* поскольку она устраняет загрузку, которая требуется в процессе инициализации и завершения IP-соединений для всех запросов, кроме первых и последних, способствуя, таким образом, *увеличению общей производительности.*

```
KeepAlive On
```

3.3.19. Директива MaxKeepAliveRequests

Эта директива задает максимальное число запросов, разрешенных во время одного устойчивого соединения (см. описание директивы KeepAlive). Для увеличения производительности этот номер должен быть достаточно большим. Значение 0 означает "неограниченно".

```
MaxKeepAliveRequests 0
```

3.3.20. Директива KeepAliveTimeout

Используйте эту директиву для задания количества секунд, на протяжении которых во время устойчивого соединения сервер Apache будет ожидать следующего запроса.

```
KeepAliveTimeout 15
```

3.3.21. Увеличение производительности: директива MinSpareServers

Для оптимальной производительности хорошо иметь несколько копий сервера Apache, простаивающих в ожидании запросов. Если доступны запасные серверы, то вам не понадобится ждать, пока экземпляр сервера будет скопирован с диска в оперативную память и запрос будет обработан. Директива MinSpareServers сообщает серверу Apache, сколько потребуется копий для предупреждения возможных скачков нагрузки. Используйте эту директиву для настройки производительности системы вместе с директивой MaxSpareServers.

```
MinSpareServers 5
```

3.3.22. Ограничение потери ресурсов: директива MaxSpareServers

Используйте эту директиву для определения максимального числа запасных серверов, простаивающих в ожидании скачков нагрузки. См. MinSpareServers.

```
MaxSpareServers 10
```

3.3.23. Количество серверов: директива startservers

Эта директива определяет количество копий файла httpd, создаваемых в момент загрузки сервера. Заметьте, что это число не должно быть меньше числа, заданного в директиве MinSpareServers, и что остальные серверы в любом случае созданы в соответствии с установками. Подойдет любое значение, совпадающее даже приблизительно.

```
Startservers 5
```

3.3.24. Знание ваших возможностей: директива MaxClients

Эта директива устанавливает максимальное количество одновременно работающих процессов. Таким образом ограничивается количество одновременно подключенных клиентов. Очень важно следить за тем, чтобы этот показатель не был слишком низким. Оптимальное значение можно получить, вычитая объем оперативной памяти, который вы хотите выделить для других процессов из общего объема доступной памяти, и разделить полученный результат на объем оперативной памяти, необходимой для работы одного httpd-сервера.

Обратите внимание, что количество процессов сервера ограничивается еще и переменной HARD_SERVER_LIMIT, которая устанавливается во время компиляции двоичного файла. По умолчанию значение HARD_SERVER_LIMIT составляет 256.

```
MaxClients 150
```

3.3.25. Ограничение возможности процесса заглушить сервер: директива MaxRequestsPerChild

Эта директива устанавливает максимальное количество запросов, которые может обработать порожденный процесс до того, как он завершит свою работу. Зачем? Предположим, какая-нибудь из поддерживаемых библиотек потребует настолько

много оперативной памяти, что она поглотит все ресурсы и спровоцирует отказ системы. Например:

```
MaxRequestsPerChild 30
```

3.3.26. Ограничение области действия директив: директива <Directory>

Как было сказано в главе 1, "Основные концепции", директивы, контролирующие доступ (Options, AllowOverride), обычно заключаются в своеобразные скобки, ограничивающие область их действия (<Directory>). Например, директивы, заключенные в такие директивные скобки, будут действовать только на файлы, находящиеся в каталоге /opt/apache/htdocs:

```
<Directory /opt/apache/htdocs>
</Directory>
```

Примечание

Директивы управления доступом, применимые к каталогу, применимы также и ко всем его подкаталогам.

3.3.27. Директива Location

С другой стороны, диапазон действия привилегий может быть ограничен URI (Uniform Resource Identifier), расположенным на вашем узле. Например, чтобы отказать в доступе к URI secure_ stuff всем пользователям, кроме локальных, достаточно задать команду:

```
<Location /secure_stuff>
      order deny, allow
      deny from all
      allow from 127.0.0.1
</Location>
```

3.3.28. Директива Options

Эта директива используется в файле .htaccess совместно с директивами <Directory> и <Location> для определения типа доступа, который может получить процесс пользователя к этой директории. Синтаксис — слово Options, сопровождаемое одной или более опциями из табл. 3.1.

Таблица 3.1. Опции доступа для файла .htaccess, директив <Directory> и <Location>

Опции	*Назначение*
All	Активизировать все опции за исключением Multiviews.
ExecCGI	Активизировать выполнение CGI-сценариев.
FollowSymLinks	Сервер просматривает символические связи в этом каталоге. Рекомендуется, даже если сервер просматривает символические ссылки, не менять пути, используемого для поиска соответствия в разделах <Directory>.
Includes	Активизировать использование вставок на стороне сервера.

Окончание табл. 3.1

Опции	Назначение
IncludesNOEXEC	Вставки на стороне сервера разрешены, но команды #exec и #include в CGI-сценариях отключены.
indexes	При этой опции, если URL ссылается на каталог, где отсутствует файл DirectoryIndex (чаще всего файл index.html), сервер возвращает отформатированную распечатку содержимого этого каталога.
Multiviews	Разрешить вывод содержимого в формате Multiviews.
SymLinksIfOwnerMatch	Просматривать символические связи только в случае, когда владелец файла или каталога совпадает с владельцем ссылки.

В случае, когда к каталогу применено несколько опций, самая последняя отменяет действие всех других. Действие опций не суммируется. Однако, если опциям директивы Options предшествует символ "+" или "—", опции действуют следующим образом:

- опции, которым предшествует символ "+", добавляются к списку уже действующих опций
- опции, которым предшествует символ "—", удаляются из списка действующих опций

3.3.29. Директива AllowOverride

В разделе "Ограничение диапазона действия директив" главы 1, "Основные концепции" упоминалось, что один механизм управления доступом заключается во включении во все каталоги файлов с именем .htaccess. Поскольку директивы, имеющиеся в этом файле, могут конфликтовать с директивами, заданными директивой Options (см. выше), для того чтобы определить, какой из вариантов возымеет действие, воспользуемся директивой AllowOverride. Эта директива принимает значения All, None или любую комбинацию значений Options, FileInfo, AuthConfig и Limit. Детали синтаксиса директивы можно найти в приложении А, "Основные директивы".

```
AllowOverride None
```

Существует возможность отменять установки, задаваемые не только файлом .htaccess. Имя такого файла можно задать с помощью директивы AccessFileName. Напомним, что установка директивы AllowOverride в on вызывает полный просмотр сервером Apache дерева файловой системы в поисках заданного конфигурационного файла (например, при доступе к файлу /apache/htdocs/index.html будет произведен поиск конфигурационных файлов в каталогах /, /apache и /apache/htdocs). А это, естественно, приведет к потере быстродействия всей системы.

3.3.30. Директива order

Эта директива является стандартной для модуля mod_access. Директива order задает порядок рассмотрения директив allow или deny. Для того, чтобы задать этот порядок, просто ответьте на вопрос, чего больше вы хотите: разрешить доступ большему количеству узлов или отказать в доступе. Если верно первое, воспользуйтесь командой:

```
order allow, deny
```

В противном случае:

```
order deny, allow
```

3.3.31. Директива allow

Директива allowзадает узел или домен, которому будет разрешен доступ к ресурсам сервера. Это значение может быть задано цифрами (192.168.20) или символическим именем *(righthere.net)*. Однако следует помнить, что в случае указания символического имени, потребуется его проверка. А это, как известно, снижает производительность сервера. В случае задания имени домена символическим образом, не забудьте задать ведущую точку *(.sample.org,* а не *sample.org)*.

```
allow from all
```

3.3.32. Директива deny

Директива deny задает узел или домен, которому будетотказано в доступе к ресурсам сервера. Это значение может быть задано цифрами (192.168.20) или символическим именем. В случае задания имени домена символическим образом обязательно укажите ведущую точку *(.sample.org, a* не *sample.org)*.

```
deny from all
```

3.3.33. Где расположены файлы HTML: директива DocumentRoot

Эта директива задает путь к каталогу, из которого будут по умолчанию выбраны все запрошенные документы. Обратите внимание, что для эффективного разрешения доступа пользователям к другим каталогам, можно использовать символические связи и псевдонимы. Следует заметить, что ошибка в выборе такой директивы (например, задание в качестве каталога root, "/") может создать совсем необязательные "дыры" в системе защиты. Лучше отвести для этих целей специальное место на диске и создать каталог DocumentRoot. Продолжая пример, начатый в главе 2, "Инсталляция Web-сервера Apache", обозначим директиву:

```
DocumentRoot /opt/apache/htdocs
```

3.3.34. Место размещения домашних страниц пользователей: директива UserDir

Эта директива задает имя подкаталога в корневом каталоге пользователя, к которому идет ссылка при получении запроса -user.

```
UserDir public_html
```

3.3.35. Создание индексов и/или поиск по индексам: директива DirectoryIndex

После того как сервер Apache связал заданный URL с каталогом, он возвращает файл или файлы, которые находятся в этом каталоге. По соглашению это файл index.html, но можно задать любое другое имя. Если задан список имен файлов, сервер Apache возвратит первый элемент списка, который будет найден в каталоге.

```
DirectoryIndex index.html index.htm index.cgi index.shtml
```

Если в списке отсутствует необходимый файл, сервер Apache возвращает пользователю листинг содержимого каталога. С точки зрения безопасности это плохо. Чтобы

этого избежать, можно задать абсолютный путь к стандартному файлу, содержащему сообщение "Not found", или что-то подобное в конец вашего списка.

3.3.36. Директива FancyIndexing

Установив значение директивы FancyIndexing в on, можно создать таблицу, содержащую столбцы Icon, Name, Last Modified, Size и Description.

```
FancyIndexing on
```

3.3.37. Директивы AddIcon, AddIconByType, AddIconByEncoding

Директивы AddIcon позволяют связывать двоичный растровый файл с определенным типом файла. Ассоциированный файл изображения появляется на экране во время генерации индекса. Описание синтаксиса и логики этой директивы можно найти в приложении А, "Основные директивы".

3.3.38. Директива DefaultIcon

Этой директивой можно задать стандартное изображение, которое будет появляться при распечатке каталога, а среди директив AddIcon соответствующее изображение отсутствует.

```
DefaultIcon    /icons/unknown.gif
```

3.3.39. Директива AddDescription

Эта директива предназначена для включения в файл краткого описания.

```
AddDescription "Access Policy" README.HTML
```

3.3.40. Директива ReadmeName

Эта директива используется для задания имени README-файла для модуля mod_autoindex, которое будет включено в автоматически генерируемые индексы каталогов.

```
ReadmeName README.html
```

3.3.41. Директива HeaderName

Этой директивой задается имя файла, который будет автоматически подключен к генерированным индексам.

```
HeaderName HEADER
```

3.3.42. Директива IndexIgnore

Эта директива может использоваться для задания набора файлов, которые не будут включаться в автоматически генерируемые индексы каталогов.

```
IndexIgnore */.??* *~ *# */HEADER* */README* */RCS
```

3.3.43. Директива AccessFileName

Эта директива задает имя файла, в котором, в случае его обнаружения в обычном каталоге, можно будет найти директивы управления доступом.

```
AccessFileName .htaccess
```

3.3.44. Директива DefaultType

Эта директива сообщает серверу, какой MIME-тип будет принят по умолчанию в том случае, если тип файла невозможно определить по его расширению. Рекомендуется задавать текстовый тип:

```
DefaultType text/plain
```

3.3.45. Директива AddLanguage

Эта директива предупреждает сервер о языке документа на основании включения соответствующего суффикса в имя документа. Например, если задана следующая директива

```
AddLanguage en .en .english
```

броузер поймет, **что файлы** filename.html.en и filename.html.english содержат **текст на английском языке.**

Конечно, эта информация не имеет смысла для броузера, есл и пользователь сам не задал предпочтительный язык.

3.3.46. Директива LanguagePriority

Эта директива предназначена для разрыва связей при согласовании содержимого. Перечень языков дается по мере уменьшения приоритета.

```
LanguagePriority en fr de
```

3.3.47. Директива Alias

Эту директиву можно использовать для посылки запросов по псевдонимам существующих каталогов. Команда будет иметь вид:

```
Alias /icons/ /usr/local/etc/httpd/icons/
```

3.3.48. Директива ScriptAlias

Эта директива задает каталоги, в которых содержатся сценарии сервера. Она имеет следующий формат: ScriptAlias **псевдоним** имя_каталога. Команда будет иметь вид:

```
ScriptAlias /cgi-bin/ /usr/local/etc/httpd/cgi-bin/
```

3.3.49. Директива AddType

Директива AddType ассоциирует MIME-тип с расширением файла. Действие этой директивы аналогично добавлению аналогичной информации в файл mime.types. Сервер Apache передает тип содержимого броузеру пользователя. Эта информация используется клиентом для обработки информации, полученной от сервера Apache.

3.3.50. Директива AddHandler

Стандартная конфигурация сервера Apache имеет семь различных дескрипторов содержимого для платформ Unix плюс еще один дескриптор для платформы Windows. Директива AddHandler предназначена для установки связи дескриптора с типом или типами файлов. Перечень встроенных дескрипторов приведен в табл. 3.2.

Таблица 3.2. Встроенные дескрипторы сервера Apache

Дескриптор	*Назначение*
cgi-script	Выполнить URL как сценарий CGI.
knap-file	Указанный URL содержит отображаемый образ.
isapi-isa	Только для среды ОС Windows. При вызове URL загрузить динамические библиотеки (DLL) в формате ISA.
server-info	Создать страницу, содержащую информацию о конфигурации сервера.
server-parsed	Произвести анализ любого такого файла для вставок на стороне сервера.
server-status	Генерировать страницу, содержащую информацию о состоянии сервера.
type-map	Рассматривает указанный URL как карту типов.

Кроме того, можно дополнительно создать свои собственные дескрипторы. Как это делается, смотрите в главе 12, "Состав модуля".

Например следующая директива ассоциирует CGI-сценарий с расширениями файлов .pl и .ksh:

```
AddHandler cgi-script .pl .ksh
```

С другой стороны, дескрипторы могут ассоциироваться с определенным местом в файловой системе:

```
<Location /cgi>
      AddHandler cgi-script
</Location>
```

3.4. Операционная система Windows

Настройка сервера Apache для работы под управлением ОС Windows в целом аналогична настройке сервера под ОС Unix. В подкаталоге conf, находящемся в корневом каталоге сервера Apache, можно найти несколько конфигурационных файлов httpd.conf и srm.conf. В основной своей массе директивы аналогичны, используются традиционным для Unix-систем образом и имеют одно и то же действие.

И это подобие настолько близко, что даже несмотря на то, что в файловой системе Windows используется совсем другое соглашение по присвоению имен файлам, чем в ОС Unix, во внутренней системе обработки сервера Apache в качестве ограничителей в спецификациях каталогов используется символ косой черты. Да, это так, в качестве ограничителей в спецификациях каталогов нужно использовать символ косой черты "/". При этом устройство указывать не обязательно. При отсутствии явно заданного устройства сервер Apache будет полагать, что все ссылки сделаны к устройству, содержащему программу Apache.

3.4.1. Отличия от ОС Unix

В среде Unix сервер Apache для обработки запроса, поступившего от пользователя, запускает отдельный процесс. В среде Windows в этом нет необходимости, так как в реализации сервера Apache для этой системы поддерживается многопоточность. Однако парадигма порожденных процессов настолько глубоко въелась в идеологию сервера

Apache, что отказ от нее при переносе в ОС Windows не имеет никакого смысла. Поэтому на машинах Windows всегда сосуществуют два процесса Apache (процесс-родитель и порожденный процесс). Запросы пользователей обрабатываются отдельными нитями порожденного процесса. Различия, которые таким образом были заложены в основу Apache, вызвали появление нескольких уникальных директив.

3.4.2. Директива MaxRequestsPerChild

Эта директива вызывает отключение порожденного процесса по достижению определенного числа обработанных запросов. Так как под управлением ОС Windows работает только один порожденный процесс, рекомендуется устанавливать максимально возможное значение.

3.4.3. Директива ThreadsPerChild

Эта директива ограничивает количество одновременно обрабатываемых порожденным процессом запросов. Она предназначена для предотвращения ситуации, когда мощности системы не будет хватать для обработки запросов пользователей. По умолчанию это значение равняется 50, и оно должно изменяться в зависимости от мощности системы.

3.5 Операционная система Mac OS X

Хотите верьте, хотите нет, но операционная система OS X разработана на основе ОС Unix BSD 4. Все это скрыто от глаз отличным графическим интерфейсом, всегда характерным для компьютеров Макинтош. У меня есть подозрения, что полный дистрибутив ОС OS X содержит сервер Apache с отличным графическим интерфейсом, но на момент создания этой книги такого интерфейса в дистрибутив OS X включено не было. С интерфейсом или без него сервер Apache, работающий под управлением OS X, можно инсталлировать с помощью файлов ASCII. Чтобы запустить приложение, необходимо:

1. щелкнуть на пиктограмме Desktop, расположенной внизу экрана;
2. в меню File выбрать окно New Finder;
3. для запуска терминального приложения запустить Terminal.app.

3.5.1. Имя сервера

Установить имя сервера Apache в ОС Mac OS можно следующим образом. Для этого необходимо войти в раздел Network установки System Preferences, щелкнуть на закладке Services и ввести имя сервера.

В следующей таблице сведены все файлы сервера Apache, известные в ОС Unix, и их аналоги, работающие под управлением OS X.

ОС *Unix*	OS X
httpd	/usr/sbin/apache
httpd.conf	/Local/Library/Webserver/Configuration/apache.conf
srm.conf	<**Не используется — СМ.** apache.conf>
access . conf	<**Не используется — см.** apache.conf>
apachectl	/usr/sbin/apachectl

Вот еще одна таблица, в которой сведены все каталоги сервера Apache, известные в ОС Unix, и их аналоги из OS X.

Стандартный каталог	*Каталог OS X*
$APACHE/conf	/local/library/Webserver/Configuration
$APACHE/cgi-bin	/local/library/Webserver/CGI-Executables
$APACHE/htdocs	/local/library/Webserver/Documents
$APACHE/htdocs	/local/library/Webserver/Logs

ОС OS X поставляется со всеми общими модулями, скомпилированными как разделяемые объекты. Эти модули размещены в каталоге /System/Library/Apache/Modules. Вот перечень модулей, которые включены в бета-дистрибутив:

- httpd.exp
  libdav.so
  libproxy.so
  libssl.so
  mod_access.so
  mod_actions.so
  mod alias.so
  mod_asis.so
  mod_auth.so
  mod_auth_anon.so
  mod_auth_dbm.so
  mod_autoindex.so
  mod_cern_meta.so
  mod_cgi.so
  mod_digest.so
  mod_dir.so
  mod_env.so
  mod_expires.so
  mod_headers.so
  mod_imap.so
  mod_include.so
  mod_info.so
  mod_log_config.so
  mod mime.so
  mod_mime_magic.so
  mod_negotiation.so
  mod_rewrite.so
  mod_setenvif.so
  mod_speling.so
  mod_status.so
  mod_unique_id.so
  mod_userdir.so
  mod_usertrack.so
  mod_vhost_alias.so

Их загружают обычным способом с помощью директивы LoadModule. В качестве примера можно посмотреть конфигурационные файлы.

3.5.2. Тип сервера

В ОС OS X опция inetd, задающая тип сервера, отсутствует. Тип сервера устанавливается в "standalone".

Глава

4

ЗАПУСК, ПЕРЕЗАПУСК И ОСТАНОВКА СЕРВЕРА

В этой главе...

4.1. Введение

Эта глава посвящена тонкостям процедуры запуска, перезапуска и остановки Web-сервера Apache. Материал этой главы содержит описание опций командных строк, поведения сервера во время запуска и, конечно, самых различных методов запуска сервера. Для тех, кто ничего кроме запуска или остановки сервера не знает и не хочет знать, можно посоветовать воспользоваться сценарием apachectl, который можно найти в любом дистрибутиве Web-сервера версий больше 1.3.

```
$APACHE/bin/apachectl [start|stop|restart]
```

Перед запуском сервера Apache и после внесения изменений в конфигурацию для проверки можно воспользоваться опцией configtest.

```
$APACHE/bin/apachectl configtest
```

Сценарий apachectl **работает с опциями** start, stop, restart **и** configtest. **С них и следует начинать работу с сервером.**

4.1.1. Операционная система Win32

Благодаря фундаментальным различиям между парадигмами операционных систем Windows и Unix, процедура запуска и остановки сервера Apache под управлением ОС Win32 имеет мало общего с аналогичными задачами самых разных клонов ОС Unix. Поэтому пользователям ОС Windows рекомендуется сразу же перейти к разделам, посвященным ОС Windows.

4.2. Запуск сервера Apache

Web-сервер Apache можно запустить в двух режимах: автономного сервера или для работы под управлением суперпроцесса inetd[1]. Режим задается директивой ServerType, которая устанавливается в standalone или inetd (см. главу 3, "Конфигурирование Web-сервера Apache").

Сценарий apachectl используется и для тестирования конфигурации на синтаксические ошибки или ошибочные директивы.

```
apachectl configtest
```

Можно порекомендовать сначала его протестировать, так как опция start не возвращает диагностических сообщений в случае возникновения ошибок. Если все в порядке (он возвращает сообщение "Syntax OK"), сервер Apache запускается следующим образом:

```
apachectl start
```

При возникновении ошибки в момент запуска сервера будет получено сообщение об ошибке. Для проверки задайте броузеру один из ваших URL. Пример такой Web-страницы показан на рис. 4.1.

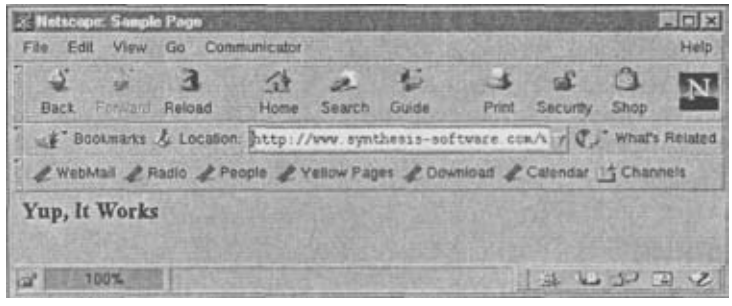

Рис. 4.1. Пример Web-страницы

Если это не дает результатов, проверьте, работает ли сервер вообще.

```
ps -ax | grep httpd
```

Эта команда выводит список запущенных процессов, из которого отфильтровываются только процессы Apache:

```
26358  ??  I      0:00.21 /usr/local/apache/bin/httpd
26638  ??  S      0:01.15 /usr/local/apache/bin/httpd
26711  ??  I      0:00.17 /usr/local/apache/bin/httpd
26712  ??  I      0:00.08 /usr/local/apache/bin/httpd
26976  ??  I      0:00.08 /usr/local/apache/bin/httpd
26993  ??  I      0:00.06 /usr/local/apache/bin/httpd
27018  ??  I      0:00.11 /usr/local/apache/bin/httpd
27020  ??  S      0:00.06 /usr/local/apache/bin/httpd
27412  ??  I      0:00.01 /usr/local/apache/bin/httpd
28361  ??  Ss     2:45.46 /usr/local/apache/bin/httpd
```

Если экземпляры процесса сервера (httpd) работают без ошибок, но Web-страницу увидеть нельзя, одной из причин может быть то, что броузер не может подобрать соответствие имени узла, которое было ему задано. Эту гипотезу можно проверить, указав непосредственно IP-адрес — http://192.168.100.1.

[1] Детальную информацию о процессе inetd можно найти в приложении E, "Концепция Unix"

Если безрезультатно — проблема заключается в соответствии имени. Решением этой проблемы является создание DNS (Domain Name Service) (или возможно в случае intranet — локального хост-файла), в котором будет известно о вашем существовании. Этот процесс обычно заключается в заполнении форм и в заполнении чека. Процедура досконально известна провайдеру.

Другим временным решением проблемы соответствия имени можно считать объявление имени сервера и IP-адреса в ваших хост-файлах. Этот файл можно найти на Unix-машинах в каталоге /etc, его можно создать в корневом каталоге на Windows-машинах, если этого никто еще не сделал до вас. Формат записей этого файла унифицирован и не зависит от платформы. Это формат

```
цифровой_IP_адрес  [псевдонимы],
```

который на практике выглядит следующим образом:

```
192.168.100.1 www.fakesite.com fake
```

Если все попытки подключиться к серверу Apache были безуспешны, но при этом не появлялось никакого сообщения, следует обратиться к разделу "Исправление ошибок", находящемуся в этой главе.

4.2.1. Запуск сервера с помощью процесса inetd

Из соображений производительности сервер Apache лучше запускать как автономный процесс. С другой стороны, для повышения уровня безопасности, которого можно добиться с помощью упаковщика tcp, работающего под управлением демона inetd, сервер Apache можно будет запустить с помощью строки в файле /etc/inetd.conf:

```
httpd stream tcp nowait httpd /opt/apache/bin/httpd -f
/opt/apache/conf/httpd.conf
```

Реальное размещение программ и конфигурационных файлов может варьироваться.

4.2.2. Запуск сервера под управлением ОС Windows

Во всех системах Windows сервер Apache запускается из меню Start. Для этого достаточно выбрать Start Apache. Таким образом сервер Apache запускается как консольное приложение (рис. 4.2).

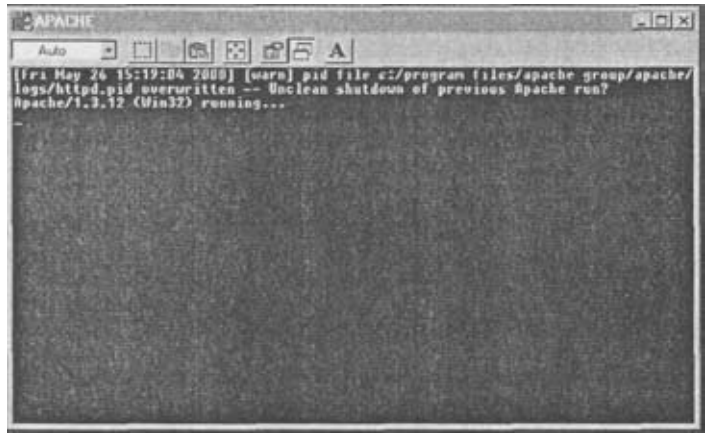

Рис. 4.2. Работа сервера Apache как консольного приложения ОС Window.

Это окно будет присутствовать столько, сколько работает сервер Apache. Оно может быть свернуто.

Работа сервера Apache в режиме консольного приложения послужит сдерживающим фактором. Например, в этом случае нет возможности для запуска приложения во время загрузки компьютера. К сожалению, для пользователей ОС Windows 95 и Windows 98 метод запуска сервера Apache как консольного приложения является единстве нным.

В среде Windows NT сервер Apache может быть запушен как сервис (см. главу 3, "Конфигурирование Web-сервера Apache"). Это самый лучший метод, так как сервисы запускаются во время загрузки системы и продолжают работать даже если пользователь, инициировавший его запуск, вышел из системы. Более того, вход в систему необязателен. Для запуска сервера Apache в качестве сервиса ОС Windows NT введите следующую команд ную строку:

```
NET START APACHE
```

Для остановки сервиса Apache задайте команду:

```
NET STOP APACHE
```

Если вы не любите работать с командной строкой, в ОС Windows NT есть возможность запуска сервера Apache из экранасервисов: Start=>Settings=>Control Panel=>Services.

Если он не запустился автоматически во время загрузки,выделите сервис и щелкните по клавише Start. Проверить работу сервера Apache можно с помощью диспетчера задач Windows NT (для запуска диспетчера задач нажмите комбинацию клавиш <Ctrl+Alt+Delete>). Это показано на рис. 4.3.

Рис. 4.3. Диспетчер задач ОС Windows NT

4.2.3. Запуск сервера под управлением ОС Mac OS X

Для запуска сервера Apache под управлением Mac OS необходимо в разделе Network установки System Preferences щелкнуть на закладке Services и переключить кнопку Web Server из положения Off в положение Ол. Кроме того, существует возможность запуска сервера сценария apachectl из командной строки:

```
apachectl start
```

В OS X процесс Apache называется apache, а не httpd. Проверить работу сервера можно с помощью команды:

```
ps aux i grep apache
```

После переключения радиокнопки Web-сервера из положения Off в положение On (глава 2, "Инсталляция Web-сервера Apache") сервер запускается автоматически. Укажите на броузере адрес узла Mac. Если все хорошо, появится экран с сообщением "Powered by Mac OS X".

4.3. Опции командных строк

Как было сказано ранее, сервер Apache можно запустить и из командной строки. Здесь, как и положено, существует большое разнообразие опций, некоторые из них могут обеспечить такую же функциональность, как и директивы конфигурации. В целом лучше работать с директивами в конфигурационном файле, чем вводить опции в командном файле или в сценариях.

Опции командных строк перечислены в табл. 4.1.

Таблица 4.1. Опции командной строки apache

Опция командной строки	*Назначение*
-d serverroot	Эта опция используется для определения начального значения директивы serverRoot. Все разночтения между значением, определенным в командной строке, и значением, заданным в конфигурационном файле, разрешаются в пользу значений, установленных в конфигурационном файле. Значение по умолчанию составляют: для ОС Unix — /usr/local/apache; для ОС Windows — /apache; для ОС OS/2 — /os2httpd; для ОС OS X — /Local/Library/Webserver.
-D name	Задать имя, которое позднее будет использоваться в директивах if Define. Эти директивы используются для условного включения/выключения различных возможностей сервера.
-f config_flie	Эта директива предназначена для задания имени файла, содержащего директивы конфигурации. Файлу можетбыть присвоен абсолютный или относительный путь. Стандартное имя этого файла httpd. conf, стандартный каталог conf, расположенный в корневом каталоге сервера.
-C "directive"	Эта опция задает выполнение указанной директивы до чтения конфигурационных файлов сервера Apache.
	Запуск сервера в одно-процессном режиме работы (только в целях отладки); процесс жестко привязан к терминалу и не создает порожденных процессов. Не прибегайте к этому режиму для создания обычного Web-сервера.
	Отобразить информацию о версии и выйти.
	Отобразить дату создания, основную версию и опции, заданные при компиляции.

Окончание табл. 4.1

Опция командной строки	Назначение
-L	Вывести перечень директив, вероятные аргументы, описание возможного содержания для каждой директивы (сервер Apache 1.3.4 и выше).
-l	Для сервера Apache 1.3.4 и выше. Отобразить все скомпилированные модули. Для сервера Apache от 1.2 до 1.3.3: перечень директив, их аргументов и описание ожидаемого содержимого каждой директивы.
-h	Для сервера Apache 1.3.4 и выше отобразить опции командной строки и выйти. Отобразить интерпретацию установок vhost из конфигурационного файла, не запуская сервер.
-t	В тестовом режиме прочесть и проинтерпретировать конфигурационные файлы, не запуская при этом сервер. Если конфигурационные файлы не содержат ошибок, будет выведено сообщение "Syntax OK" и нулевой код завершения. При обнаружении ошибки выводится соответствующее сообщение об ошибке и возвращается ненулевой код. Такой способ тестирования проверяет существование всех корневых каталогов. На выполнение этой операции потребуется определенное время. Для более быстрого теста воспользуйтесь опцией -т.
-т	Не очень тщательная проверка. При этом читаются и интерпретируются конфигурационные файлы, но сам сервер не запускается. Если конфигурационные файлы сформированы правильно, в результате этого теста будет получено сообщение "Syntax OK" и кодошибки, равный 0. При обнаружении ошибки выводится соответствующее сообщение об ошибке и возвращается ненулевой код. Не проверяет корневой каталог. Для более тщательной проверки следует воспользоваться опцией -t.
-k option	Только для версий, работающих с ОС Windows. Сообщает серверу Apache о том, что необходимо в зависимости от значения опции ("shutdown" или "restart"), закончить работу или перезапуститься. Для версий меньше 1.3.4, отобразить опции командной строки httpd. В более поздних версиях для этой цели используется опция -h.

4.4. Перезапуск сервера Apache

После внесения изменений в конфигурационные файлы для того, чтобы изменения возымели действие, требуется произвести перезапуск сервера. Ранее указывалось, что процесс, который назывался нами просто "Apache", в действительности представляет собой множество экземпляров, выполняемых одновременно. Каждый раз, когда клиент запрашивает соединение (но не больше, чем значение, указанное директивой

MaxClients), для обработки запроса сервер Apache порождает копию самого себя. В результате этого на достаточно загруженном сервере могут присутствовать сотни копий процесса httpd, выполняемых одновременно.

```
ps -aux I grep httpd
```

Удаление каждого порожденного процесса в отдельности требует много усилий, но к счастью этого делать и не требуется. Сервер пользуется преимуществом механизма взаимодействия между процессами, имеющегося в ОС Unix. Этот механизм существует для передачи сигналов. Он позволяет процессам-родителям передавать порожденным процессам сигнал о том, что пришло время остановки.

4.4.1. Процессы-родители и порожденные процессы

ОС Unix не имеет механизма создания новых процессов на пустом месте. В действительности это происходит следующим образом. Предположим, что возникла необходимость в новом процессе. Работающий процесс (например оболочка csh, ksh, bash и т.д.) генерирует запрос на создание копии самого себя с помощью системного вызова `fork()`. Результирующий процесс (порожденный) является точной копией процесса-родителя, сгенерировавшего этот процесс, за исключением идентификатора процесса, который будет абсолютно новым.

С этого момента на одной машине в системе работают два почти идентичных процесса с одним и тем же объемом используемой памяти, аналогичными значениями переменных и т.д. Очевидно, что для того, чтобы такая работа имела какой-то смысл, должен существовать механизм замены содержимого порожденного процесса. Для этих целей служит системный вызов `exec()`.

Зачем об этом беспокоиться? Порождающий процесс сервера Apache следит за PID (Process IDentification number — идентификационный номер процесса) и имеет право на передачу ему сигнала. Следовательно, процесс-родитель можно использовать в качестве источника, посылающего сигналы своим порожденным процессам. Нужно только знать PID процесса-родителя. Такую информацию можно найти в подкаталоге `logs` корневого каталога сервера в файле `httpd.pld` или в другом файле, если таковой был задан директивой PidFile. В нашей конфигурации это файл

```
/opt/apache/logs/httpd.pid
```

или

```
$APACHE/logs/httpd.pld
```

Остается только решить, какой сигнал послать.

4.4.2. Сигналы

В стандартном дистрибутиве ОС Unix располагает широким диапазоном сигналов. Все они предназначены для остановки процесса, а задача программиста заключается в том, чтобы обработать входящие сигналы и обеспечить механизмы, которые будут обрабатывать их соответствующим образом. Сервер Apache реагирует только на три из них. Они перечислены втабл. 4.2.

Таблица 4.2. Сигналы и их значение

Сигнал	*Результат*
TERM	Завершает работу порожденных процессов и завершается сам.
HUP	Завершает работу порожденных процессов, перечитывает конфигурационные файлы, открывает повторно регистрещионные файлы и повторно запускает порожденные процессы.

Окончание табл. 4.2

Сигнал	*Результат*
USR1	Независимо от задач, выполняемых порожденными процессами (например, обслуживание процессов пользователя), процесс-родитель запрашивает остановку порожденных процессов. Порожденные процессы по мере необходимости заменяются новыми порожденными процессами, которые будут использовать информацию из конфигурационного файла.

Сигналы передаются системным вызовом kill(). Если известен PID процесса-родителя, его можно задать определенным образом. Синтаксис такой команды имеет следующий вид:

```
kill -TERM 12345
```

Значительно элегантнее воспользоваться содержимым файла PidFile **(значение по умолчанию** $APACHE/logs/httpd.pld). **Например:**

```
kill -TERM 'cat $APACHE/logs/http.pid'
```

Иногда для этого требуется время. Можно задать отслеживание этого процесса с помощью команды:

```
tail -f /opt/apache/logs/error_log
```

4.4.3. ОС Windows

Под управлением ОС Windows 95 и Windows 98 сервер Apache можно перезапустить с помощью командной строки:

```
Apache -k restart
```

Для ОС Windows NT этакоманда имеет несколько иной вид. Ваш локальный сервер Apache имеет имя http и команда

```
Apache -n "http" -k restart
```

даст сигнал этому сервису перезапуститься. Если вам неудобно работать с командной строкой, можно сначала остановить сервер Apache, а потом повторно его запустить. Эти манипуляции можно произвести в группе Apache, которая находится в меню Start.

4.4.4. ОС Mac OS X

Перезапуск сервера в среде Mac OS X можно осуществить с помощью сценария apachectl:

```
apachectl restart
```

4.5. Остановка сервера Apache

При необходимости остановить все экземпляры сервера Apache посылается сигнал kill.

```
kill -TERM 'cat $APACHE/logs/http.pid'
```

Кроме того, можно воспользоваться сценарием apachectl:

```
$APACHE/bin/apachectl stop
```

4.5.1. Сигналы

Как уже отмечалось, ОС Unix имеет множество способов просигнализировать процессу, что ему пора завершать работу. В зависимости от нужного эффекта можно выбрать следующие сигналы:

- **сигнал** TERM. Сообщает процессу-родителю о том, что необходимо завершить все порожденные процессы. Эта процедура требует определенного времени, но по завершению сам процесс-родитель продолжает существовать. По этому сигналу все обрабатывающиеся в данный момент запросы будут завершены, а новые приниматься не будут.
- **сигнал** HUP. Аналогичносигналу TERM, сигнал HUP вызывает завершение процессом-родителем всех порожденных процессов. Однако после завершения всех порожденных процессов, процесс-родитель считывает все конфигурационные файлы[2], повторно открывает конфигурационные файлы и запускается с новыми установками.
- **сигнал** USR1. Сигнал USR1 предназначен для плавного перезапуска. Получив его процесс-родитель предупреждает порожденные процессы о том, что они должны завершиться, но только после того, как будут обслужены запросы пользователей. Следующее поколение порожденных процессов будет использовать новую конфигурационную информацию, если таковая будет иметь место.

4.5.2. ОС Windows

В среде ОС Windows для этого можно воспользоваться опцией Stop Apache из меню Start или ввести команду

```
apache -k
```

на консоли MS DOS. В ОС Windows NT консольная команда по обыкновению имеет немного другой вид (включает в себя имя сервиса).

```
apache -n "http" -k shutdown
```

Возымеет действие также щелчок по окну Apache и комбинация клавиш <Ctrl+C>.

4.5.3. ОС Mac OS X

Для остановки сервера Apache, работающего под управлением ОС Mac OS X, нужно, войдя в часть Network меню System Preferences и щелкнув по закладке Services, нажать радиоклавишу Stop. А можно воспользоваться возможностями сценария apachectl.

```
apachectl stop
```

4.6. Исправление ошибок

Определить качество работы сервера можно по журналу регистрации ошибок. Точное местоположение журнала регистрации ошибок определяется директивой ErrorLog в файле httpd.conf. При возникновении сбоев и проблем сервер Apache выдает достаточно содержательные диагностические сообщения.

[2] Заметим, что при использовании сигнала HUP порожденные процессы не запустятся при наличии ошибок в конфигурационном файле. Чтобы убедиться, что конфигурационные файлы не содержат ошибок, воспользуйтесь опциями -t или -т командной строки httpd.

Приведем список сообщений, наиболее часто встречающихся при запуске, краткое объяснение и выход из сложившейся ситуации. При возникновении ситуации, не упомянутой в этом списке, лучше всего будет внимательно прочитать сообщение. По мере возможности программисты стараются включать достаточно полные информативные сообщения об ошибках в свои программы. При незначительном усилии и усидчивости рядовой пользователь вполне в состоянии их расшифровать:

- fcntl: F_SETLKW: No record locks available. Эта немного скрытая системная программная ошибка означает, что сервер Apache не имеет возможности заблокировать один из системных ресурсов. В этом случае блокировка имеет отношение к структуре данных, которая используется совместно операционной системой и прикладной программой (аналогичной httpd) для того, чтобы программа взяла временное, но полное управление над определенным системным ресурсом. В случае возникновения такой ошибки во время старта программы скорее всего необходимо изменить местоположение файла блокировки (см. LockFile). Если сервер уже проработал определенное время и возникла такая ошибка, вполне вероятно, что следует перекомпилировать ядро операционной системы с тем, чтобы включить побольше блокировок.
- Cannot, determine host name. Use ServerName directive to set it manually. Эта ошибка достаточно прозрачна. Здесь сервер Apache не может определить, что система вызывает сама себя ибо ей не хватает информации для запуска. В таком случае необходимо отредактировать файл httpd. conf для того, чтобы включить туда не закомментированную директиву (т.е., не предваряемую символом #) типа:

  ServerNamewww.example.com

  или

  ServerName localhost
- setgid: Invalid argument. Группы, определенной директивой Group, на вашем компьютере в действительности не существует. Сначала создайте соответствующую группу или укажите в директиве Group группу, которая в действительности существует на вашей машине.
- Linux Problems. Если сервер Apache не запускается на компьютере, работающем под управлением ОС Linux с сообщением shmget: function not found, это означает, что ядро системы было построено без возможности Sys V IPC (InterProcess Communication). В таком случае необходимо перестроить ядро ОС Linux и включить такую возможность.
- windows Problems. Если сервер Apache не запускается на компьютере, работающем под управлением ОС Windows 95, с ошибкой Unable to Locate WS2_32. DLL, необходимо установить программу winsock2. Она добавляет сетевые возможности. Ее можно загрузить по адресу htt p://www.microsoft.com/windows 95/downloads.
- Error 1067. При возникновении такой ошибки во время работы под управлением Windows NT необходимо установить имя сервера или другие конфигурационные параметры.

Часть II

Администрирование Web-сервера

В этой части...

5. Хостинг нескольких Web-узлов
6. Proxy-серверы и кэширование
7. Регистрация и мониторинг
8. Безопасность
9. Динамические Web-страницы
10. Настройка рабочих характеристик сервера
11. Переназначение адреса
12. Безопасность

Глава

5

ХОСТИНГ НЕСКОЛЬКИХ WEB-УЗЛОВ

В этой главе...

5.1. Введение

До сих пор наше повествование ограничивалось простейшими случаями. Это когда на одной узловой машине установлен один экземпляр сервера Apache и он обслуживает только один IP-адрес для обработки запросов пользователей, поступающих на один Web-узел. В реальной практике такой вариант маловероятен. Более вероятно существование десятков, если не сотен узлов, каждый из которых имеет специфические требования к ресурсам и конфигурации.

Сервер Apache имеет возможность конфигурирования для поддержки множества IP-адресов (см. директивы BindAddress и Listen). Для каждого IP-адреса он может поддерживать множество портов[1]. Каждая комбинация сопровождаемых IP-адресов и портов имеет один или много узлов. В этой главе детализируется три метода настройки сервера Apache для обеспечения хостинга нескольких узлов:

1. Домашние страницы пользователей.
2. Виртуальный хостинг по имени.
3. Виртуальный хостинг по IP-адресу.

Первый метод — пользовательские домашние страницы — самый простой, но он вряд ли удовлетворит пользователей, которые хотят платить деньги за эту услугу. Этот метод заключается в том, что в каталоге User создаются пользовательские подкаталоги, и все запросы к домашним страницам сервер Apache перенаправляет на каталоги пользователей.

[1] Термины IP-адрес и порт детально обсуждаются в приложении А, "Основные директивы".

Виртуальный хостинг является возможностью сервера Apache, которая позволяет одному серверу обрабатывать запросы к множеству различных Web-узлов. Сервер настраивается таким образом, чтобы различать запросы виртуальных узлов по имени, IP-адресу или по тому и другому одновременно. Сервер Apache можно сконфигурировать, чтобы на разные узлы действовали разные директивы (и, следовательно, вели себя по-разному). Для этих целей используется директива virtualHost. По умолчанию виртуальные узлы наследуют свойства основного экземпляра сервера. Однако почти все основные директивы сервера могут быть или проигнорированы или дополнены с помощью директивы virtualHost.

Виртуальный хостинг очень популярен из экономических соображений. От десятков до тысяч узлов с трафиком от самого низкого до среднего могут поддерживаться с помощью лишь одного физического узла. Доменные имена рекомендуется регистрировать. Обычно взимается плата за несколько имен. Виртуальный хостинг осуществляется по имени или по IP-адресу.

Второй метод, виртуальный хостинг по имени, заключается в связи множества имен сервера с одним IP-адресом. По мере увеличения дефицита IP-адресов вероятность использования этого метода возрастает.

Вероятно, вы уже догадались, что третий метод, в иртуальный хостинг по IP-адресу, заключается в том, что одна машина является узлом для множества IP-адресов. Ранее это был один-единственный способ множественного хостинга и сейчас он все еще остается единственным решением для обеспечения трафика, поступающего от старых версий броузеров[2]. IP-адреса могут связываться с несколькими сетевыми картами (NIC). С другой стороны, современные операционные системы позволяют поддерживать множество IP-адресов с помощью одной сетевой карты.

Чтобы изменения вконфигурации возымели действие, сервер Apache, как обычно, необходимо перезагрузить.

5.2. Домашние страницы пользователей

При применении этого метода используется директива UserDir для того, чтобы разместить URL в некоторых каталогах системы, как это показано на рис. 5.1.

5.2.1. Директива UserDir some_directory

Эта директива предназначена для того, чтобы показать, что Web-содержимое будет найдено в конкретном подкаталоге корневого каталога пользователя. Когда работает эта директива, сервер Apache принимает запросы в виде:

```
http://www.example.com/~userguy
```

и использует системные ресурсы для определения корневого каталога пользователя userguy. Скажем, он находится в каталоге /home, таким образом, путь к корневому каталогу пользователя userguy будет составлять /home/userguy. Если действует директива, аналогичная

```
UserDir some_directory,
```

сервер Apache будет искать отображаемое Web-содержимое в подкаталоге some_directory каталога /home/userguy[3]. В соответствии с логикой нашего примера это приведет в каталог

```
/home/userguy/some_directory
```

[2] Броузеры, реализующие стандарты HTTP меньше, чем 1.1, из-за причин, о которых мы расскажем позднее, не могут поддерживать хостинг по имени.

[3] **Значение по** умолчанию public_html

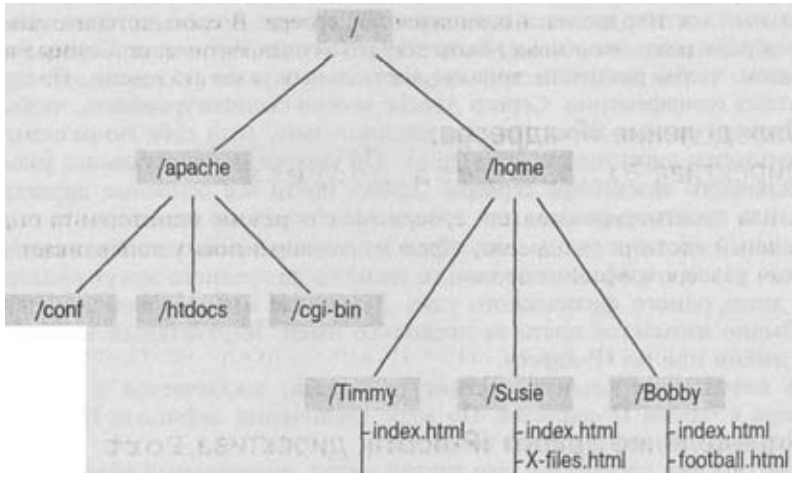

Рис. 5.1. Домашние страницы пользователей

5.2.2. Директива UserDir /an/absolute/path

Есть способ, заключающийся в задании абсолютного пути к определенному каталогу, в котором хранится Web-содержимое всех пользователей. Этот метод предполагает, что каждый пользователь имеет свой собственный подкаталог в каталоге, заданном директивой UserDir.Например, если сервер Apache получает URL

```
http://www.example.com/~timmy/x-flies.html,
```

когда действует директива

```
UserDir /var/user/webspace,
```

то сервер возвращает

```
/var/user/webspace/timmy/x-files.html
```

5.2.3. Директива UserDir /an/absolute/*/with/wildcard

Из всех трех вариантов директивы UserDir этот вариант самый вероятный претендент на применение. В этом методе задается абсолютный путь к каталогу, где пользователи хранят свои Web-документы. Здесь вместо имени пользователя указывается звездочка "*". И когда сервер Apache получает запрос к определенному пользователю

```
http://www.example.com/~susie,
```

он анализирует имя пользователя (об этом свидетельствует символ "~") и замещает звездочку именем пользователя. Например, если в настоящий момент действует директива UserDir:

```
UserDir /home/*/public_html,
```

сервер Apache переадресует этот URL в каталог

```
/home/susie/public_html
```

5.3. IP-адреса и порты

Перед тем как сконфигурировать виртуальный хостинг, необходимо настроить сервер на прослушивание соответствующих портов. По умолчанию сервер Apache следит за IP-

портом 80 для всех IP-адресов, имеющихся на сервере. В соответствии с задачами, стоящими перед сервером, этого может быть достаточно. Директивы, описанные в этом разделе, позволят вам производить настройку ваших адресов и мониторинг порта.

5.3.1. Определение IP-адресов: директива BindAddress address

Директива BindAddress задает серверу Apache режим мониторинга определенного IP-адреса или целого ряда адресов. Следующая директива устанавливает режим прослушивания сервером Apache адреса 192.168.1.10:

```
BindAddress 192.168.1.10
```

Чтобы прослушивались все активные IP-адреса, нужно задать команду

```
BindAddress *
```

5.3.2. Определение одного IP-порта: директива Port portnum

По умолчанию сервер Apache прослушивает IP-порт 80 заданного IP-адреса. Это правильно, так как порт 80 является "стандартным портом", определенным протоколом HTTP, и поэтому к нему будет обращаться наибольшее число Web-броузеров.

С другой стороны, может потребоваться изменить эту стандартную установку и, таким образом, ограничить доступ только теми броузерами, которые знают прослушиваемый порт. Типичным примером такого использования сервера Apache является использование его в качестве proxy—сервера, а еще такой режим работы может пригодиться для настройки корпоративной intranet. Отметим, что в отличие от директивы Listen (которая будет рассмотрена в следующем разделе), только одна директива Port может быть применена в один момент времени. Например, чтобы сервер Apache начал прослушивать порт 4444, необходимо задать директиву

```
Port 4444
```

5.3.3. Определение одного или более IP-порта: директива Listen

В отличие от директивы Port, директива Listen не отменяет действие других директив Listen. Чтобы сервер Apache слушал порт 80 (стандарт HTML) и порт (гипотетический локальный порт), воспользуемся последовательностью директив

```
Listen 80
Listen
```

Кроме того, директива Listen может использоваться для определения IP-адресов. Предположим, что в вышеприведенном примере прослушивается IP-адрес 192.168.1.2. Следующие команды Listen будут иметь аналогичный эффект:

```
Listen 192.168.1.2:80
Listen 192.168.1.2:
```

5.3.4. Настройка множества IP-портов

Можно предложить два метода поддержки множества IP-адресов на одной системе:

- Купить и установить несколько интерфейсных плат.
- В некоторых операционных системах для установки мониторинга одной интерфейсной платы множеством IP-адресов можно воспользоваться командой ifconfig.

Совершенно очевидно, что идея покупки множества интерфейсных карт не нуждается в комментариях. Чего нельзя сказать об использовании команды ifconfig . Команда ifconfig (interface configurationконфигурация интерфейса) выполняет две функции:

- Отображение информации о конфигурации существующего интерфейса.
- Изменение или добавление информации о конфигурации интерфейса.

Первая функция имеет неоспоримую пользу, но не относится непосредственно к серверу. Для этого достаточно командой ifconfig определить имя нужной карты.

```
/home/root> ifconfig eth0
```

В результате будет получен следующий ответ (конечно, в зависимости от типа системы):

```
eth0 Link encap:Ethernet HWaddr 00:20:78:17:9A:EB
inet addr:192.168.1.1 Beast:192.168.1.255 mask:255.255.255.0
UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:!
RX packets:260652 errors:0 dropped:0 overruns:0 frame:0
TX packets:565370 errors:0 dropped:0 overruns:0 carrier:0 collisions:0
```

С другой стороны, команду ifconfig иногда можно использовать для ввода новой информации. Вот команда, которая создает виртуальное устройство eth0:1. Оно будет отслеживать различные IP-адреса (192.168.100.2).

```
/home/root> ifconfig eth0:1 192.168.1.2 netmask 255.255.255.0
```

В случае успешного конфигурирования, устройство eth0:1 ведет себя так, будто оно присутствует в действительности, отвечая на запросы команды pings и запросы пользователей, как если бы вы и правда потратили 50 долларов на новую карту.

5.4. Виртуальный хостинг по имени

Виртуальный хостинг по имени является относительно новой доработкой стандарта HTTP. По этому методу множество различных доменных имен ассоциируется с одним IP-адресом. Все доменные имена зарегистрированы, и все запросы перенаправляются к одному и тому же IP-адресу. Сервер отличает один запрос от другого с помощью заголовка HOST, настроенного для каждого виртуального узла, сконфигурированного на сервере.

Это, без всяких сомнений, хорошее решение несколько снизило скорость истощения адресного пространства Internet. Проблема заключается только в том, что с заголовком HOST работают только броузеры, удовлетворяющие стандарту HTTP 1.1. Поэтому получить доступ к таким виртуальным узлам в помощью устаревших броузеров будет довольно проблематично.

Процесс настройки такого хостинга можно разбить на три этапа:

- Создание и регистрация нового имени виртуального узла.
- Информирование DNS о том, что уже существующий IP-адрес также имеет отношение к имени нового виртуального узла.
- Передача сведений серверу Apache о том, каким образом запросы направляются на виртуальный узел.

5.4.1. Система доменных имен и регистрация имени

Система доменных имен (DNS) — это своеобразный аналог желтых страниц Internet. Это распределенная база данных IP-адресов и связанных с ними доменных имен. Без системы доменных имен или чего-нибудь подобного Internet не смог бы существовать. Без базы DNS, задаваемый в броузере URL является совершенно бесполезным до тех пор, пока соответствующий ему IP-адрес не будет найден в базе данных DNS.

Учитывая продолжающееся расширение Internet, хороших доменных имен остается все меньше и меньше. Уже все возможные трехбуквенные комбинации имен доменов исчерпаны, так что по этому поводу даже не стоит беспокоиться. Большинство английских слов уже тоже использованы. Запрос по получению новых доменных имен можно направлять по адресу http://www.networksolutions.com (см. рис. 5.2).

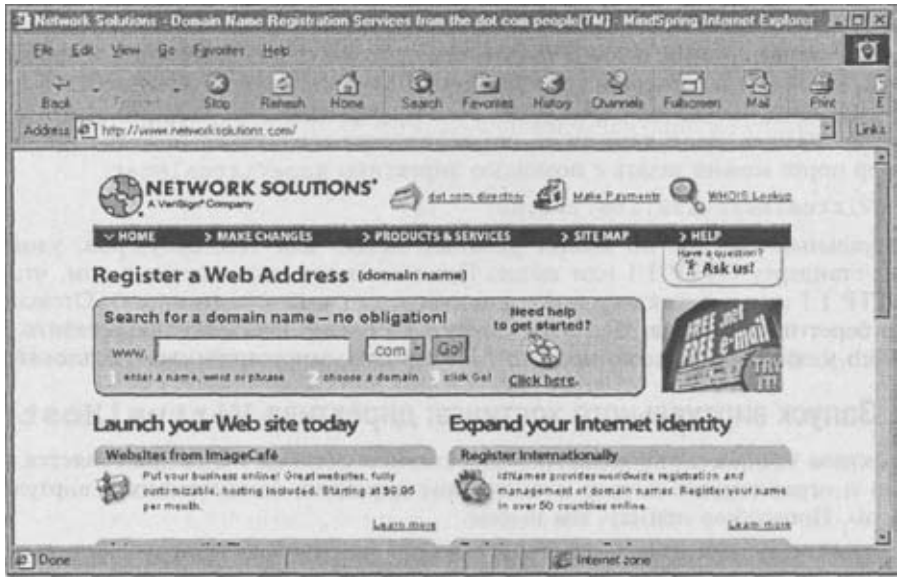

Рис. 5.2. Регистрация доменного имени

Если у вас имеется доменное имя, которое вас во всем удовлетворяет, то худшее позади. В обычной практике обычному пользователю совсем не обязательно прописывать свое доменное имя самостоятельно. Для большинства регистрация DNS заключается в заполнении формы и отсылке ее соответствующим исполнителям провайдера.

DNS представляет собой распределенную базу данных, т.е. ее нельзя найти всю сразу на одном сервере. Запрос пользователя может обойти много серверов, и только после этого будет найдено совпадение. Побочный эффект заключается в том, что прописка нового доменного имени по всемирной сети DNS занимает часы, а иногда даже и сутки.

5.5. Настройка виртуального хостинга по имени на сервере Apache

Хорошо, что виртуальный хостинг по имени осуществляется относительно безболезненно. Для этого достаточно сделать два действия, хотя, откровенно говоря, второе может вызывать затруднения.

1. С помощью директивы NameVirtualHost определить IP-адрес, который будет использоваться для виртуального хостинга.
2. С помощью пары директив VirtualHost выделить директивы, которые будут иметь отношение только к определенному виртуальному Web-узлу.

5.5.1. Назначение IP для виртуального хостинга по имени: NameVirtualHost

С помощью директивы NameVirtualHost задайте IP-адрес виртуального узла в конфигурационном файле httpd.conf. Например директива вида

```
NameVirtualHost 192.168.1.1
```

предупредит сервер Apache о том, что существует возможность получения запроса по адресу 192.168.1.22 к серверам, отличающимся от стандартного сервера. Чтобы сервер Apache смог извлечь какую-то пользу из этой информации, необходимо задать с помощью скобок VirtualHost директивы, специфические для виртуального хостинга.

Номер порта можно задать с помощью директивы NameVirtualHost.

```
NameVirtualHost 192.168.1.1:80
```

Виртуальный хостинг по имени работает только для Web-броузеров, удовлетворяющих стандарту HTTP 1.1 или выше. Причина этого заключается в том, что стандарт HTTP 1.1 имеет директиву HOST, которая задает имя узла (и порта). Отсюда Web-броузер берет информацию. Без этой директивы сервер не сможет определить, какой же из Web-узлов можно ассоциировать с IP-адресом, запрашиваемым пользователем.

5.5.2. Запуск виртуального хостинга: директива VirtualHost

Директива VirtualHost является "операторной скобкой". Она применяется только попарно и ограничивает начало и окончание директив, занимающихся виртуальным хостингом. Продолжая пример, мы имеем:

```
<VirtualHost 192.168.1.1>
    ServerName www.examplel.org
    DocumentRoot/some/other/directory
</VirtualHost>
```

Необходимо обратить внимание на то, что директивы, находящиеся внутри скобок <VirtualHost>, относятся только к виртуальному узлу, заданному директивой ServerName. Директивы, заключенные в скобки <VirtualHost>, отменяют стандартные установки, действующие для данного IP-адреса. Ограничений на количество директив, которые могут быть заключены в операторные скобки <VirtualHost>, нет. Но есть определенные разумные пределы (см. табл. 5.1).

Таблица 5.1. Директивы, *неприменимые* в виртуальном хостинге

Директива	*Назначение*
BindAddress	Директива BindAddress используется для того, чтобы задать один или несколько IP-адресов, прослушиваемых сервером.
Listen	Директива Listen используется для того, чтобы задать IP-адрес и, возможно, порт. Без тестирования и отладки подключение к виртуальному узлу невозможно.
MaxSpareServers	Максимальное количество простаивающих серверов, работающих вхолостую в любой заданный момент времени, для определенного узла не задаются.
MinSpareServers	Минимальное количество серверов, работающих вхолостую в любой заданный момент времени, для определенного узла не задаются.

Окончание табл. 5.1

Директива	***Назначение***
MaxRequestsPerChild	Максимальное количество запросов к порожденному процессу.
PidFile	Определяет размещение файла, содержащего PID первоначального процесса на сервере.
ServerRoot	Определяет размещение глобальных конфигурационных файлов.
ServerType	Задает режим запуска сервера процессом inetd или в качестве автономного процесса.
TypesConfig	MIME-типы должны быть сконфигурированы глобально.
NameVirtualHost	Эта операция производится только за пределами скобок `VirtualHost`.

5.5.3. Виртуальный узел по умолчанию

Использование ключевого слова _default_ вместо IP-адреса свидетельствует о том, что заданная вами конфигурационная информация будет использоваться по умолчанию в случае, когда поступающие запросы не смогут найти нужный виртуальный узел среди существующих узлов. Делать это не обязательно, но сделав это, вы не пожалеете. Текущая конфигурация сервера может быть настолько простой, что это совершенно не потребуется, но при всем при этом такой стандартный подход может оказаться достаточно полезным.

```
<VirtualHost _default_>
    ...стандартные директивы...
</VirtualHost>
```

5.5.4. IP-адрес или доменное имя?

При более внимательном рассмотрении команд, описанных в этой главе (VirtualHost, BindAddress и т.д.), можно заметить, что многие из них скорее предназначены для определения доменных имен, чем IP-адресов. Например набор директив

```
<VirtualHost www.idiots_anonymous.org>
    ...различные директивы...
</VirtualHost>
```

технически правильный и действительно более читабельный, чем при использовании IP-адреса. К сожалению, это отрицательно сказывается на производительности. При хостинге по имени сервер Apache, каждый раз получая запрос, будет вынужден искать заданное имя в DNS, а это существенно замедляет работу.

5.6. Виртуальный хостинг по IP-адресу

Виртуальный хостинг по IP-адресу не предполагает, что пользовательские броузеры будут посылать заголовок Host (это характерно только для броузеров, совместимых со стандартом HTTP 1.1), и, следовательно, в зависимости от требований к узлу, не может претендовать на исключительность. Вот этапы процесса виртуального хостинга по IP-адресу:

1. Создание и регистрация нового имени виртуального узла.
2. Настройка системы таким образом, чтобы она имела возможность отслеживать новые IP-адреса (см. раздел "IP-адреса и порты" в этой главе).

3. Задание в DNS связи между новым IP-адресом и именем узла.
4. Сообщение серверу Apache о том, как можно обработать запросы, направленные к виртуальному узлу.

Большая часть этих требований уже обсуждалась. Сведения о том, как создаются и регистрируются новые имена узлов, можно найти в разделе "Виртуальный хостинг по имени" в этой главе. Процедура создания нового имени является аналогичной. Однако при регистрации имени нового виртуального узла с использованием DNS необходимо создать новый IP-адрес.

Конфигурирование системы с тем, чтобы она могла отслеживать новые IP-адреса, обсуждается в разделе "IP-адреса и порты" ранее в этой главе.

Основное различие между конфигурированием сервера Apache для виртуального хостинга по имени и виртуального хостинга по IP-адресу заключается в том, что во втором случае не требуется прибегать к услугам директивы NameVirtualHost. Чтобы создать узел с именем www.example2.com по адресу 192.168.1.2, необходимо:

```
<VirtualHost  192.168.1.2>
     ServerNamewww.example2.com
</VirtualHost>
```

5.6.1. Комбинирование виртуальных узлов, базирующихся на именах и на IP-адресах

Нет причины, которая могла бы воспрепятствовать объединению обеих подходов на одной системе. Сначала создайте весь нужный виртуальный хостинг по адресу, а потом задайте соответствие адресов для виртуального хостинга по имени. Если директива NameVirtualHost хоть один раз применялась к определенному IP-адресу, то этот адрес уже будет потерян для виртуального хостинга по адресу.

5.7. Что нужно настраивать для виртуального хостинга

Во всех примерах директивы заключены в скобки <VirtualHost> намеренно. Технически совсем не обязательно указывать *что-нибудь* в скобках, хотя совсем бессмысленно их задавать в случае, когда они ничего не содержат. Несколько директив, описанных в этом разделе, можно назвать специфическими только для виртуального хостинга.

Во время конфигурирования директив нужно помнить, что виртуальные узлы наследуют свойства основного сервера Apache. Конфигурация виртуального узла должна отменять, расширять конфигурацию главного сервера или повторять его.

Как минимум, необходимо задать директиву ServerName для каждого виртуального узла:

```
ServerName www.site2.com
```

Другим необходимым элементом является директива DocumentRoot, задающая стартовую точку для любого поиска. Я думаю, что в ситуации, когда для Web-документов различных клиентов отводятся отдельные каталоги, это будет бесспорно уместно.

```
DocumentRoot /home/site2
```

Директива ServerAdmin позволяет указать точный почтовый адрес администратора каждого виртуального узла. Это адрес, куда будет поступать почта с сообщениями о проблемах, возникающих при работе сервера.

```
ServerAdmin admin@site2.com
```

Можно упомянуть также файлы ErrorLog и TransferLog, так как иханализ упрощает отладку и анализ трафика. Однако следует помнить, что Unix-системы (включая и ОС Linux) обычно накладывают ограничение на число файлов, открываемых отдельными процессами. Назначение отдельного регистрационного файла для каждого виртуального узла может быстро превысить этот предел. К сожалению, я вынужден попросить вас обратиться к системной документации, чтобы узнать, каким образом можно повлиять на ограничения, накладываемые каждой конкретной системой. При наличии таких-либо подозрений, проверьте файл syslog вашей системы. ОС Unix очень хорошо фиксирует нарушения такого типа.

Глава

6

PROXY-СЕРВЕРЫ И КЭШИРОВАНИЕ

В этой главе...

6.1. Введение

Proxy-сервер — это компьютер, который настроен для обслуживания внешнего или внутреннего сетевого трафика. Запрошенный ресурс может располагаться или отсутствовать на этом компьютере. В большинстве случаев он действует как посредник, получая запросы от многочисленных пользователей, рассылая запросы удаленным серверам и пересылая клиентам ответы, полученные от серверов. Proxy-серверы можно конфигурировать таким образом, чтобы они сохраняли в кэш-памяти наиболее популярные Web-ресурсы, улучшая тем самым производительность всей системы.

Машина под именем Odin, изображенная на рис. 6.1, настроена таким образом, чтобы действовать в качестве proxy-сервера для его локальной подсети. В этом примере относительно компьютера Odin необходимо подчеркнуть два момента:

- Сервер Apache настроен и работает как proxy.
- Компьютер Odin подключен как к локальной сети (IP-адрес 192.168.100.1), так и к внешнему миру Internet (IP-адрес 135.186.123.123).

Клиенты proxy-сервера (Loki и Fenris) находятся в той же локальной сети, что и Odin (192.168.100), и настроены таким образом, чтобы передавать запросы в Internet через компьютер Odin.

Хотя это и не входит в рамки материала этой книги, важно отметить, что настройка сервера Apache для работы в режиме proxy не имеет никакого смысла, если клиентские машины в этой сети не настроены соответствующим образом. Эта процедура будет описана в конце главы.

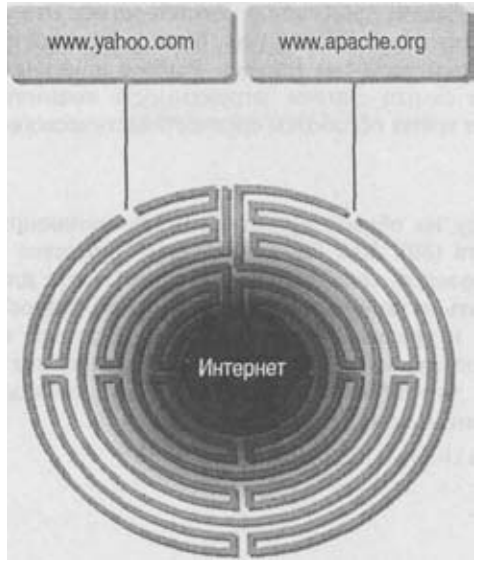

Рис. 6.1. Proxy-сервер

6.2. Стоит ли беспокоиться?

Proxy-сервера, как и все посредники, ведут к дополнительным расходам. Как мы увидим позже, настройка сервера Apache для работы в режиме proxy не очень сложная. Это вопрос двух директив. Однако, если будет принято решение кэшировать популярные Web-страницы, потребуются дополнительные мероприятия. В первую очередь — дисковое пространство на локальном компьютере. Настройка клиентов для работы с proxy может быть автоматизирована, но обычно это требует от пользователей выполнения последовательности действий по настройке броузера. Конечно, если повторять данную, процедуру пятьдесят раз в неделю, это вскоре можно будет делать с закрытыми глазами. Зачем?

Просуммировав, главное преимущество использования сервера в режиме proxy можно Выразить ОДНИМ СЛОВОМ: централизация. Вместо того, чтобы двадцать разработчиков подключалось к Internet через двадцать модемов, следует организовать доступ в Internet по од-

ной телефонной линии с большой пропускной способностью. Это также позволит осуществлять централизованную регистрацию на узлах, блокировать доступ пользователей к сомнительным и нежелательным ресурсам Internet. Выбрав локальное кэширование самых популярных узлов, мы тем самым снизим загруженность входного телефонного канала, что в свою очередь сократит время обработки среднестатистического запроса.

6.2.1. Порты

Можно настроить proxy на обработку запросов, поступающих через порт, отличный от стандартного порта (80). Как бы то ни было, придется вносить изменения в настройки локальных броузеров. Задание конкретного порта для работы с proxy позволит в дальнейшем делать четкое разграничение между запросами на обслуживание с помощью proxy-сервера и другими запросами. В следующем примере виртуальный узел настраивается на прослушивание порта 8888 на предмет наличия запросов к proxy-серверу. Напомним, что директива Listen применяется для глобального включения режима прослушивания конкретного порта.

```
Listen 8888 <VirtualHostl92.168.100.1:8888>
        . . .директивы...
</VirtualHost>
```

6.3. Настройка proxy-сервера

Модуль, обеспечивающий работу proxy, по умолчанию не компилируется. Поэтому, прежде чем начать что-либо делать, необходимо обратить внимание на модуль mod_proxy и включить его в ядро. Добавим mod_proxy в конфигурационный файл и перекомпилируем сервер Apache. После этого необходимо заменить старую программу httpd на новую. И перезапустить сервер[1].

Для запуска proxy-сервера достаточно одной директивы.

```
Proxy-Requests On
```

После запуска сервер Apache будет воспринимать входящие запросы от клиентов, передавать их в Internet, а затем принимать полученные результаты. Если это вам кажется несложным, то вас можно считать настоящим программистом.

В следующем примере сервер настраивается на прослушивание входящего трафика по порту 8888. А потом для работы с этим портом создается виртуальный узел по имени proxy.asgard.com.

```
Listen       8888
<VirtualHost       192.168.100.1:8888>
        ServerName proxy.asgard.com
        ProxyRequests On
</VirtualHost>
```

6.3.1. Ограничение доступа к определенным Web-узлам

Ранее уже было отмечено, что одной из явных выгод, которую можно извлечь из использования Apache в качестве proxy, является возможность ограничения доступа к нежелательным узлам. Ограничения доступа задаются текстовой строкой, а не адресом. Сервер Apache не пропустит запросы, которые соответствуют строкам, заданным директивой ProxyBlock. Например,

```
ProxyBlock kiddiporn.com snuffilm.org
```

[1] Детальное описание этой процедуры можно найти в главах 2 и 4.

6.3.2. Пересылка запросов на другие proxy-серверы

Если в вашей сети имеется более одного proxy-сервера, существует возможность разделить нагрузку по обработке запросов между ними. Директива ProxyRemote позволяет переключать запросы к определенному узлу или транзакции определенного типа на сервер, специально выделенный для решения таких задач. Например

```
ProxyRemote ftp http://ftpsrvr.local.com:8080
```

6.3.3. Задание исключений для удаленного проксирования

Задание пересылки широкого диапазона запросов на удаленный proxy-сервер предполагает и обратное действие: возможность задания исключений. В соответствии с идеологией "отказать многим/гарантировать немногим", применяемой в управлении доступом, это можно сделать с помощью директивы NoProxy. Например директива

```
NoProxy .example.com
```

предотвратит маршрутизацию запросов к ресурсам в локальной области на удаленный proxy-сервер.

6.3.4. Зазеркаливание удаленного узла

Если пересылки запросов на/из удаленного сервер недостаточно, в качестве псевдозеркала можно задать локальный каталог. Задайте в директиве ProxyPass имя подкаталога на вашей машине и URL удаленного узла, и все подкаталоги будут перемаршрутизированы на удаленный узел.

Предположим, что локальный узел имеет имя *www.example.edu,* и вы имеете намерение показать, что подкаталог /distant/mirror является зеркалом узла *www.example.edu.* Директива

```
ProxyPass /distant/mirror http://www.tuchman.edu
```

маршрутизирует все локальные запросы к узлу http://www.example.edu/distant/mirror **через proxy на узел** http://www.tuchman.edu.

6.3.5. Назначение стандартного домена

Есть способ уменьшить объемы работы с клавиатурой. Для этого нужно задать стандартный домен. Директива ProxyDomain помогает пересылать все запросы, в которых домен не задан явным образом, на стандартный домен, который предположительно находится в вашей локальной сети.

```
ProxyDomain .example.com
```

Это позволит после введения краткого URL, наподобие *http://www,* получить доступ к узлу *http://www.example.com.*

6.3.6. Управление доступом

Этот метод позволяет управлять доступом к proxy-серверу с использованием шаблонов для указания того, что все каталоги находятся под управлением proxy.

```
<Directory proxy:*>
      order deny,allow
      deny from [список IP-адресов]
      allow from [список IP-адресов]
<Directory>
```

6.4. Кэширование

Кэшированием называется практика сохранения копий Web-страниц (или чего-то другого), к которым недавно осуществлялся доступ на удаленном сервере, в ожидании того, что они будут запрошены в скором будущем повторно. Если запрос определенной страницы поступает повторно через десять минут после того, как эта страница была закрыта, это сокращает время на поиск имени домена, соединение с удаленным сервером и собственно обслуживание запроса. Если же на протяжении определенного периода времени эту страницу никто не запрашивает, ее копия будет удалена. Издержки этого метода заключаются во временной потере определенного дискового пространства.

6.4.1. Включение режима кэширования: директива CacheRoot

Процесс включения режима кэширования обескураживающе прост. Только одна директива CacheRoot позволяет включать и выключать режим кэширования. Чтобы включить этот режим, необходимо задать каталог, в котором будут храниться кэшированные файлы.

```
CacheRoot /cache
```

6.4.2. Определение размера кэша: директива CacheSize

Стандартный размер кэша до смешного мал — он составляет 5 Кбайт. Чтобы задать другой размер кэша, необходимо воспользоваться директивой CacheSize. Размер кэша задается в килобайтах. Например директива

```
CacheSize 10240
```

задает размер кэша равным 10 Мбайт. Важно отметить, что директива CacheSize задает не верхний предел, а, скорее всего, достижимое значение. Когда объем дискового пространства превышает заданное значение, система начинает удалять файлы. Директивой CacheSize рекомендуется задавать значение, составляющее около 70% объема пространства, требуемого под кэш.

6.4.3. Определение глубины кэширования: директива CacheDirLevels

Ограничивать глубину подкаталогов внутри каталога CacheRoot можно с помощью директивы CacheDirLevels. Задать кэширование на три уровня вглубь каталога можно с помощью директивы CacheRoot

```
CacheDirLevels 3
```

6.4.4. Ограничение длины пути: директива CacheDirLength

Существует еще один способ ограничения объема кэшируемой информации: путем установления верхнего предела длины пути к кэш-файлам. Задать предел для имен подкаталогов в 25 символов можно с помощью директивы

```
CacheDirLength 25
```

6.4.5. Определение срока хранения

В идеале информация о времени истечения срока хранения документа должна поступать вместе с документом. В процессе удаления мусора Apache будет руководствоваться сроками хранения найденных документов. Однако, вследствие того, что многие Web-мастера обладают манией величия, сервер Apache имеет в своем арсенале не-

сколько директив, которые помогают разобраться с нереальными и несуществующими сроками хранения. Они представлены в табл. 6.1.

Таблица 6.1. Директивы, помогающие менять срок хранения

Директива	*Назначение*
CacheMaxExpire	Абсолютный максимум времени хранения, заданный в часах. По умолчанию это значение равно 24 часам.
CacheDefaultExpire	Если файл поступает без срока хранения, используется величина, заданная этой директивой. По умолчанию значение равно 1.
CacheLastModifiedFactor	Если файл поступает без срока хранения, вычисляется период времени от текущего времени до времени последней модификации файла, результат умножается на коэффициент CacheLastModifiedFactor для определения того, сколько будет ждать сервер момента, до истечения срока хранения файла.

6.4.6. Задание интервала между очистками: директива CacheGcInterval

Периодически сервер просматривает кэшированные файлы и удаляет те из них, срок хранения которых истек. Этот процесс известен под названием *сборка мусора*. Одновременно сервер Apache будет соблюдать все ограничения по занимаемым объемам, заданным директивой Cachesize. Интервал между сеансами сборки мусора задается в часах директивой CacheGdlnterval. Обратите внимание, что время можно задавать и в минутах (долях часа). Например директива

```
CacheGdlnterval .5
```

задает интервал, равный 30 минутам.

6.4.7. Отключение режима кэширования: директива NoCache

Чтобы быть полностью уверенным, что каждый раз у вас будет самая свежая версия файла, можно воспользоваться директивой NoCache. Она позволяет отключить кэширование для заданного объекта. Директива NoCache принимает список слов, имен узлов, доменов или их комбинаций, которые исключаются из процесса кэширования, и вы, таким образом, всегда получаете самые последние данные

```
NoCache news www.foo1.com
```

Обратим внимание также на тот факт, что директива NoCache со звездочкой (*) полностью отключает кэширование

```
NoCache *
```

6.5. Настройка броузеров для работы с proxy-серверами

Как было сказано выше, существует еще один момент в настройке сервера Apache для работы в качестве proxy-сервера. Необходимо соответствующим образом настроить броузеры в вашей локальной сети для того, чтобы извлечь наибольшую пользу от работы с proxy. Этот раздел посвящен вопросам настройки броузеров Netscape Navi-

gator и Microsoft Internet Explorer таким образом, чтобы они посылали все свои запросы в Internet через proxy.

В этих целях настроим сервер serverodin. asgard. com какproxy-сервер.

6.5.1. Броузер Netscape Communicator

Настройка броузера Netscape Communicator для работы с proxy-сервером состоит из четырех операций:

1. Активизируйте меню Edit и выберите Preferences. В результате появится экран, изображенный на рис. 6.2.

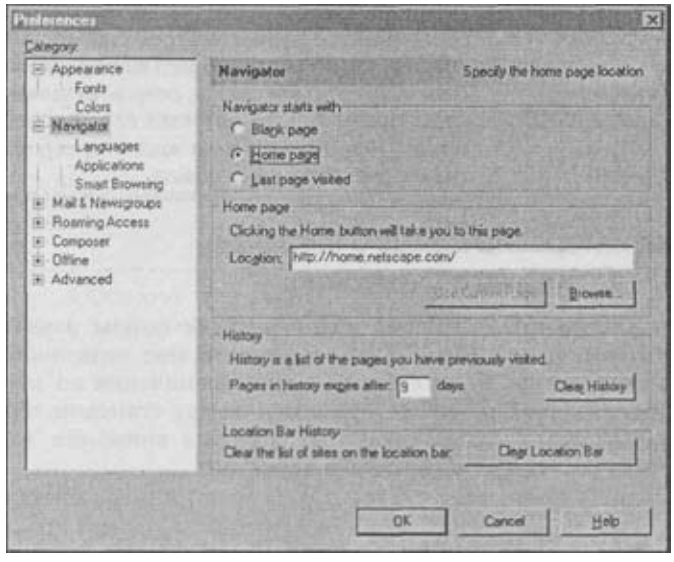

Рис. 6.2. Окно Preferences броузера Netscape

2. В окне, изображенном слева, нажмите на символ "+", расположенный рядом с элементом Advanced. В результате будет получен расширенный список. Выберите из этого списка Proxies.
3. Щелкните на радиокнопке Manual, затем кнопке View, расположенной внизу. В результате появится окно, изображенное на рис. 6.3.
4. Введите URL proxy-серве ра, сконфигурированного на вашем узле. Вам может потребоваться задать порт, отличный от стандартного. После того как все настроено, нажмите клавишу OK.

6.5.2. Броузер Internet Explorer

Настройка броузера Internet Explorer для работы с proxy-сервером состоит из четырех операций:

1. Запустите Internet Explorer. Активизируйте меню View и выберите элемент меню Internet Options. Появится окно Internet Options.
2. В окне Internet Options нажмите закладку Connection.
3. Выберите Access the Internet using a Proxy Server, затем для отображения окна Proxy Settings, который вы видите на рис. 6.4, щелкните по клавише Advanced.

4. Введите URL и порты proxy-серверов, установленных в вашей локальной сети. Для актуализации изменений щелкните по клавише OK.

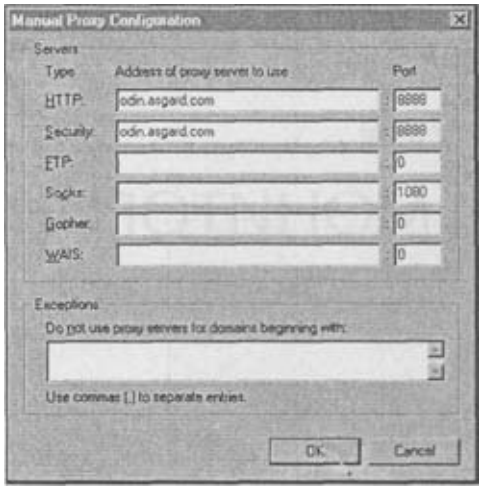

Рис. 6.3. Окно ручной настройки броузера Netscape для работы с proxy-сервером

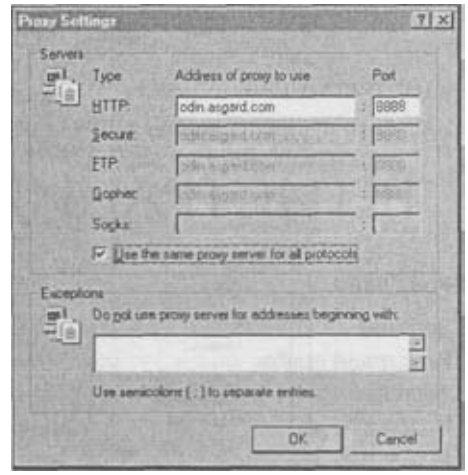

Рис. 6.4. Настройки броузера Internet Explorer для работы с proxy-сервером

Глава

7

РЕГИСТРАЦИЯ И МОНИТОРИНГ

В этой *главе...*

7.1. Введение

После того как сервер инсталлирован и работает, неплохо проверить, *как* он работает[1]. Вся диагностическая информация записывается в текстовых файлах, которые называются регистрационными файлами, их местоположение и тип поддаются настройке. Дополнительную информацию можно получить компиляцией и подключением модулей, отображающих сведения о работе сервера. Эта глава посвящена тому, каким образом можно произвести настройку сервера, чтобы он показывал то, что вас интересует, и как читать эту информацию.

7.1.1. Регистрационные журналы

Сервер Apache создает два основных типа регистрационных журналов: журнал регистрации ошибок и журнал регистрации обмена данных. Отдельные модули могут генерировать свои собственные журналы, но здесь мы этой темы касаться не будем. Вы можете задать уровень серьезности регистрируемых ошибок, начиная с которого будет производиться регистрация в журнале, но не тип регистрируемой информации. И наоборот, возможно задать тип сохраняемой информации об обмене данных, но нет возможности задать уровень этой информации.

7.1.2. Модули

Сервер Apache имеет в своем составе два модуля, отвечающих за динамическое создание и отображение информации о рабочих характеристиках сервера. Первый модуль — это модуль mod_sta tus. Он делает моментальные снимки информации о рабочих характеристиках с возможностью отображения детализированной информации о каждом порожденном процессе. Второй модуль — это модуль mod_info. Отображает подробную информацию о текущих настройках сервера. Оба модуля генерируют отчеты в формате HTML, что позволяет просматривать их с помощью обычного Web-броузера.

[1] Вероятно самым сложным моментом в процессе написания технического материала является необходимость постоянных рефренов, как будто это квинтэссенция мудрости. Да, без сомнений, очень полезно контролировать работу сервера. Кроме того, я — за мир во всем мире, и против рака.

7.2. Регистрация ошибок

Журнал регистрации ошибок представляет собой файл, в котором сервер накапливает информацию о событиях, произошедших на сервере, обычно (но не всегда) самых серьезных. Термин "журнал регистрации ошибок" немного сбивает с толку. Регистрацию ошибок можно настроить на таком "низком" уровне, что сервер будет фиксировать фактически любое событие. По умолчанию сервер записывает информацию об ошибках в файле error_log, расположенном в каталоге logs, который можно найти в каталоге ServerRoot[2].

```
$APACHE/logs/error_log
```

В ОС Unix имеется возможность задать пересылку сообщений об ошибках в системный журнал регистрации ошибок. Для этого в директиве ErrorLog необходимо задать syslog.

```
ErrorLog syslog
```

Если вам регистрация ошибок не нужна вообще, можно в качестве устройства регистрации ошибок задать /dev/null[3].

```
ErrorLog /dev/null
```

Нужно отметить, что эта команда не *отключает* регистрацию ошибок как таковую. Сервер выполняет процедуру сохранения сообщений об ошибках, просто они нигде не сохраняются. Лучшей альтернативой этому будет установка уровня регистрации ошибок на очень низком уровне.

Несмотря на то, что возможности отключить регистрацию ошибок нет, с помощью директивы ErrorLog можно задать адрес ее записи. Например, чтобы записывать все сообщения об ошибках в каталог /var/log/httpd.error_log, следует задать команду

```
ErrorLog /var/log/httpd.error_log
```

Кроме того, сервер Apache различает восемь уровней ошибок. Все они перечислены в табл. 7.1. С помощью этих уровней директива LogLevel определяет объем информации, которая будет записываться в журнал регистрации ошибок.

Таблица 7.1. Восемь уровней ошибок

Уровень регистрации	*Значение*
debug	Регистрируется любое самое малозначительное событие.
info	Информационное сообщение.
notice	Значительные, но не серьезные события.
warn	Предупреждение, вероятно важное.
error	Плохо, что-то нужно предпринимать.
crit	Ужасно. Немедленно предпримите что-нибудь.
alert	Авария.
emerg	Катастрофа.

[2] В главе 2 для ссылки на ServerRoot мы пользовались системной переменной UNIX $APACHE.

[3] Устройство /dev/null представляет собой специальное устройство ОС UNIX, которое используется как своеобразная универсальная мусорная корзина.

Директивой Log Level рекомендуется устанавливать уровень warn или error. Любая установка уровней ниже этих приведет к выводу массы самой тривиальной диагностики, что приведет к потере информации о действительно важных сообщениях.

7.3. Журнал регистрации обмена данных

Журнал регистрации обмена данных (или журнал регистрации доступа) сохраняет детальную информацию о переданной и полученной информации на сервере. Эта функциональная возможность обеспечивается модулем mod_log_config, который может быть усилен модулем mod_log_common.

В отличие от журнала ошибок, журналы обмена данными не обязательны. Процедура записи информации требует времени, поэтому для повышения производительности можно отказаться от ведения журналов передачи вообще. Однако они представляют очень удобный механизм анализа трафика и не используются только в редком случае.

7.3.1. Отдельные журналы для виртуальных узлов

Существует возможность создания отдельных журналов для виртуальных узлов. Достаточно просто задать определенный файл и/или задать определенный формат в директиве virtualHost. При отсутствии такой директивы информация о виртуальных узлах будет регистрироваться в том же файле и формате (если вообще будет отличаться), что и все остальные узлы.

7.3.2. Включение регистрации обмена данных: директива TransferLog

Директива TransferLog предназначена для определения местоположения файла регистрации обмена данных (абсолютный путь или путь относительно корневого каталога ServerRoot). По умолчанию файл регистрации обмена данных располагается в файле access.log в подкаталоге logs каталога ServerRoot.

```
TransferLog logs/access_log
```

По умолчанию запись о каждой передаче данных имеет 7 полей. Они перечислены в табл. 7.2.

Таблица 7.2. Информация, которая заносится в журнал обмена данными по умолчанию

Поле записи	*Значение*
host	IP-адрес или полное имя домена клиента.
ident	Идентификатор пользователя на удаленном клиенте. Чтобы была такая возможность, нужно установить директиву IdentityCheck, и клиент должен отвечать на запросы.
authuser	Идентификатор пользователя, предоставляемый для получения доступа к документам, защищенным паролем.
date	Дата и время запроса в формате "dd/mm/yy:hh:mm: ss zone",
request	Строка запроса, поступившего от клиента.
status	Код состояния, возвращенный клиенту.
bytes	Количество байтов, возвращенных клиенту.

7.3.3. Настройка формата журнала регистрации

Формат файла access.log настраивается директивой LogFormat. Сервер распознает множество переменных (перечень переменных приведен в табл. 7.3), которые предназначены для включения новых полей в файл регистрации передач. Эти переменные могут задаваться в произвольном порядке. Например, директива LogFormat задает формат, содержащий имя узла клиента (%h), время и дату запроса (%t) и количество возвращаемых байтов (%b).

```
LogFormat "%h %t %b"
```

Чтобы сделать журнал более читабельным, можно включить в выводимые строки наименование данных.

```
LogFormat "Host=%h Date=%t BytesReturned=%b"
```

Таблица 7.3. Переменные формата журнала регистрации

Переменная	*Значение*
%A	Локальный IP-адрес необходим для распознания среди виртуальных узлов.
	Удаленный IP-адрес.
	Заголовок Content-Length, содержащийся в ответе сервера.
	Переменная берется из переменных окружения сервера.
%f	Полный путь к запрошенному документу.
	Имя узла или IP-адрес клиента.
	Определяет заголовок HTTP входящего запроса.
%l	Имя пользователя удаленного клиента. Заметим, что для этого должна быть установлена директива IdentityCheck.
	Задает внутреннюю переменную, необходимую серверу Apache для обмена данными между модулями и ядром.
	Определяет заголовок ответа сервера (исходящий).
%P	Идентификатор порожденного процесса, обслуживающего запрос.
	Порт TCP, с которого поступил запрос клиента.
%r	Первая строчка запроса HTTP (определяет метод).
%s	Код состояния HTTP первоначального запроса.
	Результирующий код состояния HTTP первоначального запроса.
%T	Количество секунд, необходимых для обработки запроса.
%t	Дата и время запроса.
%u	URL, содержащийся в запросе пользователя.
%u	Определяет идентификатор удаленного пользователя распознанного запроса.

Окончание табл. 7.3

Переменная	*Значение*
%V	Имя пользователя в соответствии с директивой UseCanonicalName.
%v	Имя сервера в соответствии с директивой serverName.

7.3.4. Перенастройка журналов

Достаточно типичная ошибка, которая допускается начинающими пользователями сервера Apache заключается в том, что в регистрационные журналы записывается любая информация. Естественной реакцией пользователя на это будет удалить или перенести файл в предположении, что сервер Apache создаст новый файл и начнет все с начала. Это не так. Сервер Apache отслеживает с помощью внутренней переменной как далеко в файле находится следующее свободное пространство. Это значение, которое называется offset, не сбрасывается, когда файл, к которому оно относится, переносится или удаляется. Следовательно, сервер Apache начинает записывать новый файл точно в том же месте, в котором он должен был записывать старый. Единственным различием в такой ситуации станет то, что там, где старый файл содержит x символов регистрационной информации, новый файл будет содержать x символов произвольного мусора. Вероятно, это не совсем то, к чему следует стремиться.

Чтобы произвести регистрацию с самого начала файла, необходимо сначала перенести или удалить старый регистрационный файл, а потом перезапустить сервер по сигналу HUP.

```
kill -1 'cat http.pid'
```

7.4. Модуль mod_status

Модуль mod_status выводит результаты работы в двух форматах. В стандартном формате, который представляет собой счетную таблицу, выводимую в графике ASCII, она отображает информацию о порожденных процессах и статистику работы. Для получения более полной информации о состоянии порожденных процессов можно прибегнуть к помощи директивы ExtendedStatus. Следует помнить, однако, что применение директивы ExtendedStatus вызывает ощутимое снижение производительности. Потому этот вариант не из лучших.

7.4.1. Настройка сервера Apache для работы с модулем mod_status

Модуль mod_status не компилируется по умолчанию. Чтобы получить к нему доступ, сначала необходимо подключить его к работающему ядру. Более детальную информацию по компиляции сервера Apache и/или отдельных модулей можно найти в главе 2, "Инсталляция Web-сервера Apache"

После включения в ядро модуля mod_status вам потребуется активизировать его, предварительно связав с определенным каталогом. Напомним, что модуль mod__status генерирует результирующий файл в формате HTML. Чтобы получить к нему доступ, необходимо задать его расположение (с помощью директивы Location). Это приведет к запуску дескриптора server-status. А потому необходимо добавить в конфигурационный файл httpd.conf следующие строки:

```
<Location /server-status>
    SetHandler server-status
</Location>
```

Чтобы получить информацию о статусе сервера из броузера, достаточно задать каталог /server-status на вашем сервере. Это показано на рис. 7.1.

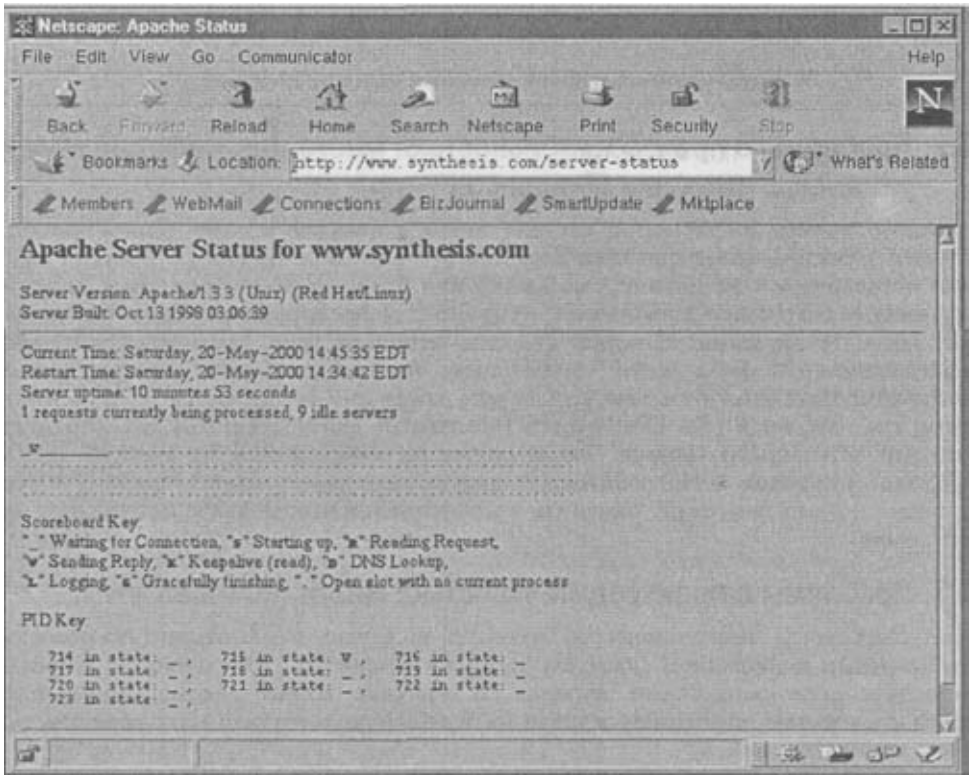

Рис. 7.1. Общая информация о состоянии сервера

Каждая позиция в счетной таблице представляет информацию о потенциальном порожденном процессе. Состояние порожденного процесса отображается определенным ASCII-символом. Они описаны в табл. 7.4.

Таблица 7.4. Счетная таблица

Символ *ASCII*	*Значение*
	Процесс не выполняется. Точка сохраняется для общности отображения количества параллельно выполняющихся процессов.
–	Процесс существует, но неактивен.
W	Процесс выполняет операцию записи.
L	Процесс выполняет операцию записи в журнал регистрации.
S	Процесс активизируется.
K	Процесс находится в состоянии ожидания запроса от клиента.
G	Порожденный процесс получил сигнал на плавное выключение. (Подробно об этом в главе 4, "Запуск, перезапуск и остановка")

Окончание табл. 7.4

Символ ASCII	*Значение*
R	Процесс выполняет операцию чтения запроса, поступившего от клиента.
D	Процесс идентифицирует доменное имя или IP-адрес.

7.4.2. Проблемы с производительностью: много идентифицируемых доменных имен или IP-адресов

Помимо прочей информации счетная таблица показывает, что много процессов находятся в стадии идентификации доменного имени или IP-адреса. В таком случае может потребоваться модифицировать одну или более конфигурационных директив. Ряд директив (например директива VirtualHosts) оперируют как доменными именами, так и IP-адресами. С одной стороны использование доменных имен делает конфигурационный файл более читабельным, но в этом случае серверу придется идентифицировать доменное имя всякий раз, когда оно упоминается. В зависимости от того, где находится ваш DNS-сервер (локальный или удаленный), это отражается на производительности сервера. Такая ситуация отображается на счетной таблице множеством символов D. Исправить сложившуюся ситуацию можно следующим образом: везде, где это возможно, заменить в конфигурационном файле доменные имена на IP-адреса.

7.4.3. Проблемы с производительностью: много пользователей

Ситуация, когда много процессов находятся на стадии регистрации (что отображено избыточным количеством символов L), сигнализирует о необходимости изменения размещения регистрационного журнала. В качестве общих рекомендаций можно предложить избегать записывать журнал на магнитную ленту или на устройства сетевой файловой системы.

7.4.4. Получение детальной информации о процессах

Как было сказано выше, модуль mod_status можно настроить таким образом, чтобы он генерировал более детализированную информацию об отдельных процессах. Иногда это временно необходимо для диагностирования возникших проблем с производительностью или других проблем. Однако, вполне можно ограничиться основной информацией, которую дает модуль mod_status, так как избыточная информация порождает перегрузку. Чтобы получить детальную информацию о процессе, необходимо установить директиву ExtendedStatus в состояние On. В результате имеется отчет, изображенный на рис. 7.2.

Заметим, что полученная распечатка включает описание столбцов.

7.4.5. Перезапуск сервера

После плавного перезапуска сервера Apache (например, с помощью сигнала USR1 в ОС Unix), статистика сервера не сбрасывается. Она сбрасывается при использовании сигналов HUP или TERM.

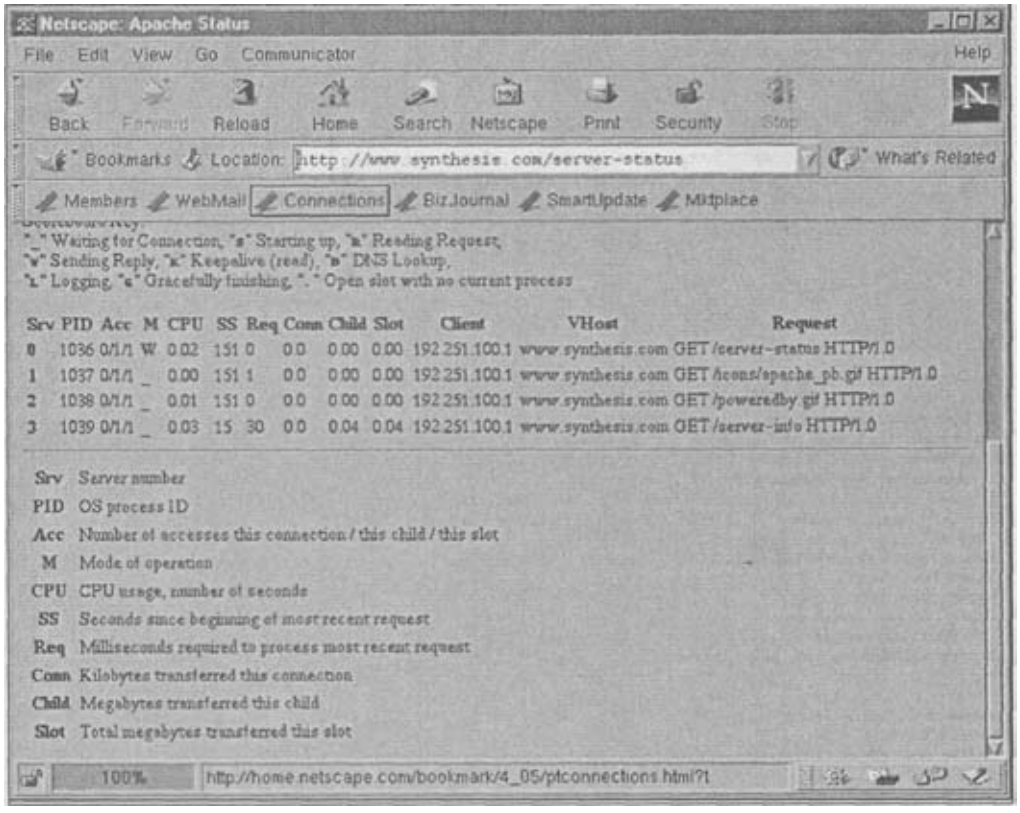

Рис. 7.2. Детализированная информация о состоянии сервера

7.5. Модуль mod_info

Модуль mod_info отображает информацию о текущей конфигурации сервера. Эта информация включает:

- Список скомпилированных модулей сервера.
- Конфигурационные директивы, влияющие на эти модули.

Как и в случае с модулем mod_status, необходимо убедиться, что модуль скомпилирован и активен. (Этот процесс аналогичен процессу, описанному в разделе "Настройка сервера Apache для работы с модулем mod_status", за исключением того, что в этом случае используется модуль mod_info.)

После подключения модуля mod_info к работающему серверу httpd, его необходимо настроить на обработку некоторых определенных URL, чтобы получить доступ к отчету. Например, директива

```
<Location /server-info>
    SetHandler server-info
</Location>
```

сообщает серверу Apache о том, что запросы для получения информации по серверу нужно пересылать в подкаталог server-info для обработки модулем mod_info. Теперь этот отчет можно получить, указав в броузере адрес, заданный директивой Location. В результате вы имеете достаточно объемный отчет, аналогичный изображенному на рис. 7.3.

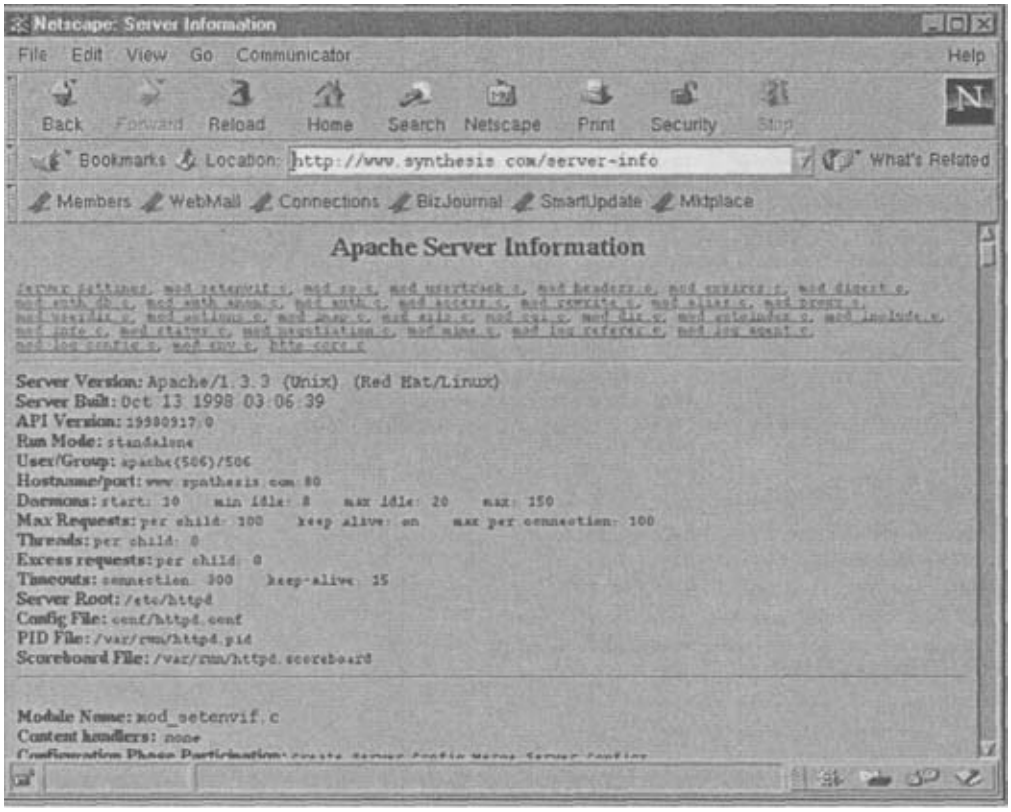

Рис. 7.3. Информация о сервере

Этот отчет занимает несколько экранов. Самая верхняя позиция содержит гиперсвязи, содержащие информацию об активных модулях. Ниже приводится распечатка общей информации о сервере с последующей информацией о модулях.

Подсказка

Чтобы модуль mod_info показывал представленную детальную информацию, пользователь, под управлением которого работает сервер httpd, должен иметь право на чтение конфигурационного файла.

7.5.1. Дополнительная информация о модулях

При необходимости отображения модулем mod_info **дополнительной информации (в текстовом формате или формате HTML), воспользуйтесь директивой** AddModuleInfo.

```
AddModuleInfo mod custom.c "<A HREF=/docs/mod custom/index.html>Notes</A>"
```

Глава

8

БЕЗОПАСНОСТЬ

В этой главе...

8.1. Введение

Независимо от задач, которые призван решать ваш узел, вопросы его безопасности (права доступа к каталогам, ограничение доступа <Directory> и т.д.) требуют особого внимания. Особенного внимания проблема безопасности требует при создании узла, предназначенного для электронной коммерции. Это гарантирует вам спокойную жизнь в будущем. Когда покупатель делает покупку товара с вашего узла с помощью кредитной карты, он доверяет вам важную информацию. Если при этом вами не было предпринято существенных мероприятий по укреплению безопасности, в результате чего был украден номер кредитной карты, вашу компанию скорее всего ожидает банкротство.

Именно поэтому необходимо уделить пристальное внимание вопросам безопасности. В контексте сервера Apache проблема *безопасности* является одной из приоритетных. Сначала будут рассмотрены проблемы безопасности на уровне операционной системы и способы предотвращения нескольких очевидных "дыр" в ее защите. Отдельные части главы посвящены проведению соответствующего исследования. Затем мы рассмотрим методы настройки сервера Apache для обеспечения максимальной безопасности. Этот раздел включает информацию о том, что *не следует* делать для того, чтобы повысить безопасность системы. Затем мы обсудим проблему санкционирования доступа пользователей. Существует множество механизмов идентификации пользователей. Одни из них гарантируют доступ на основании анализа источника запросов; другие базируются на схемах идентификации имени пользователя и его пароля. Наконец, будут обсуждены механизмы обеспечения безопасной передачи данных в Apache с применением протокола SSL (Secure Sockets Layer — протокол защищенных сокетов).

8.2. Указания по настройке

Без достаточно надежной операционной системы нет смысла предпринимать все остальные шаги для зашиты сервера. В этом разделе описываются некоторые основные этапы повышения безопасности работы сервера в операционной системе и настройки сервера Apache.

8.2.1. Безопасность каталогов

Сервер Apache обычно запускается с правами пользователя root. Так вот, **возможность** не только root-пользователей производить операции записи в каталоги, хранящие программы и конфигурационные файлы, создает пробел в системе зашиты. Убедитесь в том, что владельцем всех перечисленных ниже файлов является пользователь root, и что они не доступны никому кроме пользователя root:

```
$APACHE
$APACHE/bin
$APACHE/logs
$APACHE/conf
```

В частности, каталог, содержащий конфигурационные файлы, не должен быть доступен для записи пользователем apache.

8.2.2. Должна существовать четко сформулированная и опубликованная политика безопасности

Политика безопасности, изложенная на бумаге, поможет вам сконцентрировать мысли и пригодится при наказании нарушителей в соответствии с законодательством. Принципы этой политики должны быть четко и ясно сформулированы и опубликованы в вашей организации. Политика должна определять, кому разрешен доступ к вашим компьютерам, когда им разрешается осуществлять этот доступ и что разрешается делать в системе после регистрации.

8.2.3. Сервер должен работать на специально выделенном компьютере

Сервер не должен работать на машине, где производится разработка программного обеспечения. Если на ваш компьютер возложена только задача обработки Web-запросов, все остальные задачи с нее нужно убрать. Кроме того, чем меньше людей имеют возможность для регистрации на сервере, тем меньше возможности появления "дыр" в системе безопасности как случайных, так и преднамеренных.

8.2.4. Внимательно следите за новыми доработками

Фактически каждый продавец программного обеспечения как коммерческого, так и некоммерческого, периодически рассылает доработки, произведенные в связи с найденными ошибками и пробелами в системе безопасности. Раз в месяц необходимо посвятить несколько часов для загрузки новых доработок к операционной системе и основным программным пакетам. Обязательно ведите дневник обновлений.

8.2.5. Отказ в доступе

Стандартной политикой должна быть политика **отказа** в доступе, начиная с каталога root:

```
<Directory/>
     Order deny, allow
     Deny from all
</Directory>
```

По мере необходимости каталогам, находящимся под этим каталогом, можно разрешить доступ:

```
<Directory /opt/apache/htdocs>
      Order deny,allow
      Allow from all
</Directory>
```

8.2.6. CGI-сценарии

CGI-сценарии и программы уже сами по себе несут опасность, так как они **разрешают** произвольному пользователю запускать программы в вашей системе. Преднамеренно или нет, каждый новый сценарий может содержать ошибки. Вот несколько советов, как сделать работу с CGI-сценариями относительно безопасной.

- Обучить разработчиков системы и обучиться самому ситуациям, при которых возникают пробелы в системе зашиты. Например, всегда необходимо проверять размер длины строк, вводимых пользователями, для того, чтобы избежать использования злоумышленником ситуации переполнения буфера. Существует множество Web-узлов, которые специализируются исключительно на проблемах безопасности. Вот их краткий перечень:

  `http://www.lOpht.com`

  `http://www.privacy.org`

  `http://www.hideaway.net`

  `http://www.genocide2600.com`

- Никогда не загружайте программы из Internet, если нет четкого понимания их функций.
- Храните свои CGI-сценарии и программы в отдельном подкаталоге. Любое другое место кроет в себе возможность оплошности или ошибки. Существует вероятность возникновения такой ситуации, при которой вы не можете четко сказать, где находятся ваши программы, а не то, чтобы позаботиться об их безопасности.

8.2.7. PHP

Уязвимость ранних версий PHP коренилась в возможности переполнения буфера, что позволяло пользователям выполнять на локальном компьютере произвольные программы. Эта проблема была решена только в последних версиях. Таким образом, достаточно убедиться в том, что вы работаете с последней версией PHP.

8.2.8. Вставки на стороне сервера

Вставки на стороне сервера могут быть настроены таким образом, чтобы пользователи имели возможность запускать на сервере произвольные программы. Если такое возможно, то есть смысл отключить права на выполнение с помощью директивы Options.

```
Options IncludesNOEXEC
```

8.2.9. Отключение автоматического индексирования

Модуль mod_autoindex автоматически генерирует перечень содержимого каждого каталога в файле index.html. Эта информация используется взломщиками для поиска файлов, которые можно применить в личных целях. Скомпилируйте или отключите его с помощью директивы ClearModuleList.

8.2.10. Отключение прав пользователей

Опасность возникает, если вы позволите пользователям самим определять собственные права доступа. Кроме того, файлы .htaccess отрицательно влияют на производительность. По этой причине директивы AllowOverrides должны быть установлены в None.

```
AllowOverrides  None
```

8.2.11. Кодировка конфиденциальных данных

Независимо от того, насколько безопасна ваша система, конфиденциальные данные (информация о кредитных карточках, персональная информация) нельзя хранить на компьютере, доступном для внешнего мира. Я рекомендую, особенно в случае хранения информации о номерах кредитных карточек, удалять или кодировать все хранимые вами данные. Программное обеспечение, предназначенное для кодирования данных, можно получить бесплатно на Web-узле http://www.mit.edu/network/pgp.html.

8.3. Основы идентификации

Сервер Apache имеет несколько механизмов, позволяющих производить идентификацию пользователей на основании информации об их доменах или серверах. В этом разделе описаны процедуры настройки и принципы действия этих механизмов.

8.3.1. Идентификация по узлу: модуль mod_access

Модуль mod_acccess по умолчанию включен в стандартный дистрибутив. С его помощью реализован элементарный контроль доступа. Он заключается в возможности предоставлять права доступа на основании информации об узле, с которого поступает запрос на доступ. Узлы и домены могут задаваться как с помощью IP-адреса, так и с помощью имени.

8.3.2. Директива order

Директива order предназначена для определения порядка, в соответствии с которым сервер Apache будет оценивать директивы deny и allow. Допустимые значения перечислены в табл. 8.1.

Таблица 8.1. Значения директивы order

Значение	*Назначение*
order deny, allow	Используется для отказа в доступе большинству узлов и санкционирования доступа некоторым узлам.
order allow, deny	Используется для санкционирования доступа большинству узлов и отказа в доступе некоторым узлам.
order mutual-failure	Список разрешений узлов должен совпадать, т.е. для получения доступа узлы не должны находиться в списке узлов, которым отказано в доступе.

8.3.3. Директива allow

Эта директива предназначена для определения узлов или доменов, которым разрешен доступ к указанному каталогу.

Чтобы санкционировать доступ к каталогу /some/directory **всем, кто этого захочет, необходимо задать директиву.**

```
<Directory /some/directory>
      allow from all
</Directory>
```

Заметим, что разрешение доступа для всех может быть целиком или частично отменено директивами order и deny.

Для разрешения доступа к каталогу /some/directory **всем пользователям из домена** asgard.com **достаточно следующих команд:**

```
<Directory /some/directory>
      order deny, allow
      deny from all
      allow from .asgard.com
</Directory>
```

Санкционировать доступ можно на основании информации о подсети. Сервер Apache будет просматривать подсеть слева направо, начиная с самого левого октета. Например, чтобы разрешить доступ к каталогу /some/directory **всем членам подсети** 192.168.100, **необходимо указать:**

```
<Directory /some/directory>
      order deny, allow
      deny from all
      allow from 192168.100
</Directory>
```

Чтобы разрешить доступ к каталогу /some/directory **всем членам подсети 192.168,необходимоуказать:**

```
<Directory /some/directory>
      order deny, allow
      deny from all
      allow from 192.168
</Directory>
```

При определении подсети можно ограничиться ее маской (в этом случае 255.255.252.0).

```
<Directory /some/directory>
      order deny, allow
      deny from all
      allow from 153.168.242.0/255.255.252.0
</Directory>
```

Наконец, можно задать полное имя домена:

```
<Directory /some/directory>
      order deny, allow
      deny from all
      allow from fenris.asgard.com
</Directory>
```

или полный IP-адрес:

```
<Directory /some/directory>
      order deny, allow
      deny from all
      allow from 192.168.100.80
</Directory>
```

8.3.4. Директива allow from env

Этот вариант директивы allow позволяет определять права доступа на основании переменной окружения. Обычно это необходимо для взаимодействия с директивой BrowserMatch при санкционировании доступа на основании анализа типа броузера. Например,

```
BrowserMatch ^Mozilla netscape_yes
<Directory /opt/apache/htdocs >
        order deny, allow
        deny from all
        allow from env=netscape_yes
</Directory>
```

Кроме того, переменную окружения можно установить директивой SetEnvIf **для последующего использования** ее **в** директивах allow from env и deny from env.

8.3.5. Директива deny

Варианты синтаксиса переменной deny (IP-адрес, домен, частичный IP-адрес, частичный домен, слово *all)* подобны аналогичным конструкциям в директиве allow. Достаточно посмотреть на примеры, приведенные в отношении директивы allow.

8.3.6. Директива deny from env

Аналогично тому, как доступ может быть санкционирован на основании существования какой-то переменной окружения (см. директиву allow from env), в доступе может быть отказано на основании значения той же самой переменной окружения.

8.4. Идентификация по пользователю

В нашем распоряжении имеется много модулей, призванных санкционировать доступ, основываясь на имеющейся информации о пользователе. В общих чертах их функция заключается в запросе информации о комбинации имени пользователя и его пароле для последующей их проверки в какой-то обобщенной базе данных. Основное различие заключается в том, что принципы хранения пароля в разных модулях неодинаковы.

Независимо от выбранного метода идентификации директивы AuthName и AuthType всегда необходимы.

8.4.1. Обозначение запретной области: директива AuthName

Все механизмы идентификации используют директиву AuthName для указания каталога, доступ к которому требуется контролировать. Указанный вами текст будет включен во всплывающее окно, которое появляется на экране. Например, чтобы включить в окно слова "Restricted Site", воспользуемся следующей директивой:

```
AuthName "Restricted Site"
```

8.4.2. Ограничение доступа: директива require

Большинство из модулей управления доступом используют директиву require. Чтобы просто включить управление доступом на основании комбинации идентификатора пользователя и пароля, в этой директиве можно использовать ключевое слово *valid-user.*

```
require valid-user
```

С другой стороны, можно задать перечень пользователей, которым будет разрешен доступ (после проверки их прав).

```
require valerie joey bobby
```

Кроме того, можно реализовать доступ пользователям по принадлежности к определенной группе пользователей. Для этого необходимо с помощью директивы AuthGroupFile создать группу и разрешить доступ пользователям по принадлежности к определенной группе. Предположим, что Valerie, Joey и Bobby известны вместе как группа mathdept. Если это так, то директива

```
require mathdept
```

будет иметь действие, аналогичное предыдущему примеру. Как задать группу, можно посмотреть в описании директивы AuthGroupFile, которое следует ниже.

8.4.3. Определение метода идентификации: директива AuthType

Директива AuthType имеет один параметр, определяющий тип идентификации (basic или digest). Параметр basic (основной) означает, что пароль будет передан от клиента на сервер в виде текстовой информации. Очевидно, что при этом возникает угроза безопасности. Такая идентификационная информация может быть перехвачена на пути следования.

Другой вариант идентификации — digest (цифровой) — позволяет воспользоваться кодировкой MD5 для шифровки передаваемых пакетов. К сожалению, его использование ограничено тем, что этот тип идентификации поддерживается не всеми типами броузеров.

Чтобы задать первый тип идентификации, можно воспользоваться директивой:

```
AuthType basic
```

8.4.4. Модуль mod_auth

Модуль mod__auth включен в стандартный дистрибутив по умолчанию. Он санкционирует или запрещает доступ на основании информации, хранящейся в стандартных текстовых файлах.

8.4.5. Создание файла идентификации с помощью команды htpasswd

Перед использованием модуля mod_auth необходимо создать файл паролей таким образом, чтобы у модуля была информация, с которой можно сравнить регистрационные данные. Стандартный дистрибутив сервера Apache включает исполняемый файл htpasswd, функции которого заключаются только в этом. Вот синтаксис работы с ним:

```
htpasswd -c passwordfile username
```

Здесь опция -c означает, что требуется создать новый файл паролей. Аргумент username является обязательным. Программа htpasswd работает в интерактивном режиме. Она выдает подсказку о необходимости ввода и проверки пароля:

```
/etc/security> htpasswd -c local__passwd userguy
Adding password for userguy.
New password:
Re-type new password:
/etc/security>
```

Введенный пароль будет зашифрован с помощью функции crypt() и сохранен вместе с текстовым именем пользователя в указанном вами файле паролей. Если нужно добавить только нового пользователя в уже существующий файл паролей, опцию -c можно пропустить.

8.4.6. Включение режима контроля доступа: директива AuthUserFile

После того как файл паролей был создан, необходимо проинформировать сервер Apache, где его можно найти. Эту функцию берет на себя директива AuthUserFile. В ней задается единственный параметр, указывающий абсолютный путь к пользовательскому файлу, созданному утилитой htpasswd.

```
AuthUserFile  /etc/security/local_passwd
```

8.4.7. Контроль за групповым доступом: директива AuthGroupFile

Для уменьшения списка пользователей в директиве require можно воспользоваться возможностью санкционирования группового доступа. Это файл, строки которого содержат имя группы, двоеточие, за которым следует перечень пользователей, разделенный запятыми. Для того, чтобы сервер Apache знал о том, что файл grouplist размещается в каталоге /etc/security, воспользуемся следующей директивой:

```
AuthGroupFile  /etc/security/.grouplist
```

Теперь вместо перечня имен пользователей в директиве require можно воспользоваться именами групп, перечисленными в этом файле.

8.4.8. Передача управления модулю нижнего ранга: AuthAuthoritative

В случаях, когда модуль mod_auth отказывает в доступе, существует возможность предоставить это сделать другим работающим модулям санкционирования доступа и предоставить им право проверки их баз данных на предмет наличия прав доступа конкретного пользователя. Если директива AuthAuthoritative отключена, для идентификации пользователя избирается иной модуль санкционирования доступа, но уже более низкого ранга[1].

```
AuthAuthoritative  off
```

8.4.9. Все виды контроля одновременно: пример модуля mod_auth

Напомним, что все методы санкционирования доступа требуют использования директив AuthName и AuthType и что модуль mod_auth также требует некую форму директивы require. Вот пример, в котором доступ к узлу www.site2.com будет ограничен для всех пользователей, кроме тех, кому доступ разрешен.

```
<Directory  /home/site2>
        AuthName "Example  of Access  Control"
        AuthType Basic
        AuthUserFile  /etc/security/.htpasswd
        Require  valid-user
</Directory>
```

[1] Ранг определяется порядком следования в конфигурационном файле.

Напомним, что путь /home/site2 для узла www.site2.com является каталогом DocumentRoot. Когда пользователи пытаются получить доступ к узлу www. site2.com, они видны на экране, изображенном на рис. 8.1.

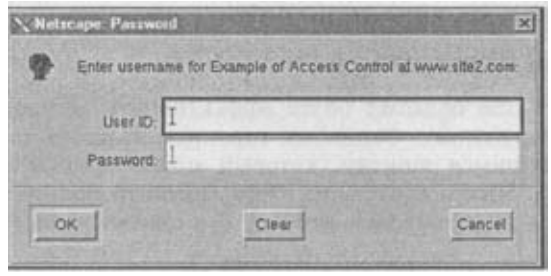

Рис. 8.1. Экран управления доступом модуля mod_auth

Пользователь получает доступ к узлу после ввода правильного идентификатора и пароля. В противном случае пользователь увидит экран, аналогичный экрану, изображенному на рис. 8.2.

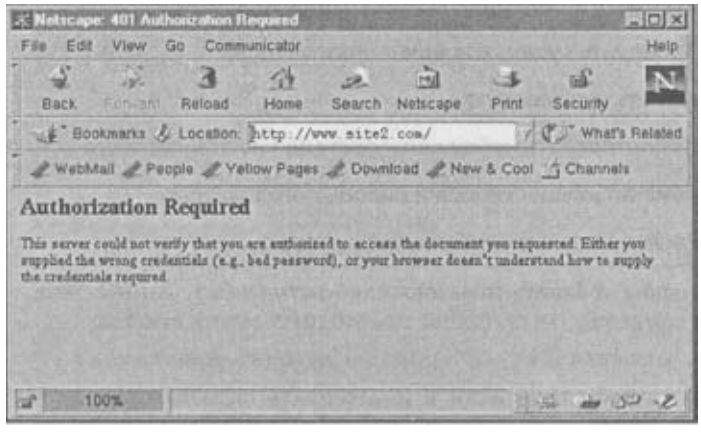

Рис. 8.2. Стандартное сообщение об ошибке

Замечу, что экран, который вы видите на этом рисунке, является стандартным экраном, сообщающим о неудачной попытке доступа. Его можно видоизменить с помощью директивы errordocument.

```
ErrorDocument 401 /site_error.html
```

8.4.10. Модуль mod_auth_dbm

Как отмечалось ранее, основное различие между модулями mod_auth_* заключается в способе хранения пароля. Напомним, что модуль mod_auth использует в своей работе простой текстовый файл, хранящий имена пользователей и закодированные пароли, известный на профессиональном жаргоне как *плоский файл*. Этот метод хранения предполагает, что сервер Apache производит поиск файла по дереву сверху вниз всякий раз, когда требуется идентифицировать пользователя. Такой метод вполне подходит для осуществления поиска в небольших базах данных, но будет значительно влиять на производительность, если количество идентифицированных пользователей превысит определенный показатель.

Просто существует какой-то предел, после которого все уже зависит от типа используемой вами базы данных, но в нашем случае количество пользователей не будет превышать пары сотен.

8.4.11. Создание индексированной базы данных: директива dbmmanage

Модуль mod_auth_dbm обладает более эффективным методом поиска в больших пользовательских базах данных. Основным отличием является то, что здесь применяется индекс по хранящимся записям, который может значительно сократить время поиска в базе данных. Чтобы извлечь из этого какую-то пользу, необходимо с помощью утилиты dbmmanage создать базу данных. Вот синтаксис вызова этой утилиты:

```
dbmmanage filename [command] [username],
```

где command — одна из команд перечисленных в табл. 8.2.

Таблица 8.2. Команды утилиты dbmmanage

Команда	*Назначение*
adduser	Добавить в базу данных нового пользователя.
check	Проверить существование пользователя.
delete	Удалить пользователя.
import	Импортировать набор пользователей из стандартного устройства ввода.
update	Изменить существующий пароль пользователя.
view	Распечатать содержимое базы данных.

Например, чтобы добавить пользователя *usertwo* в базу данных dbm, находящуюся в каталоге /etc/security/httpdbase, необходимо задать команду

```
dbmmanage /etc/security/httpdbase adduser usertwo
```

После этого потребуется ввести и подтвердить пароль, который был вами выбран для этого пользователя.

8.4.12. Включение режима контроля доступа с помощью базы данных DBM: директива AuthDBMUserFile

После создания базы данных DBM необходимо указать ее место нахождения для сервера Apache. Абсолютный путь к базе данных DBM можно задать с помощью директивы AuthDBMUserFile.

```
AuthDBMUserFile /etc/security/httpdbase
```

8.4.13. Директива Auth_Dbm_Authoritative

Директивой AuthDbmAuthoritative в качестве последней инстанции в управлении доступом для определенного каталога задается модуль mod_auth_dbm.

```
AuthDbmAuthoritative on
```

8.4.14. Директива AuthDbmGroupFile

Как и mod_auth, модуль mod_auth_dbm имеет механизм объединения наборов пользователей в группы. Файл групп DBM содержит список пользователей, разделенных запятыми.

8.4.15. Модуль mod_auth_db

Еще одним из идентификационных модулей является модуль mod_auth_db. Он работает с DB-файлами стандарта Беркли и содержит информационные пары пользователь-пароль. Если ваша система не поддерживает стандарт DBM, он может быть единственным возможным вариантом. Директивы, связанные с этим модулем, работают точно таким же образом, как директивы модуля mod_auth_dbm, и не требуют дополнительных объяснений. Приведем простой перечень.

Директива	***Назначение***
AuthDBUserFile	<абсолютный путь к DB-файлу>
AuthDBGroupFile	<абсолютный путь к групповому файлу>
AuthDBAuthoritative	<on|off>

8.4.16. Модуль mod_auth_anon

Модуль mod_auth_anon представляетдругой метод управления доступом к вашему узлу, хотя и не такой мощный. Он предоставит доступ, но при этом потребует сначала ввести определенную информацию. Основная идея заключается в том, что потенциальный пользователь вводит некое обобщенное имя пользователя, обычно "anonymous", и предоставляет в качестве пароля адрес электронной почты. Ужесточить требования к вводимым адресам электронной почты практически нет возможности. Удовлетворяет всякая текстовая строка, содержащая в качестве разделителей символы "@" и ".". Таким образом, в определенной степени вы находитесь во власти пользователей. Многие из них достаточно воспитаны, чтобы просто подделать электронный адрес, однако модуль mod_auth_anon имеет еще ивозможность регистрации независимо от содержания напечатанного текста.

8.4.17. Определение действующих пользователей: директива Anonymous

Как упоминалось ранее, обобщающее имя пользователя, под которым возможен анонимный доступ, является имя "anonymous". Менее популярное, но время от времени появляющееся имя "guest". Установка анонимного доступа для пользователя с другим именем, вероятно, сильно огорчит завсегдатаев Internet. Чтобы гарантировать анонимный доступ псевдоидентификаторам "anonymous" и "guest", необходимо ввести:

```
Anonymous anonymous guest
```

8.4.18. Регистрация доступа: директива Anonymous_LogEmail

Обычный этикет анонимных визитов заключается в том, что визитер должен предоставить вместо пароля свой электронный адрес. Для этого задайте директиву Anonymous_LogEmail.

```
Anonymous_LogEmail on
```

8.4.19. Как заставить пользователей вводить какую-нибудь информацию при регистрации: директива Anonymous_MustGiveEmail

Чтобы заставить пользователя при регистрации вводить электронный адрес вместо пароля, установите значение этой директивы в *on*. Однако прислушайтесь к совету: нет способа гарантировать, что пользователь ввел реальные или хотя бы правильно отформатированные данные.

```
Anonymous_MustGiveEmail on
```

8.4.20. Патетическое извинение по поводу проверки формата: директива Anonymous_VerifyEmail

Чтобы проверить, что пользователем были введены как минимум символы "@" и ".", воспользуйтесь директивой.

```
Anonymous_VerifyEmail on
```

8.4.21. Директива Anonymous_NoUserID

Мне кажется, что принудительный ввод пустого имени пользователя несет в себе больше вреда, чем пользы. Если вы считаете, что можно без этого обойтись и доверяете фальшивому электронному адресу, то обратитесь за помощью к директиве Anonymous_NoUserID.

```
Anonymous_NoUserID on
```

8.4.22. Директива Anonymous_Authoritative

Установка этой директивы в состояние on соответствует ситуации, когда для рассматриваемого каталога возможно применение другого механизма управления доступом. Последнее слово в управлении доступом остается за модулем mod_auth_anon.

```
Anonymous_Authoritative on
```

Чтобы дать возможность другим модулям идентифицировать пользователей, необходимо указать

```
Anonymous_Authoritative off
```

8.5. Протокол SSL

Протоколы передачи данных TCP/IP создавались с целью обеспечения надежного механизма передачи данных. В этом направлении разработчики создали достаточно интеллектуальные механизмы маршрутизации, способные из множества путей из точки А в точку В выбрать наиболее оптимальный в данный момент путь. Следовательно, не существует способа предсказать маршрут, которым передаваемые пакеты достигнут пункта назначения. С точки зрения безопасности вполне можно предположить, что все пакеты накапливаются и собираются произвольным количеством умных негодяев, лелеющих надежду приобрести стимуляторы или детскую порнографию за ваши деньги. Совершенно очевидно, что такое положение вещей неприемлемо.

Ранее было замечено, что протокол HTTP не имеет серьезного механизма безопасной передачи данных. Для обеспечения определенного уровня безопасности при передаче информации с броузера пользователя на сервер лучшим вариантом является протокол Secured Sockets Layer или SSL. Протокол SSL был создан компанией Netscape Corporation как средство поддержки электронной коммерции. В нем применяется система шифрования

данных, известная под термином *шифрование с открытым ключом*. Она позволяет броузеру закодировать данные таким образом, что сервер сможет их легко расшифровать. Теоретически существует возможность расшифровки таких данных методом перебора всех возможных ключей по порядку в случае перехвата данных злоумышленником. Но, принимая во внимание время, необходимое для этого (а на это могут потребоваться недели), такой метод не имеет практического смысла.

Возможность безопасной передачи информации по мировой сети является критическом моментом для электронной коммерции и может пригодиться для многих других Web-приложений. Может возникнуть вопрос, почему возможность работы с протоколом SSL не включена в стандартный дистрибутив Apache. Такая ситуация сложилась в результате позиции правительства США по этому вопросу. Во времена холодной войны было принято законодательство, запрещающее экспорт шифровальных технологий за пределы США. Даже в пределах США публикация статей или книг, посвященных шифровальным технологиям, была ограничена. Во времена учебы в колледже (1987-1993) я взялся за написание научной статьи, посвященной проблеме шифрования, и был немало удивлен, обнаружив, как мало было доступной информации, посвященной данной теме. Помогли только архивы университетской библиотеки[2]. В 1994 году мною была найдена маленькая книжка по криптографии, изданная примерно в 1950 году в одном сельском колледже в штате Джорджия.

Несколькими месяцами позднее я с удивлением столкнулся с популярным теперь изданием *"Прикладная криптография" Брюса Шейнера ("Applied Cryptography" Bruce Schemer)*. Очевидно, что при том общественном интересе, который поднялся на волне развития Internet, больше невозможно было держать такие книги на полке[3]. Несмотря на то, что свобода печати распространяется на все, что напечатано на бумаге (включая твердые копии криптографического программного обеспечения), она не затрагивает само программное обеспечение. В Соединенных Штатах передача криптографического программного обеспечения за рубеж по-прежнему является преступлением, и, как видно по делу *Фила Циммермана (Phil Zimmerman)*[4], нарушителей этого закона ожидает очень жесткое наказание. Кроме всего прочего, некоторые механизмы, примененные в протоколе SSL, запатентованы. Потому включение протокола SSL в дистрибутиве Apache может привести к серьезным противоречиям с законом.

8.5.1. Шифрование с открытым ключом

Шифрованием называется процесс преобразования последовательности читабельных данных в нечитабельные. Обычно в этой процедуре задействуется одна или более служебных строк данных, которые называются *ключами*. Метод шифрования с открытым ключом кодирует данные с помощью одного ключа, скажем ключа A, таким образом, что она потом может быть расшифрована другим ключом, скажем, ключом B.

[2] Совершенно серьезно. Ни статей, ни книг, ничего даже похожего на обычные ссылки. Смутно припоминаю, что я прочитал в статье, посвященной стандарту шифрования данных (который в те времена использовался банками при кодировке фондовой информации для передачи ее в электронном виде), что Управление национальной безопасности препятствовало публикации материалов такого типа. К сожалению, мне не удалось найти первоисточник, на который тогда делалась ссылка, и теперь даже не могу сказать с полной уверенностью, читал я это или это мне показалось. Если вам встречалась такая информация, напишите, пожалуйста, мне об этом.

[3] Вот интересный факт, касающийся этой книги. *"Прикладная криптография" ("Applied Cryptography")* является бестселлером № 13 в Чешской Республике и № 6 в Венгрии, следуя в местном рейтинге сразу же за изданием *"Руководство современной домохозяйки по обогащению ядерного топлива" ("The Modern Housewife's Guide to Refining Fissionable Material")*.

[4] *Фил Циммерман* разработал PGP и разместил его в Internet, предоставив возможность копировать иностранцам.

В свою очередь данные, зашифрованные с помощью ключа B, могут быть расшифрованы только с помощью ключа A.

Один из этих ключей доступен всем (это и есть *"открытый ключ"),* в то время как другой ключ хранится в строгом секрете (это *"секретный ключ").* Желающие общаться с вами безопасно, получают ваш открытый ключ и пользуются им для шифрования своих сообщений. Получив такое зашифрованное сообщение, вы с помощью секретного ключа расшифровываете сообщение и, вероятно, шифруете свой ответ.

8.5.2. Коммерческое использование протокола SSL

Вероятно, прочитав следующие несколько разделов, вы решите, что у вас есть более важные дела на сегодня. Это так, можно уделить внимание и другому материалу. Преодолевая легальные барьеры, поставленные на пути получения сервера Apache, оснащенного SSL, *бесплатно,* помните, что это совсем нетрудно сделать, уплатив определенную сумму.

Распространители коммерческого клона Linux недавно приступили к продаже как серверной, так и настольной версии своего программного обеспечения. Серверная версия (стоит в пять раз дороже) обычно снабжается версией сервера Apache, уже оснащенной протоколом SSL. Честно говоря, сам не пробовал, но есть подозрения, что они работают отлично. Самая дешевая версия для ОС Linux Caldera. Она стоит около 80 долларов. Это один из лучших вариантов.

Кроме того, на рынке действуют компании, которые продают коммерческие версии сервера Apache.

- **Компания Stronghold.** Предлагает самые последние версии сервера Apache, снабженного отличным инструментарием настройки, протоколом SSL и поддержкой программного обеспечения. Дополнительную информацию можно получить по адресу *http://www.c2.net.*
- **Компания Raven.** Продает двоичный код сервера Apache, снабженный протоколом SSL. Дополнительную информацию можно получить по адресу *http://www.covalent.net.*

8.5.3. Модуль mod_ssl

Оценив имеющиеся на рынке коммерческие предложения, можно попробовать реализовать SSL собственноручно. Модуль mod_ssl для сервера Apache представляет собой бесплатную реализацию SSL. Этот модуль можно загрузить с Web-узла *http://www.modssl.org.*

Зайдите туда и загрузите дистрибутив. Чтобы использовать все возможности модуля mod_ssl, необходимо иметь библиотеку SSL. Их тоже можно свободно загружать с узла *http://www.openssl.org.*

После получения этих компонентов, скопируйте их в определенный каталог, например каталог /opt (предназначенный обычно для стороннего программного обеспечения).

```
cp openssl-X_Y_Z_tar.gz /opt
tar xvfz openssl-X_Y_Z_tar.gz
```

Дистрибутив включает также файл INSTALL, содержащий инструкции по компилированию исходных текстов и по их размещению. Последовательность команд компиляции сервера без протокола SSL имеет вид:

```
./config
make
make test
make install
```

и разместит полученные библиотеки в стандартном каталоге /usr/local/ssl. После того как библиотеки скомпилированы, необходимо извлечь сам модуль mod_ssl. При этом будем исходить из того, что набор файлов находится в корневом каталоге сервера Apache (но, в принципе, их можно разместить где угодно).

```
cp  mod_ssl-A_B_C-X_Y_Z_tar.tar  /opt/apache
tar  xvfz  mod_ssl-A_B_C-X_Y_Z_tar.tar
```

Дистрибутив модуля mod_ssl, существующий в настоящий момент, в процессе построения новой утилиты полностью перекомпилирует и переинсталлирует сервер Apache. Детали этого процесса можно найти в файле INSTALL, который только что был вами распакован. Однако в зависимости от размещения текущих версий различных продуктов он выглядит следующим образом:

```
./configure \
--with-apache=../apache_1.3.x\
--with-ssl=../openssl-0.9.x \
--with-rsa=../rsaref-2.0/local \
--with-mm=../mm-1.0.x  \
--with-crt=/path/to/your/server.crt \
--with-key=/path/to/your/server.key \
--prefix=/path/to/apache \
```

В результате будет создан новый Makefile (для Apache) в каталоге, который был задан как /path/to/apache. Не стоит волноваться, если у вас еще нет сертификата — в процессе инсталляции, можно получить временный сертификат. Перейдите в каталог, определенный вами как /path/to/apache, и введите группу команд

```
make
make certificate
make install
```

Первая команда make перестраивает Apache таким образом, что он теперь включает модуль mod_ssl. Вторая команда позволит создать тестовый сертификат, а последняя команда действительно установит новую версию сервера Apache с модулем mod_ssl.

8.5.4. Сертификация

Сертификатом является способ информирования Web-общественности о том, что вы тот, за кого себя выдаете. Для проверки можно сгенерировать свой собственный сертификат в соответствии с описанной выше процедурой. Однако, несмотря на то, что вами используется коммерческий подписанный сертификат, большинство броузеров клиентов в момент начала сеанса связи SSL будут отображать некое предупреждение.

Для расшифровки данных используется ключ, предоставленный сертификационным органом. Если клиент запрашивает защищенные данные, вы отвечаете следующей информацией:

- Сертификат вашего сервера, подписанный каким-либо известным сертификационным органом.
- Некое простое текстовое сообщение.

Клиент использует открытый ключ, полученный у сертификационного органа для того, чтобы расшифровать сообщение, поступившее с сервера. Расшифрованное сообщение содержит ваш открытый ключ. Затем вы пересылаете сообщение клиенту, зашифрованное с помощью вашего секретного ключа. Сообщение содержит данные, зашифрованные с помощью вашего открытого ключа. С помощью открытого ключа, полученного на первом этапе, сервер расшифровывает сообщение, посланное вами. Он расшифровывает текстовое сообщение и сравнивает результат с содержанием второго сообщения. Если они совпада-

ют, клиент приходит к выводу, что вы есть тот, за кого себя выдаете, и с вашим сервером можно продолжать общение.

Коммерческий сертификат можно приобрести на узлах *http://www.verisign.com* и *http://www.thawte.com*.

8.5.5. Применение протокола SSL

Предположим, что вы разрабатываете коммерческий узел, который будет иметь имя *www.securesite.com*. В этом разделе показано, как с помощью директив модуля mod_ssl можно сделать передачу данных на этот узел безопасной.

Обычно протокол SSL используется в контексте виртуального узла. Начнем с использования директив NameVirtualHost и <VirtualHost>. Они позволяют осуществлять виртуальный хостинг по имени. Это предоставит в наше распоряжение готовый объект реализации политики безопасности (более подробно о виртуальном хостинге можно прочитать в главе 5, "Хостинг нескольких Web-узлов").

```
NameVirtualHost 64.82.73.226
<VirtualHost 64.82.73.226>
     ServerName www.securesite.com
     DocumentRoot /home/site3
</VirtualHost>
```

8.5.6. Актуализация протокола SSL: директива ssLEngine

Чтобы использовать SSL, его необходимо актуализировать. Для такой цели обратимся к директиве SSLEngine.

```
SSLEngine on
```

8.5.7.Определениесертификата:директива SSLCertificateFile

Следующий шаг — необходимо сообщить SSL, где можно взять файл сертификата, который был получен предварительно. Этот файл должен храниться в безопасном месте на диске. Задайте абсолютный путь с помощью директивы SSLCertificateFile.

```
SSLCertificateFile  /var/ssl/server.crt
```

8.5.8. Определение ключа:директива ssLCertificateKeyFile

Не следует хранить ваш сертификат и ключ в одном файле, хотя технически это возможно. Полагая, что секретный ключ, который используется для расшифровки, не включен в указанный файл сертификата, необходимо также указать местоположение файла, содержащего ключ сертификата. Это можно сделать с помощью директивы SSLCertificateKeyFile.

```
SSLCertificateKeyFile /var/ssl/server.key
```

8.5.9. Директива SSLCACertificatePath

Эта директива задает путь к сертификату, полученному в сертификационном органе.

```
SSLCACertificatePath  /some/secure/directory
```

8.5.10. Директива SSLCACertificateFile

Эта директива задает путь к файлу, содержащему сертификат, полученный в сертификационном органе.

```
SSLCACertificateFile  /path/to/certificate
```

8.5.11. Запуск регистрации: директива ssLLog

Протокол SSL имеет свой собственный механизм регистрации. Очевидно, это очень удобно для решения проблем, неизбежно возникающих во время запуска, и его рекомендуется оставлять в рабочем состоянии. Директива SSLLog предназначена для указания места размещения этого файла как абсолютного пути, а в случае отсутствия в начале пути символа обратной косой черты — относительно ServerRoot. Заметим, что сама по себе эта директива не включает режим регистрации SSL (см. SSLLogLevel ниже).

```
SSLLog  /var/log/ssllog
```

8.5.12. Определение уровня регистрации SSL: директива SSLLogLevel

Как и сервер Apache, протокол SSL имеет множество уровней регистрации. Детально они описаны в табл. 8.3.

Таблица 8.3. Уровни регистрации SSL

Уровень регистрации	Значение
none	Регистрация не производится.
error	Регистрируются только фатальные сообщения.
warn	Подходит для ежедневного использования, но необходимо регулярно очищать журнал.
info	Подходит для отслеживания процесса отладки.
trace	Вероятно, это больше информации, чем вам нужно.
debug	Подробные и скучные детали.

Для более полного отображения появляющихся проблем без создания при этом избыточной информации рекомендуется устанавливать уровень warn.

```
SSLLogLevel warn
```

В этом случае сообщения будут выглядеть примерно следующим образом:

```
NameVirtualHost 64.82.73.226
<VirtualHost 64.82.73.226>
     ServerName www.securesite.com
     DocumentRoot /home/site3
     SSLEngine on
     SSLCertificateFile /var/ssl/server.crt
     SSLCertificateKeyFile /var/ssl/server.key
     SSLLog /var/log/ssllog
     SSLLogLevel warn
</VirtualHost>
```

8.5.13. Директива SSLVerifyClient

Эта директива необходима для определения политики относительно сертификатов клиентов. Параметры могут иметь следующие значения: 0 (не требуется), 1 (факультативно) или 2 (обязательно).

```
SSLVerifyClient 1
```

8.5.14. Директива SSLVerifyDepth

Одни сертификационные органы иногда практикуют ссылки на другие при проверке подлинности сертификатов. Эта переменная устанавливает количество связей, которые будут отслеживаться. Основное предназначение директивы заключается в проверке сертификатов клиентов. В противном случае стандартное значение менять не следует.

```
SSLVerifyDepth 3
```

Глава

9

ДИНАМИЧЕСКИЕ WEB-СТРАНИЦЫ

В этой главе…

9.1. Введение

В основной своей массе Web-узлы и узлы электронной коммерции используют в своей работе динамическое содержимое. Традиционные Web-страницы являются статическими: их вид не зависит от внешних переменных — от времени, от того, кто получает к ним доступ, и т.д. *Динамическое содержимое* является общим понятием для Web-страниц, генерирующих одно или все возможные отображения во время доступа к таким страницам.

Сервер Apache имеет два основных механизма генерации динамического содержимого. На стороне сервера — это метод вставок, который заключается в том, что процесс httpd, просматривающий исходящий HTML-код на наличие ключевых слов SSI, замещает их переменными окружения или результатами работы вызванных программ. CGI-интерфейс является протоколом, позволяющим серверу Apache запустить такую программу, как сценарий Perl (Practical Extraction and Report Language) или двоичный код, созданный на языке программирования C, и передать полученный результат в HTML-коде на броузер клиента.

В этой главе мы детально рассмотрим шаги, необходимые для создания сервером Apache динамического содержимого. Однако следует отметить, что несмотря на то, что эта глава изобилует примерами кода CGI и SSI, ее нельзя рассматривать в качестве учебника по искусству создания динамических Web-страниц.

9.2. Вставки на стороне сервера (SSI)

Вставки на стороне сервера (SSI) представляют собой механизм добавления динамического содержимого ограниченного объема к Web-страницам. В соответствии с этим методом часть или все содержимое, имеющееся в наличии на сервере, маркируется как потенциально содержащее код SSI, включенный в код HTML. При обслуживании страницы сервер Apache сначала просматривает ее на наличие ключевых слов SSI и вносит изменения в соответствии с обнаруженными командами в передаваемый на броузер пользователя результат.

9.2.1. SSI и производительность сервера

Однако вставки на стороне сервера по умолчанию отключаются. Основной причиной этого является падение производительности, вызываемое использованием этой технологии. Совершенно очевидно, что сервер Apache, который вынужден просматривать все передаваемое содержимое или только часть его, будет работать значительно медленней, чем сервер просто передающий страницы, при этом их не просматривающий. По этой причине рекомендуем ограничить действие SSI как можно меньшим количеством файлов и/или каталогов.

9.2.2. Включение режима SSI

Возможность вставки на стороне сервера обеспечивается модулем mod_include. Этим модулем будет обработан любой документ, имеющий дескриптор serve r-parsed. Кроме того, любой документ MIME-типа text/x-server-parsed-html или text/x-server-parsed-htmlS в целях обратной совместимости будет проанализирован модулем mod_include.

Модуль mod_include по умолчанию включен в стандартные дистрибутивы сервера Apache. Однако если его у вас еще нет, сервер Apache придется полностью перекомпилировать.

Режим SSI включается как опция. Чтобы задействовать режим SSI, воспользуйтесь директивой:

```
Options +Include
```

Немного более безопасным методом включения режима SSI является директива includeNOEXEC, которая дает команду серверу Apache не запускать сценарии.

```
Options +IncludeNOEXEC
```

Чтобы запустить режим SSI в работу, достаточно воспользоваться одной из указанных директив. Кроме того, если директива AllowOverride установлена в попе, режим SSI включить нельзя.

Как сказано выше, из соображений повышения производительности будет лучше разделить Web-страницы с SSI и без SSI. Есть два способа это сделать. Первое решение — выделить отдельный каталог, который содержит только файлы, имеющие SS1. Второе решение — они должны иметь расширение, присущее только им.

Функция SSI является опцией, а опции действуют только в ограниченных пределах (например файл .htaccess, директива <Loc ation> или директива <Directory>), ее действие можно эффективно ограничить определенным каталогам. В качестве такового можно указать каталог DocumentRoot и, таким образом, фактически сделать функцию SSI глобальной.

9.2.3. Ограничение SSI по расположению

Диапазон действия директивы может быть ограничен парой директив <Location>, или директив <Directory>, или размещением его в файле .htaccess, находящемся в этом каталоге.

```
<Location /ssidir>
     Options +Include
</Location>
```

9.2.4. Первый вариант ограничения режима SSI по расширению файла: директива AddHandler

Еще одним способом ограничения количества файлов, просматриваемых сервером Apache до того, как они будут обслужены, является указание нового расширения файлов, содержащих ключевые слова SSI. Например, чтобы ограничить SSI-вывод в вашей системе только файлами с расширением .shtml, можно прибегнуть к помощи директивы AddHandier.

```
AddHandier server-parsed .shtml
```

Эта директива сообщает серверу, что перед передачей страниц пользователям нужно просматривать файлы с расширением .shtml и делать соответствующие замены директив SSI. Конечно использовать именно расширение .shtml не обязательно, но такой метод является наиболее традиционным.

Уместно напомнить, что директиву AddHandier можно применять только к SSI-файлам. Это общий метод подключения типов файлов к дескрипторам внутреннего содержимого на сервере Apache. Эта директива опять встретится нам при обсуждении CGI-интерфейса.

9.2.5. Второй вариант ограничения режима SSI по расширению файла: директива AddType

Связав расширение .shtml с соответствующим внутренним дескриптором, необходимо сообщить всем броузерам клиентов, что это было сделано. Броузеры ожидают информацию о типе получаемого содержимого. Директива AddType ассоциирует расширение файла с типом содержимого, переданного броузеру клиента. В следующем примере серверу дана команда сообщать всем клиентам, что файлы, имеющие расширение .shtml, содержат данные типа text/html.

```
AddType text/html .shtml
```

С другой стороны, директиву AddType можно использовать для полного обхода директивы AddHandier. Например:

```
AddType application/x-server-parsed .shtml
```

9.2.6. Определение элементарных SSI: директива XBitCrack

Единственной директивой модуля mod_include является директива XBitCrack. Если ее установить в on, это будет означать, что сервер рассматривает все документы, исполняемые в файловой системе и имеющие MIME-тип text/html как SSI-документы.

```
XBitCrack on
```

Помимо значений off и on, директива XBitCrack имеет опцию fall, задающую серверу режим вставки заголовка Last-Modified при передаче файла. Это позволяет proxy—серверу производить кэширование.

9.3. Листинг вставок

В этом разделе перечислены SSI-вставки и приведены примеры их использования.

9.3.1. Установка опций SSI: команда config

Вставка команды config позволяет определить следующие три момента:

- errmsg является текстовым сообщением, передаваемым клиенту в случае возникновения ошибки в процессе синтаксического анализа SSI.

- sizefmt [bytes|abbrev] **задает формат отображения размера.**
- timefmt задает **формат строки, используемый при отображении календарных** значений.

<!--#config errmsg="Ошибкавозникла во время разборки SSI"-->

9.3.2. Отображение конфигурационных переменных: команда echo

Команда echo предназначена для отображения значения переменной окружения.

<!--#echo var="$APACHEDIR"-->

9.3.3. Запуск сценария: команда exec

Эта команда используется для запуска программы из страницы SSI. Такая программа может быть исполняемым кодом. В этом случае используется параметр cmd:

<!--#exec cmd="/bin/date"-->

С другой стороны, это может быть сценарий CGI. В таком случае используется параметр cgi:

<!--#exec cgi ="/cgi-bin/program.pl"-->

Отметим, что разрешение запуска сценариев CGI несет в себе определенный риск для системы безопасности, и на это не стоит идти безкрайней необходимости.

9.3.4. Отображение размера файла: команда fsize

Для отображения размера файла в соответствии с форматом, заданным командой config sizefmt, предназначена команда fsize.

```
<!--#config sizefmt="bytes"-->
<!--#fsize file="/bin/date" -->
```

9.3.5. Отображение времени последней модификации файла: команда flastmod

Для отображения времени последней модификации файла в формате, заданного командой config timefmt, необходимо указать:

```
<!--#config timefmt="%m %d %h:%m"-->
<!--#flastmod file="/bin/whatever"-->
```

9.3.6. Условное выполнение: команды if и elif

SSI-код имеет простейшее (элементарное) управление выполнением. Для выполнения некоего оператора по определенному условию возьмите его в операторные скобки if. Возможно также выполнение оператора elif.

```
<!--#if expr="$var1 = true" -->
Var1 is true
<!--#elif expr="$var1 = false" -->
Var 1 is false
<!--#endif" -->
```

9.3.7. Отображение других файлов: команда include

Для отображения в качестве вывода текущего документа содержимого других файлов (при этом указывается абсолютный путь) используется директива include:

```
<!--#include file="/etc/motd"-->
```

или задается имя узла:

```
<!--#includevirtual="http://www.some.com"-->
```

9.3.8. Отображение списка всех переменных окружения: команда printenv

Эта команда предназначена для отображения сразу всех переменных окружения (это очень удобно при отладке).

```
<!--#printenv -->
```

9.3.9. Изменение значения переменной: команда set

Команда set предназначена для создания переменных и присвоения им значений.

```
<!--#set var="v1" value="the value" -->
```

9.4. Интерфейс CGI

CGI является общим интерфейсом отображения результатов работы программ. Формат CGI не зависит от языка программирования. С этим интерфейсом может работать любая программа или сценарий из существующего на данный момент инструментария программирования, начиная с достаточно "древнего" языка программирования COBOL и до языка написания сценариев Perl, генерирующего код HTML. Однако на практике одни языки лучше подходят для генерирования CGI-содержимого, чем другие. В частности, язык Perl более подходит для обработки текстовых строк, содержащих HTML-код.

9.4.1. Переменные окружения CGI

Программам CGI необходим метод определения того, кто их вызвал, и что они должны делать. Для этого сервер Apache устанавливает переменные окружения, указанные в табл. 9.1.

Таблица 9.1. Переменные окружения CGI

Переменная	*Назначение*
PATH_INFO	Относительный путь к запрошенному ресурсу.
PATH_TRANSLATED	Абсолютный путь к запрошенному ресурсу.
QUERY_STRING	Параметры, которые передаются клиенту.
REMOTE_HOST	Устанавливается только в случае, если сервер не был скомпилирован с установками MINIMAL_DNS.
REMOTE_IDENT	Устанавливается только в случае, если ключ IdentityCheck установлен в on.
REQUEST_METHOD	Метод вызова сценария (GET, POST).
REMOTE_USER	Устанавливается только в том случае, когда сценарий CGI является объектом санкционирования.
SCRIPT_NAME	Имя исполняемого модуля.

9.4.2. Настройка сервера Apache

Использование CGI-сценариев требует определенных настроек сервера. Простого сценария или программного файла, который будет создавать HTML-код, здесь недостаточно. Сервер должен знать, что такой файл выполнится, и будет возвращен какой-то результат, а не просто отработать безрезультатно. Для этого есть два пути.

- Задается место на сервере, где хранятся исполняемые файлы.
- Дескриптор CGI-содержимого связывается с файлами определенного типа.

9.4.3. Включение режима CGI: директива Options +ExecCGI

Независимо от MIME-типа или каталога размещения сервер даже не пробует выполнить файл по умолчанию. Необходимо только явным образом задать выполнение CGI-содержимого до того, как отработают директивы ScriptAlias, Set-Handler или AddHandler. Это можно сделать с помощью директивы Options. Например, последовательность команд

```
<Directory /usr/local/cgi-bin>
        Options +ExecCGI
<Directory>
```

задает режим обработки CGI-содержимого, находящегося в каталоге /usr/local/cgi-bin. Символ "+", стоящий перед опцией ExecCGI, сообщает серверу, что к списку уже действующих в этом каталоге опций необходимо добавить опцию ExecCGI. При отсутствии символа "+" сервер будет считать, что опция ExecCGI является единственной опцией, действие которой распространяется на этот каталог.

Обратите внимание, что перед тем, как запустить какую-либо обработку, необходимо дать знать серверу, что файлы, находящиеся в этом каталоге (или файлы заданного типа) являются CGI-сценариями. Более подробно это описано в следующих разделах.

9.4.4. Определение расположения файла сценария: директива ScriptAlias

Директива ScriptAlias является функцией модуля mod_alias. Как и директива Alias, она позволяет задать каталог для хранения файлов, отличный от каталога DocumentRoot. Но если директива Alias только пересылает запросы клиента в указанный каталог, то директива ScriptAlias маркирует содержимое каталога как исполнимое.

Например каталог ServerRoot на вашем сервере находится в дереве /opt/apache. По соглашению CGI-сценарии и программы хранятся в подкаталоге cgi-bin. Из соображений безопасности может потребоваться хранить все CGI-программы в другой файловой системе. Директива

```
ScriptAlias /cgi-bin/ /usr/local/cgi-bin/
```

будет выбирать все запросы из каталога

```
/opt/apache/cgi-bin
```

и пересылать их в каталог

```
/usr/local/cgi-bin
```

Найдя файл, сервер Apache скорее всего выполнит его или, по крайней мере, попытается выполнить, возвращая результат, а не обрабатывая файл как таковой. Поэтому директиву Alias можно использовать подобным образом и промаркировать исполняемые файлы с помощью директив SetHandler и/или AddHandler. Но, с другой стороны, централизация CGI-файлов делает конфигурационные файлы более читабельными.

9.4.5. Маркировать целые каталоги как исполняемые: директива SetHandler

Возможно, что вследствие какой-то причины вы не захотите использовать директиву ScriptAlias, тогда все файлы каталога с помощью директивы SetHandler можно промаркировать как исполняемые. Эта директива может быть задана как в скобках <Directory> или <Location>, так и помещена в файл .htaccess. Это сообщает серверу Apache, что все файлы, находящиеся в этом каталоге, можно рассматривать как исполняемые. Например:

```
<Directory /usr/local/cgi-bin>
    SetHandler cgi-script
</Directory>
```

9.4.6. Определение дескриптора по расширениям файлов: директива AddHandler

Вот еще один метод, позволяющий серверу Apache распознавать различные типы файлов как сценарии. Это метод ассоциации внутреннего дескриптора с расширением файла. С концепцией дескрипторов можно ознакомиться в главе 1, "Основные концепции". Например, вашему вниманию представлена директива, которая сообщает серверу Apache, что файлы с расширениями .pl или . cgi необходимо рассматривать как CGI-сценарии, и они должны быть обработаны (что, в данном случае, фактически означает — выполнены) дескриптором CGI-сценариев, который имеется в модуле mod_cgi.

```
AddHandler cgi-script .pl .cgi
```

9.4.7. Задание исполняемого MIME-типа: директива AddType

Файлы, которые сервер Apache распознает как исполняемые файлы MIME-типа, выполняются автоматически. Тип, связанный с файлами сценариев, application/x-httpd-cgi. Пусть все файлы, содержащие сценарии имеют расширение .pl. Чтобы ассоциировать их с соответствующим MIME-типом, необходимо задать директиву

```
AddType application/x-httpd-cgi .pl
```

Данные по MIME-типам можно найти в главе 1, "Основные концепции".

9.4.8. Отладка CGI: директива ScriptLog

То, что CGI-сценарии не работают, клиент определяет по отсутствию ответа на свой запрос. Это делает отладку CGI-сценариев более проблематичной. Однако модуль mod_cgi дает возможность записывать любое диагностическое сообщение, созданное CGI-сценарием, в регистрационный файл. Для настройки данного процесса в нашем распоряжении имеется три директивы:

- ScriptLog. Когда эта директива задана, диагностические сообщения, посылаемые CGI-сценариями, записываются в заданный каталог. Если директива не задана, ошибки не диагностируются. Файлы регистрации ошибок могут быть заданы как с указанием абсолютного пути, так и относительно корневого каталога сервера.

  ```
  ScriptLog /var/logs/cgilog
  ```

- ScriptLogBuffer. Чтобы ограничить размер регистрационного файла, эта директива ограничивает объем регистрируемых значений PUT или POST. Значение по умолчанию составляет 1024 байт.

  ```
  ScriptLogBuffer 256
  ```

- ScriptLogLength. Эта директива ограничивает размер файла, в который записываются диагностические сообщения об ошибках. По достижению предельного размера дальнейшие ошибки не регистрируются. Размер задается в байтах. Значение, принимаемое по умолчанию, равно 10385760.

  ```
  ScriptLogLength 20000000
  ```

9.5. Управление потреблением ресурсов

В зависимости от интенсивности использования вашими узлами CGI-ресурсов, процессор может оказаться сильно перегруженным созданием динамического вывода. Даже один плохо написанный сценарий может монополизировать ресурсы всей системы. Дополнительно к возможности управления на уровне операционной системы сервер Apache имеет три директивы. Они ограничивают объем диагностической информации, которую оставляют программы CGI.

- Директива RLimitCPU задаетверхний предел времени в секундах, на протяжении которого порожденный CGI-процесс может использовать центральный процессор.

  ```
  RLimitCPU 20
  ```

- Директива RLimitMEM задает максимальный объем оперативной памяти, который может быть выделен для порожденного процесса.

  ```
  RLimitMEM 10000000
  ```

- Директива RLimitNPROC задает верхний предел количества порожденных процессов, которые могут запускаться одновременно.

  ```
  RLimitNPROC 200
  ```

9.5,1. Модуль mod_perl

Язык написания сценариев Perl, вероятно, является одним из наиболее удачных из когда-либо созданных инструментов обработки текстовых строк, каковыми являются HTML-коды. CGI-сценарии в своем большинстве написаны на языке Perl. Если в планы входит обрабатывать сценарии, созданные на языке Perl, обязательно нужно рассмотреть возможность использования модуля mod_perl.

Модуль mod_perl, разработанный Дугом **Мак-Ичерном** (Doug MacEachern), является полнофункциональным интерпретатором языка программирования Perl, который реализован в виде модуля Apache. Из-за большого размера он не включен в стандартный дистрибутив, но его можно просто загрузить с Web-узлов группы Apache. Как это делается, можно посмотреть в главе 12, "Состав модуля".

Выигрыш в производительности при работе с CGI-сценариями, который можно получить от использования модуля mod_perl, основан натом, что этот модуль полностью включен в программу httpd. Это снижает количество обращений к диску и уменьшает создание одновременно работающих процессов.

При необходимости использования mod_perl для обеспечения работы CGI-программ, следует воспользоваться следующей конфигурацией:

```
<Location /perl>
      SetHandler peri-script
      PerlHandler Apache::Registry
      Options +ExecCGI
</Location>
```

9.6. Модуль FastCGI

CGI-сценарии сильно влияют на производительность. Так как CGI-процессы исполняются отдельно от процесса httpd, они автоматически завершают работу после своего запроса. Нередко один и тот же пользователь можетделать запрос к одному и тому же сценарию несколько раз во время одного и того же сеанса, вызывая тем самым издержки запуска этого сценария (обращение к диску, процессорное время).

Для решения этой проблемы и предназначен модуль FastCGI. При работе этого модуля CGI-сценарий кэшируется вместо того, чтобы постоянно загружаться и запускаться. Кроме того, модуль FastCGI расширяет переносимость сценариев FastCGI, которые теперь могут запускаться на любом сервере, где работает модуль FastCGI.

9.6.1. Загрузка модуля FastCGI и его инсталляция

Модуль FastCGI можно загрузить бесплатно с узла *http://www.fastcgi.com*. Распакуйте его обычным способом.

```
tar xvzf mod_fastcgi_X.X.X.tar.gz
```

Чтобы скомпилировать программу httpd **с модулем** mod_fastcgi, **сначала перенесите исходный каталог, который только что был распакован вами, в каталог** /src/modules/fastcgi.

```
mv mod_fastcgi.X.X.X /opt/apache/src/modules/fastcgi
```

Затем в каталоге /opt/apache перезапустите сценарий ./configure со следующей опцией:

```
--activate-module=/src/modules/fastcgi/libfastcgi.a
```

Перестройте и переинсталлируйте сервер Apache.

```
make
make install
```

Чтобы скомпилировать модуль mod_fastcgi как разделяемый объектный файл (рекомендуется), сначала надо распаковать каталог. Затем перейдите в этот каталог и перекомпилируйте программу с помощью утилиты apxs.

```
apxs -o mod_fastcgi.so -c *.c
```

Инсталлируйте скомпилированную программу.

```
apxs -i -a -n fastcgi mod_fastcgi.so
```

Убедитесь, что файл httpd.conf содержит нужную нам строку.

```
LoadModule libexec/mod_fastcgi.so
```

Кроме того, при использовании директивы ClearModuleList **для определения модулей необходимо применить команду** AddModule mod_fastcgi:

```
AddModule mod_fastcgi.c
```

После того как дескриптор FastCGI создан, его можно использовать (fastcgi-script) вместо стандартного дескриптора CGI (cgi-script). Однако, во избежание возможных неприятностей, определение дескриптора необходимо заключить в операторные скобки IfModule.

```
<IfModule mod_fastcgi.c>
        AddHandler fastcgi-script .fcgi
</IfModule>
```

Модуль FastCGI позволяет создавать сценарии, которые запускаются вместе с сервером и постоянно находятся в оперативной памяти. Для загрузки сценария в оперативную память во время запуска в нашем распоряжении имеется директива FastCgiServer.

```
FastCgiServer  /opt/apache/cgi-bin/handy.fcgi
```

А еще сценарии FastCGI можно запускать с помощью утилиты suexec, гарантирующей временные привилегии суперпользователя. Чтобы пользоваться такой возможностью, ее необходимо запустить следующим образом:

```
FastCgiSuexec on
```

9.6.2. Взаимодействие между процессами: директива FastCgiIpcDir

Сервер Apache взаимодействует со сценариями FastCGI с помощью сокетов. Сокетом является программная конструкция, которая используется в сетевых соединениях. Она также применима для взаимодействия в пределах одного компьютера. По умолчанию каталогом, где эти сокеты хранятся, является каталог /tmp/fcgi, но с помощью директивы FastCgilpcDir эту стандартную установку можно изменить.

```
FastCgilpcDir  /some/directory
```

Глава

10

НАСТРОЙКА РАБОЧИХ ХАРАКТЕРИСТИК СЕРВЕРА

В этой главе...

10.1. Введение

На протяжении всего повествования автор постоянно обращал внимание читателя на то влияние, которое будет иметь любая настройка на рабочие характеристики сервера. Однако совершенно очевидно, что на практике ваши системы достаточно хорошо оснащены, чтобы обращать внимание на все мои увещевания и предостережения в бешенной гонке за оптимизацией основных функциональных характеристик. Целью этой главы является посильная помощь в исправлении некоторых ошибок.

10.1.1. Использование утилиты vmstat

Хорошо известная утилита vmstat (virtual memory status — состояние виртуальной памяти) является основным средством диагностирования дефицита памяти. Виртуальная память представляет собой пространство на жестком диске, зарезервированное операционной системой для обработки ситуаций, при которых случается переполнение оперативной памяти. Виртуальная память эффективно увеличивает объем и количество процессов, которые одновременно может обрабатывать сервер. Однако это происходит за счет производительности. Необходимо убедиться в том, что память используется эффективно, если используется вообще.

Наиболее известным инструментом наблюдения за использованием виртуальной памяти в ОС Unix является утилита vmstat. Варианты этой утилиты, работающие на различных платформах ОС Unix (частности синтаксиса можно узнать в справочнике man) различаются, но вот такой листинг можно назвать типичным:

```
procs                  memory      swap      io system      cpu
r  b  w   swpd  free  buff  cache  si  so  bi  bo  in  cs  us  sy  id
0  0  0  19076  8704  13940  30740  0   0   1   1   108 45  0   0   99
```

В этой распечатке столбцы si и so отражают количество страниц памяти, закачанных/выкачанных в/из виртуальной памяти. Обратите внимание, что оба показателя равны 0, что, собственно, и требуется. Это показывает, что физической памяти для хранения процессов сервера и порожденных ими процессов вполне достаточно. Допустим незначительный трафик обмена данными между памятью и областью подкач-

ки, но большое количество операций обмена между памятью и областью подкачки свидетельствует о возникновении узкого места.

10.1.2. Настройка httpd

Уже говорилось о том, что размер программы httpd можно регулировать в процессе компиляции, ограничивая количество включаемых модулей только самыми необходимыми. Перечень модулей, включенных в программу, можно получить командой

```
httpd -l
```

Начинающие администраторы не могут устоять перед соблазном сразу инсталлировать программу, полностью избегая тем самым применения процесса компиляции. Это стремление вполне объяснимо, но оно может повлечь осложнения с точки зрения настройки эффективной работы сервера в будущем:

1. Вероятно, все возможности стандартной установки вам просто не нужны.
2. Если заблаговременно не освоить процесс компиляции, вы как администратор лишитесь одного из наиболее эффективных средств управления рабочими характеристиками сервера. Процесс компиляции с помощью менеджера конфигурации apaci достаточно прост. Примеры можно найти в главе 2, "Инсталляция Web-сервера Apache".

Очевидным выигрышем, который можно получить от создания оптимальной программы httpd заключается в том, что чем меньше отдельные экземпляры программы httpd, тем большее их количество сможет разместиться в имеющейся в вашем распоряжении памяти. Это позволяет обработать большее количество запросов от пользователей одновременно.

10.1.3. Активные серверы

Сервер Apache имеет несколько директив, позволяющих управлять количеством экземпляров сервера в определенный период времени. Эти директивы перечислены в табл. 10.1.

Таблица 10.1. Директивы, контролирующие количество экземпляров сервера

Директива	*Описание*
startservers	Число порожденных экземпляров процесса httpd, созданных в момент запуска. Эта директива неимеет большого значения, так как ее действие будет быстро перекрыто значениями MaxSpareServers и MinSpareServers.
MaxSpareServers	Максимальное число одновременно порожденных процессов сервера. Если это число будет превышено, процесс httpd, являющийся родителем, немедленно станет удалять один или более экземпляров httpd до тех пор, пока количество процессов не будет равно заданному значению.
MinSpareServers	Минимальное число одновременно простаивающих порожденных процессов сервера.
MaxClients	Это значение устанавливает верхний предел количества процессов, одновременно активных в системе. Он перекрывает значения, установленные директивами Max- и MinSpareServers. Увеличение стандартного значения, равного 256, требует изменения переменной HARD_SERVER_LIMIT и перекомпиляции сервера Apache.

Окончание табл. 10.1

Директива	*Описание*
MaxRequestsPerChild	Эта директива предназначена для предотвращения перерасхода памяти. Она удаляет порожденные экземпляры сервера (освобождая тем самым ресурсы) после того, как указанное количество запросов было обработано. Во всех случаях должно быть задано достаточно большое значение (>1000) (значение 0 предназначено для неопределенных запросов).

10.1.4. Файлы .htaccess

В режиме работы с файлами .htaccess сервер Apache просматривает установки, сделанные в файле .htaccess не только в текущем каталоге, но и в каталогах, находящихся выше по дереву вплоть до корневого каталога. Следствием этого является увеличение обращений к диску, а они, как мы уже видели, приводят к существенному снижению производительности. Поэтому при отсутствии острой необходимости использование файлов .htaccess лучше отключать.

```
AllowOverride None
```

В противном случае для ограничения области поиска файла .htaccess можно воспользоваться одной из операторных скобок (<Directory> или <Location>). Тогда по-прежнему потребуется обращение к диску, но это будет справедливо только для ограниченной области.

10.1.5. Определение DNS

Неразумно сплошь и рядом в конфигурационных файлах пользоваться символическими именами узлов. Но всегда существует соблазн сделать это. Особенно при создании пробного варианта системы, так как текстовая информация становится более понятной. А символические имена воспринимаются значительно легче, чем IP-адреса.

Процесс определения доменного имени очень медленный. Он обычно требует одного или более подключения к сети, которое устанавливается при каждой ссылке на символическое имя в конфигурационном файле. Так, каждый раз запрос рассматривается на DNS-сервере, и есть большая вероятность, что дополучения ответа на запрос он будет передан еще не на один DNS-сервер.

В частности, необходимо избегать соблазна использовать символические имена в директивах, список которых приведен в табл. 10.2.

Таблица 10.2. Директивы, не принимающие символические имена

Директива	*Описание*
allow, deny	Пользуйтесь только IP-адресами.
HostNameLookups	Должен быть установлен в off.
ProxyBlock, NoProxy, NoCache	Пользуйтесь только IP-адресами.

10.1.6. Регистрация

Процесс регистрации дважды виноват: в том, что ему требуется непропорционально большое количество процессорного времени и много обращений к диску. Поэтому, вероятно, будет разумным задать максимально низкий уровень регистрации.

```
LogLevel Error
```

Заметим, что если отключить эту возможность полностью или установить ее на минимальный уровень, существует риск потери действительно важной диагностической информации.

10.1.7. Расширенная информация о состоянии

При установке директивы ExtendedStatus модуля mod_status в on сервер Apache будет делать запись подробной информации о запросах пользователей. В некоторых ситуациях это может пригодиться, например при необходимости анализа трафика, но это отрицательно влияет на работу сервера в целом. При отсутствии настоятельной необходимости все-таки лучше эту возможность отключать.

10.1.8. Кэширование

Кэшированием называется принцип хранения proxy-сервером копий запрошенных документов на протяжении определенного периода времени, который сокращает тем самым количество обращений в сеть при получении запросов к наиболее часто запрашиваемым документам. При установке сервера в режим proxy (см. главу 6, "Proxy-серверы и кэширование") можно порекомендовать включить режим кэширования.

10.1.9. Включение режима KeepAlives

По умолчанию сервер Apache будет сохранять соединение активным при многочисленных HTTP-запросах. Это уменьшает перегрузку, вызванную частыми подключениями к сети и отключениями от нее. Такая возможность упоминается здесь только для полноты картины. Не углубляясь в доказательства, просто рекомендуем убедиться в том, что у вас нет команды:

```
KeepAlive off # это ПЛОХО. НЕ делайте так никогда.
```

10.1.10. Наблюдение за использованием ресурсов CGI-процессами

Сервер Apache имеет три директивы, которые позволяют существенно ограничить вред от использования CGI-сценариев (RLimitCPU, RLimitMEM, RLimitNPROC). Детали можно узнать в главе 9, "Динамические Web-страницы".

10.1.11. Загрузка наиболее общих файлов в память: модуль mod_mmap_static

Модуль mod_mmap_static использует возможность размещения в памяти, имеющуюся в некоторых вариантах Unix. С помощью единственной директивы этого модуля, директивы MMapFile, сервер Apache будет знать, что наиболее часто вызываемые модули в момент загрузки могут помещаться в память. Это снижает задержки, вызванные обращениями к диску во время сеанса работы. Однако это преимущество применимо только по отношению к статическим файлам. В частности, не стоит пытаться загружать CGI-сценарии с помощью этой директивы.

```
MMapFile    /opt/apache/htdocs/index.html
```

Модуль mod_mmap_static также включен в стандартный дистрибутив, его несложно подключить к серверу.

10.2. Проблема производительности в ОС Windows

Некоторые из перечисленных директив, таких как MaxClients и StartServers, не имеют отношения к платформе Windows, в которой реализован единственный, но многонитевой порожденный процесс, обрабатывающий все пользовательские запросы. На этой

платформе аналогичного результата можно добиться с помощью директивы ThreadsPerChild, так как число запросов пользователей, которые могут быть обработаны одновременно на платформе Windows, привязано к числу нитей в порожденном процессе. Директива ThreadsPerChild в Windows является функционально аналогичной директиве MaxClients, имеющейся на платформе Unix. Чтобы ограничить число одновременных подключений 128-ю подключениями, задайте следующую директиву:

```
ThreadsPerChild 128
```

Глава

11

ПЕРЕНАЗНАЧЕНИЕ АДРЕСА

В этой главе...

11.1. Введение

Модуль mod_rewrite позволяет перезаписывать полученные запросы на основании шаблонов переменных окружения и даже данных, выбранных из баз данных. Гибкость и мощь этого модуля потребовали включения его в стандартный дистрибутив Apache, но он не загружается по умолчанию из-за большого размера.

Чтобы загрузить модуль mod_rewrite, достаточно включить в конфигурационный файл следующие строки:

```
AddModulemod_rewrite.c
LoadModule rewrite_module libexec/mod_rewrite.so
```

Из-за своей сложности модуль mod_rewrite заслуженно имеет дурную репутацию. Но при том, что он считается одним из самых сложных модулей Apache, по степени своей сложности он ушел недалеко от, скажем, программы sendmail или агрессивной неопределенности лицензий на коммерческое программное обеспечение. Вероятно, самым сложным этапом изучения поведения mod_rewrite является постижение правил чтения и записи регулярных выражений. Если вы знакомы с ними, то это не составит большого труда. Если нет — советую для краткого обучения обратиться к приложению А, "Основные директивы".

Подсказка

Функциональность модуля mod_rewrite можно локализировать. С помощью стандартных ограничителей, таких как `Directory` или `VirtualHost`, ее можно ограничить или задать в конфигурационных файлах. Например в файле .htaccess.

11.2. Модуль mod_rewrite

Я думаю, что будет совершенно справедливо сказать о модуле mod_rewrite как о модуле, имеющем три составляющих. Об этом сейчас идет много споров, но мне представляется, что директивы, перечисленные в табл. 11.1, являются основными директивами модуля mod_rewrite.

Таблица 11.1. Основные директивы модуля mod_rewrite

Директива	*Назначение*
RewriteEngine	Включение/отключение режима перезаписи URL
RewriteRule	Задать правило перезаписи URL по шаблону.
RewriteCond	Задать предварительное условие, которое должно соблюдаться перед перезаписью URL.

11.2.1. Запуск модуля mod_rewrite: директива RewriteEngine

Директива RewriteEngine используется для включения и отключения режима перезаписи URL. Для включения режима необходимо задать следующую директиву:

```
RewriteEngine on
```

11.2.2. Перезапись URL по шаблону: директива RewriteRule

Директива RewriteRule занимает в модуле mod_rewrite центральное место. Она имеет два параметра. Это регулярное выражение, которое является правилом, и новое значение. Сама по себе директива достаточно проста, но синтаксис параметров шаблона и замещения довольно замысловат. Директива RewriteRule имеет следующий синтаксис:

```
RewriteRule pattern substitution [flag],
```

где параметр flagможет принимать одно из значений, указанных в табл. 11.2.

Таблица 11.2. Значения параметра flag директивы RewriteRule

Флаг	*Действие*
R, redirect	Инициировать внешнее переназначение.
F, forbidden	Инициировать ответ HTTP 403 (запрещено).
G, gone	Инициировать ответ HTTP 410 (запрос прошел).
P, proxy	Инициировать замену для внутреннего proxy. Для этого нужен модуль mod proxy.
L, last	Остановить процесс перезаписи.
N, next	Перезапустить процесс перезаписи, начиная с первого правила, но не с первоначального URL, и сохр аняя при этом все, что было сгенерировано до этого момента.
C, chain	Связать текущее правило в одну цепь с последующим правилом.
T=mtyp	Инициировать использование указанного MIME-типа.
NS, nosubreq	Пропустить это правило с помощью внутреннего подзапроса.
QS A, qsappend	Присоединить строку запроса к заменяемой строке, а не замещать ее.
PT, passthrough	Включить обработку результатов перезаписи модуля mod_rew rite другими модулями перезаписи (например модулем mod_alias).

Следующие три примера иллюстрируют различные способы проверки соответствия шаблону в порядке возрастания их сложности. Во всех примерах в качестве основного URL будем рассматривать *www.examplel.com.* Помните также, что для того, чтобы директива RewriteRule работала эффективно, сначала необходимо директивой RewriteEngine **активизировать модуль** mod_rewrite.

Пример 11.1.

Для того, чтобы модуль mod_rewrite перезаписывал (фактически перемаршрутизировал) запросы вида
wvw.example1.com/obsolete
на
www.example1.com/newandimproved,
необходима следующая директива:
RewriteRule A/obsolete.html/newandimproved.cgi

Пример 11.2.

Предположим, что возникла необходимость провести обслуживание файловой системы, имеющей DocumentRoot, не прерывая при этом работы сервера. Чтобы перенаправить весь трафик на узле в резервный каталог /apache_backup, задайте следующее правило:

```
RewriteRule ^/$ /apache backup [R]
```

Пример 11.3.

Предположим, что на узел поступают запросы с
www.examplel.com/~someguy
и их необходимо переписать в каноническом формате URL. Этот пример предполагает, что в файловой системе под каталогом DocumentRoot существует каталог /usr. Следующее правило
RewriteRule ^~ ([^/I+)/?(.*> /usr/$1/$2 [R]
: перепишет URL из приведенного выше формата в формат
www.examplel.com/usr/someguy

11.2.3. Заданиепредварительного условия: директива RewriteCond

Сама по себе директива RewriteCond ничего не переписывает. Она используется в совокупности с директивой Rewrit eRule для определения предварительных условий, которые должны быть удовлетворены до того, как механизм попробует подобрать соответствие шаблону, заданному директивой RewriteRule.

Многие переменные директивы RewriteCond могут составлять последовательности (логическое "И"). Если вам необходимо выполнить соответствующую директиву RewriteRule, когда одна директива RewriteCond или их последовательность принимает значение "истина", воспользуйтесь флагом "OR".

Директива RewriteCond имеет следующий синтаксис:

```
RewriteCond testvar condpattern flag,
```

где параметр testvar должен принимать одно из значений, перечисленных в табл. 11.3.

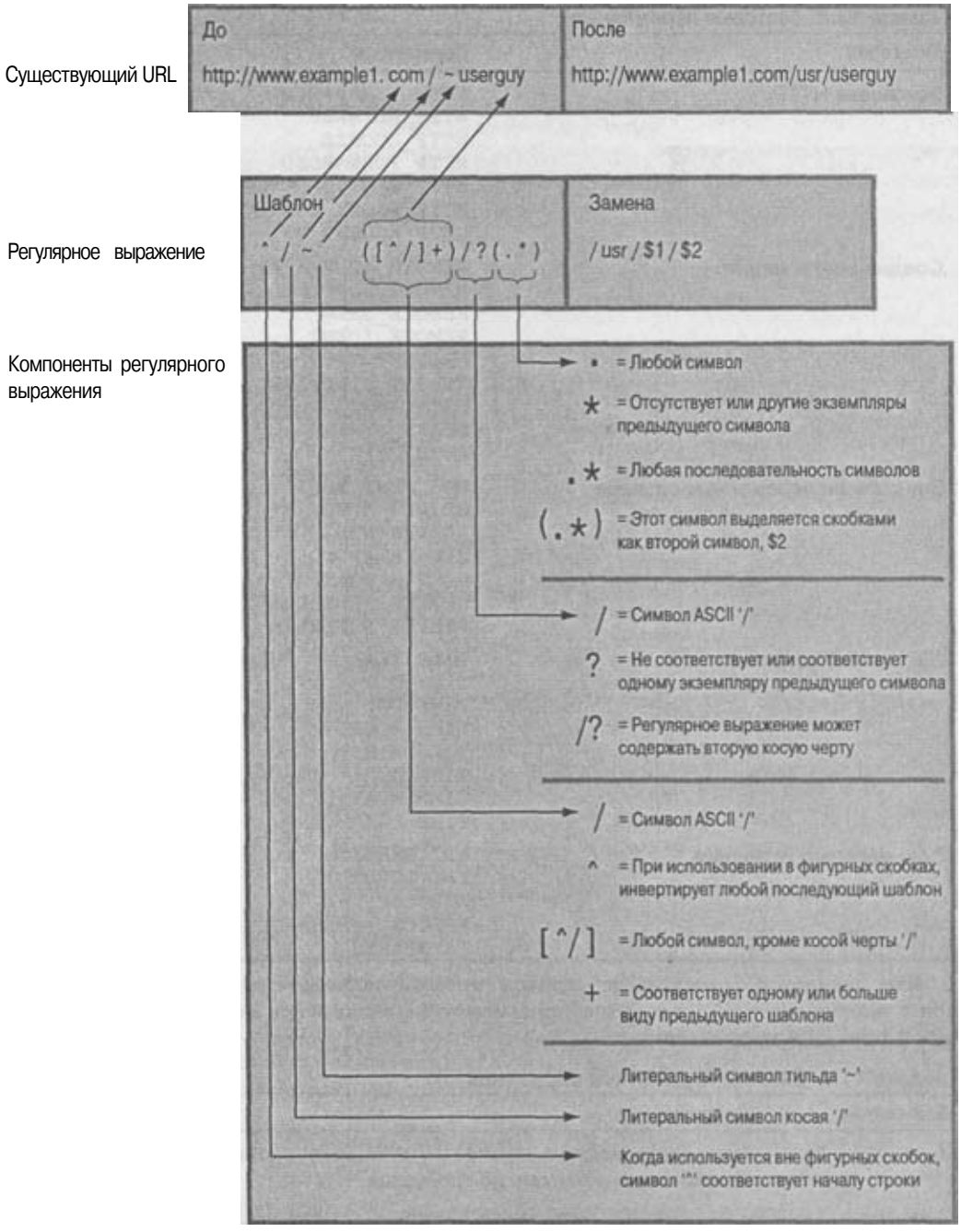

Рис. 11.1. Декомпозиция регулярного выражения

Таблица 11.3. Тестовые переменные

Категория	*Переменная*
Заголовки HTTP	HTTP_USER_AGENT HTTP_REFERER HTTP_COOKIE HTTP_FORWARDED HTTP_HOST HTTP_PROXY_CONNECTION HTTP_ACCEPT
Соединение и запрос	REMOTE_ADDR REMOTE_HOST REMOTE_USER REMOTE_IDENT REQUEST_METHOD SCRIPT_FILENAME PATH_INFO QUERY_STRING AUTH_TYPE
Внутренние переменные сервера	DOCUMENT_ROOT SERVER_ADMIN SERVER_NAME SERVER_ADDR SERVER_PORT SERVER_PROTOCOL SERVER_SOFTWARE
Системные установки	TIME_YEAR TIME_MON TIME_DAY TIME_HOUR TIME_MIN TIME_SEC TIME_WDAY TIME
Специальные установки	API_VERSION THE_REQUEST REQUEST_URI REQUEST_FILENAME IS_SUBREQ

Параметр condpattern задает шаблон, который используется в операциях сравнения с переменной testvar. В шаблоне может быть применен любой из перечисленных в табл. 11.4 префиксов.

Таблица 11.4. Синтаксис условного выражения

Выражение	*Действие*
	Предшествование символа "!" (отрицание) в строке задает шаблон, соответствие которому не требуется.
<CondPattern	**Возвращает значение "Истина", если сравниваемая строка лексически меньше, чем заданное значение Condpattern.**
>CondPattern	**Возвращает значение "Истина", если сравниваемая строка лексически больше, чем заданное значение Condpattern.**

Окончание табл. 11.4

Выражение	*Действие*
=CondPattern	Посимвольное сравнение.
-d	Возвращает значение "Истина", если сравниваемая строка задает путь к существующему каталогу.
-f	Возвращает значение "Истина", если сравниваемая строка задает путь к существующему файлу.
-s	Возвращает значение "Истина", если сравниваемая строка задает путь к существующему файлу, размер которого больше 0.
-1	Возвращает значение "Истина", если сравниваемая строка задает путь к символической ссылке.
-F	Возвращает значение "Истина", если сравниваемая строка задает путь к существующему файлу, доступному серверу. Проверочный тест заключает в себе подзапрос и поэтому снижает производительность сервера в случае, если она используется не часто.
-U	Возвращает значение "Истина", если сравниваемая строка задает путь к существующему URL.

Кроме того, можно задать дополнительные флаги CondPattern:

nocaseINC	Сделать нечувствительным к регистру вводимых букв.
ornext | OR	Скомбинированное правило как если бы было задано логическое "Или".

Пример 11.4.

Пример директивы, проверяющей значение переменной SERVERPORT на значение 443.

```
RewriteCond SERVERPORT =443
```

Пример 11.5.

Пример директив, которые проверяют значение переменной SERVERPORT на значение 443 или 80. Соответствующее значение RewriteRule будет проверено, если одно из условий возвращает значение "Истина".

```
RewriteCond SERVERPORT =443 OR
RewriteCond SERVERPORT =80
```

Пример 11.6.

Пример директив, которые проверяют значение переменной SERVERPORT на значение 443 и то, что переменная REMOTEHOST не равна badguy, Соответствующее значение RewriteRule будет проверено, только если одно или оба условия возвращают значение "Истина".

```
RewriteCond SERVERPORT =443
RewriteCond REMOTEHOST !badguy
```

11.3. Усовершенствованный модуль mod_rewrite

Дополнительно к своим основным возможностям модуль mod_rewrite имеет ряд достаточно мощных дополнительных возможностей. Одной из таковых является возможность настройки модуля mod_rewrite для работы с текстовым файлом, базой данных или работающей программой. Таким образом можно получить информацию по перезаписи URL. Кроме того, модуль mod_rewrite имеет свои собственные возможности регистрации (уровень RewriteLog и RewiteLogLevel). Можно задать уровень, до которого на виртуальный сервер влияет настройка родительского сервера (RewriteOptions). Наконец, можно задать основной URL для использования в предполагаемых перезаписях (RewriteBase).

11.3.1. Упорядочение файлов: директива RewriteMap

Директива RewriteMap настраивает имена и размещение таких внешних ресурсов, как текстовые файлы, файлы баз данных и самых различных программ, для их последующего использования директивой RewriteRule. Примеры использования настроенного RewriteMap в директиве RewriteRule можно посмотреть в конце этого раздела. Директива RewriteMap имеет следующий синтаксис:

RewriteMap MapName MapType:M apSource

Параметр MapName задает имя настройки для последующего использования в директиве RewriteRule.Параметр MapName будет использоваться в правиле перезаписи с использованием одной из следующих конструкций:

```
${MapName : LookupKey }
${MapName : LookupKey | DefaultValue}
```

При использовании этих правил значение LookupKey будет использоваться в настройке, заданной в MapName.

Значения MapType и MapSource могут быть комбинацией:

- стандартного текста
- случайного текста
- хеш-файла
- внутренней функции
- внешней программы перезаписи

Стандартный текст

Тип	txt
Источник	/path/to/map/file

Если MapType является текстовой строкой, значение источника должно указывать на текстовый файл, отформатированный как

longdata abbreviation,

и на практике выглядит следующим образом:

```
##
## mapl.txt   —  rewriting map
##
Ralf.S.Engelschall   rse
Mr.Joe.Average    joe
```

Пример 11.7.

Предположим, что файл настройки находится в /opt/apache/maps/map1.txt. Это означает, что директива должна задаваться следующим образом:

```
RewriteMap real—to—user txt: /opt/apache/maps/map1.txt
```

Произвольный текстовый файл

Тип rnd

Источник /path/to/unix/file

При использовании этой возможности, полученное значение, например, www1|www2 |www3 |www4, анализируется на составляющие с использованием символа "|" в качестве разделителя, а возвращается одно из составляющих (например www3). Это можно использовать в установках proxy для балансирования нагрузки.

Пример 11.8.

Предположим, что есть файл размещения в формате . txt. Он имеет следующий вид:

```
##
## map2.txt — rewriting map
##
static     www1|www2|www3|www4
dynamic    www5|www6
```

и хранится в каталоге /opt/apache/maps, В таком случае воспользуемся следующей директивой:

```
RewriteMap servers rnd:/opt/apache/maps/map2.txt
```

Хеш-файл

Тип dbm

Источник /path/to/unix/file

В этом случае исходным файлом является двоичный файл NDBM, который был создан из текстового файла (см. выше), но потом преобразован для более эффективного поиска.

Пример 11.9.

Допустим существует файл NDBM, размещенный в каталоге /opt/apache/maps/map3.ndbm. В этом случае к нему можно получить доступ с помощью строки

```
RewriteMap quick_servers dbm:/opt/apache/maps/map3.ndbm
```

Внутренняя функция

Тип int

внутренняя функция сервера Apache

Здесь источником является внутренняя функция Apache. В настоящее время возможность создавать свои функции отсутствует, но можно воспользоваться уже существующими функциями, перечисленными в табл. 11.5.

Таблица 11.5. Функции

Функция	*Действие*
toupper	Преобразует весь ключ в верхний регистр.
tolower	Преобразует весь ключ в нижний регистр.
escape	Транслирует специальные символы в шестнадцатеричный код.
unescape	Транслирует шестнадцатеричный код обратно в специальные символы.

Пример 11.10.

Предположим, что возникла необходимость перевести все получаемые URL в нижний регистр. Это можно сделать с помощью функции:

```
RewriteMap lease int:tolower
```

Внешняя программа перезаписи

Тип prg

Источник /path/to/executable/file

В таком виде источником является программа или сценарий. Будьте осторожны при создании таких программ. Их зависание вызывает зависание сервера Apache.

Пример 11.11.

Предположим, что требуется некая произвольная трансляция URL. Для этого был создан сценарий на языке Perl **funky.pl** и сохранен в каталоге **/usr/bin**. Его можно вызвать командой:

```
RewriteMap funky prg:/usr/bin/funky.pl
```

Пример 11.12.

Предположим, что нужно перераспределить нагрузку на сервер на основе достаточно произвольной характеристики запрошенного пользователем URL Для этого создается простой текстовый файл распределения **dist.txt,** который будет ассоциировать адреса, начинающиеся с букв от "a" до "f" с одним сервером, адреса, начинающиеся с букв от "д" до "l" с другим, и так далее. Файл распределения хранится в каталоге /etc. Директива **RewriteMap** применяется для того, чтобы информировать об этом сервер с помощью имени **distributed** следующим образом:

```
RewriteMap distributed txt:/etc/dist.txt
```

Затем получим доступ к заданному распределению с помощью директивы **RewriteRule.**

```
RewriteRule ^/(.){.*) ${distributed:$1|www.default.com}$1 $2 [R,L]
```

Первый символ "." просматривает первый символ и сохраняет его в аргументе *$1,* Второй символ ".*" символизирует остаток оригинальной строки. Первый символ сравнивается с ключами, хранящимися в файле **distributed,** и возвращается имя нового сервера. В случае, если совпадение не обнаружено, используется сервер *www.default.com.* Наконец, первоначальная строка присоединяется в конец к новой строке.

11.3.2. Регистрация: директивы RewriteLog, RewriteLogLevel

Модуль mod_rewrite может хранить информацию в журнале о производимых операциях. Имя и размещение журнала задает администратор. Чтобы включить такую

возможность, воспользуйтесь директивой RewriteLog, которая принимает в качестве параметра только путь к регистрационному журналу.

```
RewriteLog /var/logs/rewr itelog
```

Имеются различные уровни регистрации, начиная с 0 (регистрация не производится) до 9 (глубокая детализация). Установите уровень регистрации с помощью директивы RewriteLogLevel.

```
RewriteLogLevel 3
```

11.3.3. Наследование: директива RewriteOptions

Директива RewriteOptions была задумана как многофункциональная конфигурационная директива, но в данный момент единственным вариантом ее применения является задание возможности наследования от родителей конфигурационных свойств виртуальными узлами и каталогами (например, размещение или условия). Чтобы включить возможность наследования, необходимо задать директиву вида:

```
RewriteOptions inherit
```

11.3.4. Назначение основного каталога: директива RewriteBase

При задании правил перезаписи в конфигурационных файлах, расположенных в отдельных каталогах (например файл .htaccess), эта директива может использоваться для задания альтернативных путей, которые будут использоваться вместо текущего каталога.

```
RewriteBase /some/other/path
```

Глава

12

СОСТАВ МОДУЛЯ

В этой главе...

12.1. Введение

Среди целей, которые преследовались при разработке сервера Apache, можно отметить и следующую — создать такое программное обеспечение, которое могло бы работать с продуктами сторонних разработчиков. Для этого был создан Apache API (Application Programmer Interface — интерфейс программирования приложений). Он позволяет взаимодействовать с сервером практически во всех аспектах его работы. Это означает, что разработчик может создавать свои собственные директивы и настраивать сервер в любой фазе его работы.

Существует огромное множество вариантов, значительно больше, чем может вместить в себя одна глава. Целью этой главы является ознакомление читателя с концепциями и методами, использованными при создании модулей сервера Apache.

Для более полного изучения программного интерфейса Apache существует только одна книга, на которую стоит обратить внимание, — это книга *Writing Apache Modules with Perl and C,* написанная в соавторстве *Линкольном Штайном (Lincoln Stein)* и *Дугам Мак-Ичерном (Doug MacEachem).*

Информации, имеющейся в этой главе, вполне достаточно для того, чтобы войти в курс дела. Темы, которые здесь затронуты, включают получение и процесс компиляции модуля mod_perl (компиляция модулей сторонних разработчиков действительно является нетривиальной задачей), а также создание работающего интерпретатора языка Perl. Надо признаться, что интерпретатор берет на себя много функций, но когда он работает, вы можете возложить на сценарии Perl выполнение задач любой сложности.

12.1.1. Модуль mod_perl

Первоначальный программный интерфейс сервера Apache ограничивался только языком C. Но в 1996 году паренек по имени *Дуг Мак-Ичерн* создал модуль mod_perl, являющийся полнофункциональным интерпретатором Perl, реализованным как МОдуль Apache. Модуль mod_perl с помощью вызова методов и объектов позволяет по-

лучить доступ фактически ко всем доступным возможностям благодаря интерфейсу C API. Это хорошо потому, что модули, созданные в языке Perl, при незначительной потере производительности имеют тенденцию к большей лаконичности, чем аналогичные программы, написанные на языке C. Совершенно справедливо, что для получения максимально возможной производительности по-прежнему приходится создавать модули на языке C, но во всех остальных случаях лучше всего прибегнуть к помощи языка Perl.

12.2. Инсталляция модуля mod_perl

Модуль mod_perl в стандартный дистрибутив не включен, но его можно бесплатно загрузить с узла *http://perl.apache.org*.

Посетите этот узел и загрузите дистрибутив. Дистрибутив хранится в упакованном виде. Распакуйте его.

```
tar -zxvf mod_perl-X.XX.tar.gz
```

В результате будет создан каталог mod_perl-X.XX. Перейдите в этот каталог.

```
cd mod_perl-X.XX
```

Этот дистрибутив имеет в своем составе файл Makefile.PL, который предназначен для генерирования Makefile, отражающего особенности вашей системы. Существует множество вариантов, с которыми стоит хорошо познакомиться для того, чтобы узнать, что действительно применимо в вашей системе. Все они перечислены в табл. 12.1

Таблица 12.1. Команды файла Makefile.pl

Команды	*Описание и пример*
ADD MODULE	Эта команда задает дополнительные модули сверх того, что устанавливается по умолчанию. Модули добавляются в программу httpd. `ADD_MODULE=proxy,info`
APACHE_PREFIX	Для определения корневого каталога сервера Apache. `APACHE_PREFIX=/opt/apache.`
APACHE SRC	Команда, задающая размещение исходного дерева сервера Apache. `APACHE_SRC=/opt/apache/src`
APACI_ARGS	Дополнительные аргументы, необходимые для конфигурационного сценария Apache. Аргументы заключаются в кавычки, отдельные аргументы заключаются в скобки. `APACI_ARGS="-enable-module=ssl, -with-ssl=/opt/openssl-0.9.5.a"`
DO_HTTPD	Команда предназначена для построения сервера без вывода интерактивной подсказки. `DO_HTTPD=1`
DYNAMIC	Команда применяется для построения расширений сервера Apache:: * API какразделяемых библиотек. `DYNAMIC=1`

Окончание табл. 12.1

Команды	*Описание и пример*
EVERYTHING	Перевести модуль mod_perl в режим обработки всех фаз цикла запроса сервера Apache. EVERYTHING=1
PERL_DEBUG	Построить модуль mod_perl с включенной возможностью отладки С-ПрОГрамМ. PERL_DEBG=1
PERL_DESTRUCT_LEVEL	Эта команда используется для сообщения модулю mod_perl инструкции по проведению дополнительных проверок во время выключения сервера. Допустимые значения 1 и 2. PERL_DESTRUCT_LEVEL=1
PERL_TRACE	Диагностика модуля mod_perl во время выполнения. PERL_TRACE=1
PREP_HTTPD	Эта команда необходима для предотвращения построения всей программы httpd. Производится построение только части, отвечающей за интерфейс с Perl. PREP_HTTPD=1
SSL_BASE	При построении модуля mod_perl на сервере, который уже имеет модуль SSL, эта команда используется для определения размещения библиотек SSL. SSL_BASE=/opt;openssl-0.9.5.a
USE_APACI	Вместо *src/configuration* используется интерфейс автоконфигурации сервера Apache. USE_APACI=1
USE APXS	Сообщает модулю mod_perl о том, что он будет построен с помощью утилиты apxs. Она строит модули независимо от исходного дерева каталогов. USE_APXS=1
USE_DSO	Сообщает модулю mod_perl о том, что он строится как динамический разделяемый объект. USE_DSO=1
WITH_APXS	Этотаргумент используется для определения расположения утилиты APache extension. Обычно используется только тогда, когда программа apxs не найдена в основном инсталляционном каталоге сервера Apache. WITH APXS=some/directory/path

С другой стороны, обработку можно ограничить отдельными фазами цикла запросов. Опции Makefile. PL и соответствующие им директивы Apache перечислены в табл. 12.2.

Таблица 12.2. Опции файла Makefile.pl

Опция *Makefile.pl*	Соответствующая директива *сервера Apache*
PERL_DISPATCH=1	PerlDispatchHandler
PERL_CHILD_INIT=1	PerlChildln itHandler
PERL_CHILD_EXIT=1	PerlChildExitHandler
PERL_INIT=1	PerlInitHandler
PERL_POST_READ_REQUEST=1	PerlPostReadRequestHandler
PERL_TRANS=1	PerlTransHandler
PERL_HEADER_PARSER=1	PerlHeaderParserHandler
PERL_ACCESS=1	PerlAccessHandler
PERL_AUTHEN=1	PerlAuthenHandler
PERL_AUTHZ=1	PerlAuthzHandler
PERL_TYPE=1	PerlTypeHandler
PERL_FIXUP=1	PerlFixupHandler
PERL_HANDLER=1	PerlHandler
PERL_LOG=1	PerlLogHandler
PERL_CLEANUP=1	PerlCleanupHandler

12.2.1. Построение модуля mod_perl

Как видно из приведенной выше таблицы, файл Makefile.PL имеет множество опций. Вряд ли у вас все получится с первого раза. Поэтому рекомендуется создать отдельный сценарий, функции которого заключаются только в вызове файла Makefile.PL. При построении программы httpd с модулем mod_perl можно воспользоваться следующей последовательностью директив, генерирующей Makefile:

```
perl Makefile.PL \
       PREFIX=/opt/apache \
       APACHE_PREFIX=/opt/apache   \
       APACHE_SRC=/opt/apache/src \
       DO_HTTPD=1 \
       EVERYTHING=1
```

Чтобы построить новый файл httpd, запустите соответствующую утилиту

```
make
```

Затем проинсталлируйте его обычным способом. Полученная в результате программа httpd не содержит модуля mod_ssl или еще каких-то стандартных модулей. Если ваш вариант программы httpd, кроме модуля mod_perl, будет включать и другие модули сторонних разработчиков, можно порекомендовать построить модуль mod_perl как динамический модуль.

```
perl Makefile.PL \
       USE_APXS=1
       WITH_APXS=/opt/apache/bin/apxs  \
       EVERYTHING=1
```

Запустите утилиту make и создайте модуль libexec.so, затем командой

```
make install
```

модифицируйте файл httpd.conf для того, чтобы установить полученный модуль в каталоге libexec. С этого момента будьте готовы приступить к созданию своих собственных модулей.

12.3. Программный интерфейс Apache API

Сервер Apache обрабатывает полученные запросы по определенному алгоритму. Во время каждого шага Apache делает проверку, был ли задан дескриптор. Эта проверка включает такую последовательность операций:

- Транслирование запрошенного адреса в файл.
- Проверку пользователя.
- Проверку прав пользователя.
- Определение MIME-типов.
- Генерацию ответа.
- Регистрацию проделанных операций.

На протяжении всех этих фаз есть возможность вызова дескриптора с помощью директив модуля mod_perl. Сервер Apache передает всем дескрипторам в качестве аргумента структуру request_rec ($r). В приложении K, "Интерфейс mod_perl API" приводится полная распечатка методов и классов, имеющихся в модуле mod_perl.

12.4. Создание дескрипторов

Нет сомнения, что после успешной установки модуля mod_perl, у вас возникнет желание создать свой собственный модуль сервера Apache. Это тема отдельной книги, но материала, приведенного здесь, будет вполне достаточно для того, чтобы попробовать сделать это.

12.4.1. Размещение модуля Perl

Сервер Apache должен знать, где можно найти созданные вами модули. По умолчанию модуль mod_perl размещен в ServerRoot в каталоге lib/perl, находящемся в каталоге ServerRoot. Из двух вариантов лучшим, конечно же, будет lib/perl, ведь очевидно, что каталог ServerRoot к этому моменту уже переполнен. Если же вам не подходит ни один их этих каталогов, для изменения значения переменной PERL5LIB воспользуйтесь директивой PerlSetEnv.

```
PerlSetEnv PERL5LIB /some/other/location
```

Далее, так как большая часть операций будет производиться в пространстве имен Apache::, вам потребуется создать подкаталог Apache в каталоге, в котором находятся ваши библиотеки.

```
mkdir /opt/apache/lib/perl/Apache
```

С другой стороны, для запуска стартового файла может потребоваться создать Perl-сценарий (расширение .pl).

```
#!/usr/bin/perl
BEGIN {
     use Apache();
     use lib Apache->server_root_relative ('lib/perl');
}
1;
```

Этот файл можно запустить с помощью директивы PerlRequire:

```
PerlRequire conf/setup.pl
```

12.4.2. Объект запроса

Объект запроса представляет собой первичное средство обмена данными между сервером и его модулями. Детальное изложение его реализации и эксплуатации находятся вне пределов этой книги, но из приложения K, "Интерфейс mod_perl API" можно увидеть, с чем придется работать, и пример, приведенный ниже, позволит вам получить общее представление об их использовании. В общих чертах объект запроса используется в следующих целях:

- Получение информации о соединении (метод get_server_port ()) и пользователе (метод get_remote_host ()).
- Получение данных методом POST или загрузка данных (метод read ()).
- Пересылка данных на броузер пользователя (метод print ()).
- Получение или модифицирование заголовков HTTP (метод header_out ()).
- Запись информации в регистрационный журнал (метод log error ()).

12.4.3. Основной модуль

Модуль, описанный ниже, связан с определенным <Location>. Этот модуль при запуске создает простейший HTML-файл.

```
1. package Apache::BasicModule;
2. use Apache::Constants qw(OK);
3. sub handler {
4.      my $req_obj=shift;
5.      $req_obj->content_type('text/html') ;
6.      $req_obj->send_http_header;
7.      $req_obj->print(
8.           "<HTMLXHEAD>",
9.          "<TITLE> Basic Handler </TITLE>",
10.     "</HEADXBODY>"    ,
11.        "<H1> This is a test handler.</Hl>",
12.        "</BODYX/HTML>") ;
13.     return OK;
14.  }
15.  1;
```

В строке 1 делается объявление, что это будет часть пространства имен сервера Apache. В строке 2 предоставляется доступ к классу Apache::Constants, в котором содержатся константы, методы, коды ответа и т.д.

Со строки 3 начинается собственно тело мини-модуля, подпрограммы, handler. Эта подпрограмма вызывается сервером Apache по запросу к каталогу <Location>, который будет задан в следующей части. В строке 4 объект запроса, переданного сервером, сохраняется в переменной $req_obj.

Строка 5 устанавливает MIME-тип в text/html, и строка 6 с помощью функции send_http_header () объекта запроса посылает заголовки http. Строки от 1 до 12 с помощью метода print объекта запроса записывают несколько строк HTML-кода для передачи его клиенту.

12.4.4. Вызов основного модуля

В этом частном случае мы попытаемся связать только что созданный дескриптор с каталогом размещения perl-demo.

```
<Location /perl-demo>
    SetHandler perl-script
    PerlHandler Apache::BasicHandler
</Location>
```

Директива <Location> предупреждает сервер Apache о том, что каталог /perl-demo особенный. Директива SetHandler сообщает серверу Apache о необходимости использовать модуль mod_perl для обработки всех запросов, которые были адресованы к этому каталогу. Наконец, директива PerlHandler сообщает mod_perl о том, что все запросы, адресованные к этому каталогу, должны обрабатываться модулем BasicHandler.

Задавая в броузере адрес perl-demo, вы увидите на своем экране картинку, аналогичную изображенной на рис. 12.1.

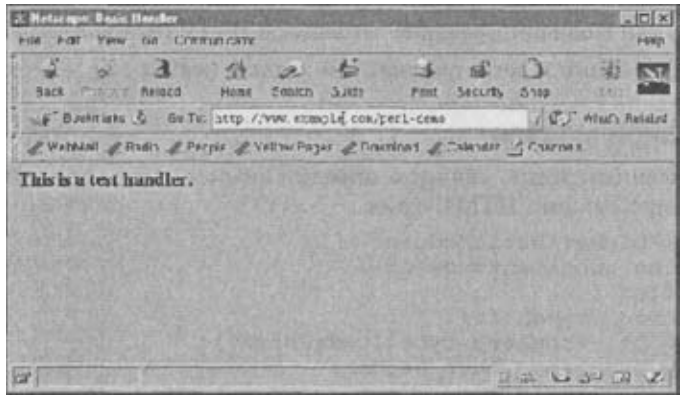

Рис. 12.1. Вывод основного модуля

12.5. Директивы настройки Perl API

Директивы, перечисленные в таблице 12.3, связаны с программным интерфейсом Perl API.

Таблица 12.3. Директивы программного интерфейса Perl API

Директива	*Описание и пример*
PerlFreshRestart	Перезагрузить все модули, работающие под управлением модуля mod_perl, которые были найдены в %INC после перезапуска сервера. `PerlFreshRestart on`
PerlModule	Определить пути @INC для файла .pm, соответствующего указанному имени, и если он найден, загрузить его. `PerlModule Apache::BasicHandler`
PerlRequire	Загрузить Perl-файл с диска аналогично модулю PerlModule. Однако эта директива в качестве параметра принимает путь к файлу модуля. `PerlRequire /opt/apache/lib/perl/Apache/BasicHandler.pm`

12.6. Директивы дескриптора

Перечисленные в табл. 12.4 директивы, используются для запуска модулей в определенные моменты жизненного цикла сервера Apache. Каждая из них принимает в качестве параметра имя модуля (например Apache::BasicHandler).

Таблица 12.4. Директивы запуска модулей

Директива	*Фаза*
PerlchildinitHandler	Эта директива инсталлирует дескриптор, который запускается сразу же после запуска порожденного процесса.
PerlPostReadRequestHandler	Эта директива инсталлирует дескриптор, который запускается каждый раз при получении процессом Apache запроса пользователя.
PerlinitHandler	Эта директива инсталлирует дескриптор, который запускается первым HTTP.
PerlTransHandler	Эта директива задает дескриптор, который запускается после анализа сервером запроса URL, для того, чтобы транслировать его в файл.
PerlHeaderParserHandler	**Эта директива запускается после трансляции URL.** Это очень существенно на начальной фазе, когда URL определенно ссылается на реальный файл (и поэтому может быть использована в директивах <Directory>, <Location> и операторных скобках).
PerlAccessHandler	Эта директива запускает дескриптор, который осуществляет проверку доступа на основании свойств броузеров клиента.
PerlAuthenHandler	Эта директива запускает дескриптор, проверяющий идентичность пользователя на основании имени пользователя и пароля.
PerlAuthzHandler	Эта директива запускает дескриптор, который проверяет права доступа пользователя, идентичность которого была проверена.
PerlTypeHandler	Эта директива запускает дескриптор, который оценивает запрошенный документ и назначает для него временный MIME-тип.
PerlFixupHandler	Эта директива задает дескриптор, который действует в промежутке между проверкой типа (директива PerlTypeHandler) и генерацией содержимого (директива PerlHandler). При необходимости что-либо сделать в этой точке, воспользуйтесь этойдирективой.
PerlHandler	Эта директива задает дескриптор, который запускается во время фазы генерации содержимого. Предположительно — это самый популярный тип модуля Perl.

Окончание табл. 12.4

Директива	***Фаза***
PerLogHandler	Эта директива задает дескриптор, который запускается до очистки. Независимо от того, что в конце концов регистрация закончилась, модуль, который запускается этой директивой, перебирает все, что сервер Apache знает о только что законченной транзакции, чтобы связанный модуль отделил то, что ему необходимо сохранить, от всего остального.
PerlcleanupHandler	Эта директива задает дескриптор, который запускается во время фазы очистки для удаления временных файлов, освобождения разделенной памяти и общей очистки.
PerlchildExitHandler	Эта директива задает дескриптор, который запускается только перед уничтожением порожденного процесса.

12.7. Соображения на тему производительности

Модуль mod_perl отличается большим размером и соответственно требует много памяти. Несмотря на все возможные предосторожности, с большой вероятностью можно сказать, что увеличение размера процесса httpd приведет к обмену данными между процессами httpd и областью подкачки. Этот обмен только изредка будет прерываться для осуществления полезной работы. Идеальным способом избежать этого является создание двух отдельных физических серверов на одном из которых будет работать сервер, поддерживающий работу модуля mod_perl, и соответственно обрабатывающий запросы, связанные с обработкой модулем mod_perl, а на другом, менее мощном, модуля mod_perl не будет вообще. Направьте все запросы, связанные с обработкой модулем mod_perl, на соответствующий сервер.

Принимая во внимание то, что наличие двух серверов может не вписываться в рамки вашего бюджета, можно воспользоваться двумя имеющимися в распоряжении IP-адресами. Это позволит ограничить масштаб загрузки памяти запуском двух серверов на одной и той же машине. Воспользуйтесь директивой BindAddress для того, чтобы сервер, работающий с модулем mod_perl, обрабатывал запросы, поступающие на IP-адрес A, а сервер, не работающий с модулем mod_perl, обрабатывал запросы, поступающие на IP-адрес B.

Часть III

Электронная коммерция

В этой части...

1. Денежные платежи
2. Взаимодействие с базами данных
3. Пример коммерческого узла

Глава

13

ДЕНЕЖНЫЕ ПЛАТЕЖИ

В этой главе...

13.1. Введение

Эта глава по своей проблематике ближе предпринимателям, а технические детали обмена финансовыми данными находятся в компетенции вашего банка. Самое лучшее, что может эта книга — предложить варианты решений и направление для дальнейшего поиска.

13.2. Кредитные карточки

Кредитные карточки — это стремительно развивающийся стандарт электронной коммерции. Фактически все участники Internet-процесса уже имеют в своем распоряжении электронную кредитную карточку, будь это стандартная карточка Visa или карточка дебетового типа. Как бы там ни было, если вы планируете развивать бизнес в Internet, лучше всего принимать платежи по кредитным карточкам.

Чтобы делать это с полной ответственностью, ваш сервер должен иметь возможность работы с протоколом SSL (Secure Sockets Layer). Описание этого процесса можно найти в главе 8, "Безопасность". Такое требование выставляется перед системами оплаты потому, что передавать информацию о кредитной карточке по открытому каналу небезопасно. При наличии протокола SSL в Web-страницу можно включать ссылку на безопасную страницу, куда пользователи будут передавать информацию о своих кредитных карточках.

13.2.1. Проверка номера кредитной карточки

Номера кредитных карточек проверяются в соответствии с алгоритмом, определенным стандартом ISO 2894. Этот алгоритм заключается в следующей последовательности действий:

1. Определение весового значения для каждой цифры.
   - Если номер карточки имеет четное число цифр — первая цифра имеет вес 2. При нечетном количестве цифр в номере карточки — первая цифра имеет вес 1.

« Веса последующих цифр колеблются между 1 и 2.

2. Каждая цифра умножается на ее вес.
3. Из каждой цифры, вес которой превышает 9, вычитается 9.
4. Веса всех цифр складываются, и полученный результат делится на 10. В результате этой операции получаем остаток.

После проверки информации с кредитной карточки транзакция идентифицируется. Эту задачу решает программное обеспечение, предоставленное вашим банком. После того как платеж идентифицирован, информация о кредитной карточке шифруется и записывается в безопасном месте в локальной сети, формируя пакет транзакций, который затем посылается в соответствии с интервалом рассылки, установленным между вами и вашим банком. Оставив номер кредитной карточки незашифрованным, вы навлекаете на себя большие неприятности.

Информация о кредитной карточке, за исключением случаев периодических проплат по ней, не должна храниться долгое время.

13.3. Автоматизированная расчетная палата

Передача данных методом "Автоматизированная расчетная палата" (ACH — Automated Clearing House) является популярным средством оформления периодических оплат. АРП автоматически вычитает деньги прямо с расчетного счета. Этот метод не совсем подходит для совершения одноразовых платежей, так как процесс его установки требует значительного объема бумажной работы[1].

Сам по себе этот метод относительно прост. В файл данных собирается и записывается масса информации о ваших клиентах (имена, номера счетов и т.д.), которая затем передается в ваш банк. Получив этот файл, банк оценивает его формат, вносит в него необходимые изменения, и передает в Федеральный Резерв. Днем позже вы получаете ваши деньги.

13.4. Коммерческие продукты, использующие шифрование с открытым ключом

Существует множество компаний, которые пытаются утвердить себя в качестве разработчиков и поставщиков решений в электронной коммерции. По-моему, необходимость их существования сегодня находится под большим вопросом, но мы рассмотрим этот вариант в качестве приемлемой альтернативы разработке собственного программного обеспечения.

13.4.1. Компания e-Cash

- Компания e-Cash строит свою стратегию на том, что покупатель и продавец должны быть зарегистрированы в определенном финансовом учреждении, с которым у компании e-Cash имеются соответствующие договорные отношения. В принципе нет большого различия между наличием таковых и необходимостью иметь кредитную карточку. Недостатком можно считать не столь широкое распространение e-Cash по сравнению с универсальной карточкой Visa.
- Она недоступна на территории США.

[1] Недавно в США был принят закон, дающий законную силу электронным подписям. Так что банки в скором времени пойдут на существенное сокращение бумажной волокиты, необходимой для оформления такого типа проплат.

13.4.2. Компания CyberCash

Существует еще одна компания, которая использует возможности метода шифрования с открытым ключом в платежах, — это компания CyberCash. Эта компания предлагает на рынке инструмент разработки программ в среде Web, так называемый Merchant Connection Kit (MCK). Он имеет в своем арсенале инструментарий разработки на языках программирования C и Perl. Инструментарий MCK используется для обмена данными между вашим приложением и их службой регистрации платежей (Cash Register).

Эта служба (CashRegister) действует как своеобразный посредник между вами и вашим покупателем. Когда покупатель выражает желание что-либо купить, информация о нем шифруется и передается в компанию CyberCash посредством CashRegister. В свою очередь компания CyberCash пересылает информацию в ваш банк, который одобряет либо отклоняет транзакцию и информирует CyberCash о своем решении. Теперь компания CyberCash информирует уже вас, и вы завершаете процедуру купли-продажи.

Кроме того, этот интерфейс разработки приложений можно использовать для построения пакетного файла обработки платежей. Этот процесс не похож на процесс обработки платежей с помощью пакета ACH, он больше подходит для работы с кредитными карточками, а не расчетными счетами. Когда потребитель вводит информацию, она хранится на сервере до тех пор, пока не будет передана в пакет. Эта информация периодически используется для форматирования файла учета платежей, который отсылается в компанию CyberCash. Они обрабатывают платежи и отсылают остаток, имеющийся на вашем счете.

Наконец, если вы собираетесь получать платежи по телефону, MCK имеет в своем составе утилиту, позволяющую производить платежи с помощью кредитной карточки вручную через Internet. Для этого необходимо зарегистрироваться на сервере безопасности компании CyberCash, ввести информацию о потребителе, а затем получить подтверждение или отказ.

Для ознакомления с деталями процесса можно обратиться по адресу *http ://www. cybercash .com.*

13.5. Протокол SET

Протокол SET (Secure Electronic Transaction — безопасная электронная транзакция) представляет собой совместную разработку компаний Visa и MasterCard. Идея, заложенная в нем, заключается в идентификации с помощью специального цифрового сертификата как потребителя, так и продавца. Эта идентификация производится до того, как начнется любой обмен информацией. Такой подход справедлив благодаря тому, что та часть Web-сообщества, которая занимается торговлей, все больше укрепляется во мнении, что продавец в свою очередь тоже может быть ложным.

Протокол SET требует, чтобы покупатель обзавелся своим электронным счетом и цифровым сертификатом до момента свершения сделки. Фатальным недостатком всех новых стандартов является их неудобство. Однако протокол SET имеет следующие преимущества:

- Есть поддержка таких гигантов, как Visa и MasterCard.
- Имеется поддержка основных известных броузеров.
- Использованные в нем методы шифрования не настолько секретны, чтобы правительство США каким-то образом могло ограничить экспорт технологии SET.

Еще одним преимуществом протокола SET является то, что он может предоставить покупателям определенную конфиденциальность. Покупая что-либо с помощью прото-

кола SET, продавец может получить от системы SET простое подтверждение того, что платеж произведен. Информация о вашем адресе и даже имени необязательна. Конечно, при покупке реальных товаров потребления вам придется указать адрес доставки, что несколько нарушает конфиденциальность. Кроме того, SET-банкиру совсем необязательно знать, что именно вы покупаете, его интересует только сумма покупки.

Глава

14

ВЗАИМОДЕЙСТВИЕ С БАЗАМИ ДАННЫХ

В этой главе...

14.1. Введение

В электронной коммерции любое решение средней сложности требует определенного взаимодействия с базами данных. Базы данных де-факто стали стандартным инструментом отслеживания заказов, хранения информации о товарах и т.д.

Сервер Apache не имеет никаких механизмов взаимодействия с реляционными базами данных. Модули санкционирования доступа (mod_auth_dbm, mod_auth_db и т.д.) имеют достаточно ограниченные возможности, поэтому их в расчет можно не брать. На данный момент можно с уверенностью сказать, что сервер не имеет модулей, которые могли бы обеспечить полноценную работу с базами данных.

На практике для решения проблемы работы с базами данных придется обратиться к программным продуктам сторонних производителей. Целью этой главы явл яется освещение наиболее оптимальных решений задачи взаимодействия с базами данных.

14.2. Базы данных

Факторы, которые принимаются во внимание в процессе выбора самой подходящей базы данных, лежат за пределами тематики этой книги. Очевидно, что любая компания уже давно определилась с используемой в своем бизнес процессе базой данных. В каждом конкретном случае вам придется иметь дело с администратором базы данных, бюджетом компании и некоторыми другими трудноразрешимыми проблемами. Этот раздел может пригодиться тем немногим счастливчикам, у кого еще есть роскошь выбора своей собственной базы данных, но вряд ли читатель сможет получить ответ, как это сделать. Опытные администраторы баз данных могут с чистой совестью перейти к другому материалу.

14.2.1. СУБД MySQL

База данных MySQL является полнофункциональной базой данных, работающей на платформах Unix и Win32/NT. Это масштабируемая реляционная база данных с достаточно высокими рабочими характеристиками. СУБД MySQL может заинтересовать малобюджетные организации. СУБД MySQL можно получить в свое распоряжение бесплатно, правда не всегда. Детально условия поставки изложены в лицензии, но вкратце суть заключается в том, что если вы хотите использовать СУБД MySQL в каком-либо продукте или услуге, которые будут предоставляться за деньги, за лицензию вам придется заплатить. В настоящее время стоимость лицензии для неограниченного числа пользователей составляет 200 долларов. Устаревшие версии поставляются бесплатно.

Существует множество групп новостей и списков рассылки, в которых можно найти бесплатную техническую поддержку. Кроме того, за определенную плату можно получить MySQL на одноименном Web-узле. Среди моря высококачественной документации хочется выделить книгу *Поля Дюбуа "MySQL",* выпушенную Издательским домом "Вильямс".

В качестве одного из основных недостатков СУБД MySQL можно назвать отсутствие поддержки транзакций, и, таким образом, отсутствие операций commit и rollback как таковых. Транзакция — это понятие, органически входящее в концепцию баз данных, заключающееся в объединении операторов в группы. Операции COMMIT и ROLLBACK являются методом передачи базе данных информации о том, что группа операторов выполняется как единое целое, частичное выполнение компонентов которого недопустимо. Если один из операторов, составляющих эту группу, не будет выполнен, то не будет выполнена и вся группа операторов целиком. Возможности транзакции в этой СУБД воспроизводятся с помощью оператора LOCK TABLE, но разработчики, привыкшие к использованию операторов COMMIT и ROLLBACK, при их отсутствии почувствуют определенный дискомфорт.

С тонкостями применения СУБД MySQL можно познакомиться на Web-узле *http://www.mysql.com,* откуда можно загрузить и сам дистрибутив этой базы данных.

14.2.2. СУБД Oracle

СУБД Oracle является непререкаемым авторитетом в мире баз данных. Компания Oracle распространяет полнофункциональные версии своей базы данных для всех имеющихся в настоящее время платформ. СУБД Oracle имеет круглосуточную техническую поддержку. Естественно, как и полагается лидеру отрасли СУБД, Oracle имеет самый богатый выбор документации и литературы.

Основным недостатком платформы Oracle является ее цена, которая для многих покупателей просто недоступна. Однако в этом смысле наметились определенные подвижки: недавно компания Oracle выпустила версию СУБД Oracle 8i для платформы Linux подостаточно низкой цене.

14.2.3. СУБД Informix

Компания Informix была одним из первых продавцов на рынке СУБД, предоставивших версию для платформы Linux. Компания Informix продает большое количество полнофункциональных реляционных баз, дополненных отличными Internet-пакетами. Исторически СУБД Informix была лидером технологического прогресса и одно время даже опережала Oracle. Эта компания предлагает множество продуктов по вполне доступным ценам. Пробные версии и другое достаточно интересное программное обеспечение можно найти по адресу *http://www.infomux.com.*

Во всех примерах, приведенных ниже, предполагается, что база данных MySQL с именем ec_example уже существует.

14.3. Обмен данными, выбранными из базы данных

Предположим, что этап выбора базы данных успешно преодолен. Теперь перед нами стоит проблема иного порядка. Это проблема доставки содержимого базы данных через Internet. В этом разделе мы рассмотрим три различных подхода: CGI-приложения, модуль Apache и использование оболочек сторонних разработчиков.

14.3.1. CGI-решения с использованием модуля mod_perl и интерфейса Perl DBI

CGI-сценарии или программы, осуществляющие доступ к вашей базе данных, создавать очень просто и удобно. Каждая из вышеперечисленных платформ предоставляет в распоряжение программиста интерфейс разработки приложений на языке программирования C, позволяющий создавать программы с возможностью обмена данными с базами данных. Модуль mod_perl сочетает в себе почти все возможности программного интерфейса на языке C, реализованные в качестве объектов и методов. Их перечень приведен в главе 12, "Состав модуля".

14.3.2. Интерфейс PerlDBI

Основным недостатком прямых CGI-соединений с базами данных является их неустойчивость. При запросе к базе данных создается новое соединение с базой данных. Как только запрос клиента обслужен, соединение прерывается. Как нетрудно догадаться — это мощный удар по производительности системы.

Apache::DBI из модуля mod_perl является решением проблемы устойчивых соединений. Очевидно, что при этом требуется, чтобы модуль mod_perl был установлен вместе с Perl DBI и любым требующимся драйвером базы данных, необходимым для данной системы. Документацию и дистрибутивы можно получить по адресу *http://www.perl.com*.

Для этого в конфигурационном файле httpd.conf задайте следующую строку Apache::DBI

```
PerlModule::DBI
```

или строку

```
use Apache::DBI();
```

в стартовом файле Perl. При таких установках все соединения, создаваемые с помощью mod_perl, будут устойчивыми.

Интерфейс Perl DBI действует как посредник между приложением и драйвером, необходимым для подключения к определенной базе данных. Основной смысл заключается в том, что появилась возможность создавать переносимые между различными платформами баз данных приложения. Основным препятствием в обеспечении полной переносимости является изменчивость синтаксиса языка SQL (например, операторы COMMIT и ROLLBACK работают на Oracle и не работают на MySQL).

14.3.3. Дескриптор

Дескриптор базы данных в Perl DBI является соединением к конкретной базе данных. Создать дескриптор базы данных и получить доступ к локальной базе данных с помощью Perl DBI относительно несложно. Можно с помощью метода DBI->connect создать соединение, которое будет существовать на протяжении всего существования процесса httpd,

```
$dbh = DBI->CONNECT($datasource, $username, $password),
```

где datasource имеет вид

```
dbi:database_type:database_name,
```

а что такое username и password, догадаться совсем несложно. После подключения к базе данных дескриптор используется для подготовки дескриптора оператора.

```
$qry = $dbh->prepare ("SELECT field1, field2 FROM some_table")
```

И, наконец, выполняется сам запрос

```
$qry->execute ();
```

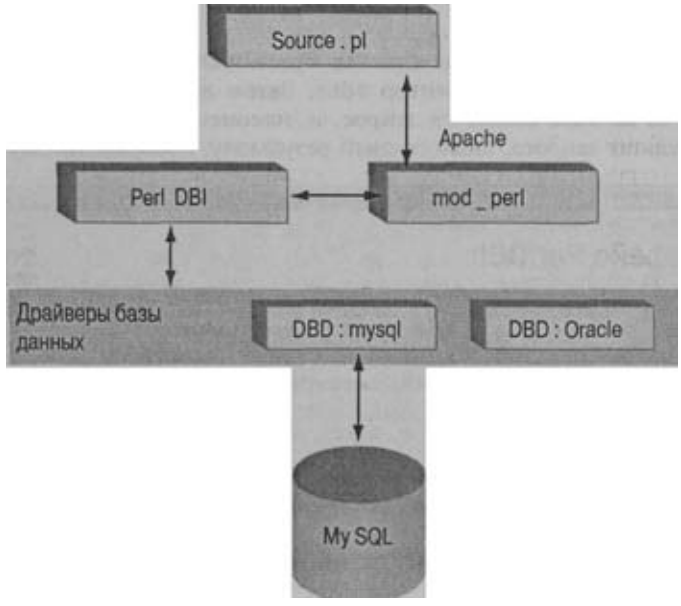

Рис. 14.1. Взаимосвязи между Perl DBI и Apache

Приведенный ниже пример программы является не чем иным, как модифицированным модулем BasicHandler.pm, который упоминается впервые в главе 12, "Состав модуля".

```
package Apache::BasicHandler;
use  Apache::Constants  qw(OK);
use  Apache::DBI(OK) ;
sub handler {
      my $req_obj=shift;

      $req_obj->content_type('text/html');
      $req_obj->send_http_header;

      $req_obj->print(
     "<HTML><HEAD><TITLE>",
           " Демонстрация Perl DBI </TITLE></HEAD>",
     "<BODY><H1> Этот пример иллюстрирует доступ к СУБД MySQL",
           " через perl DBI </H1>",
      my $dbh = DBI->CONNECT("dbi:mysql:ec_example","httpd", " ")
           or die "Нельзя подключиться к MySQL\n";
      my $qry = $dbh->prepare ("SELECT * FROM ccard ")
```

```
        or die "Нельзя создать SQL-оператор\n";
    $qry->execute();
        or die "Нельзя выполнить SQL-операТор\n";
    my @row
    while (@row = $qry->f etchrow_array())
          {
          $reg_obj -> print("@row <BR>");
          }
          $req_obj->print ("</BODY></HTML>");
    return OK;
}
I;
```

В этом примере доступ к базе данных будет получен с помощью метода DBI->connect, который создает дескриптор $dbh. Затем из текстовой строки с помощью дескриптора базы данных создается запрос, и, наконец, с помощью метода print генерируется результат запроса, аналогичный результату, изображенному на рис. 14.2.

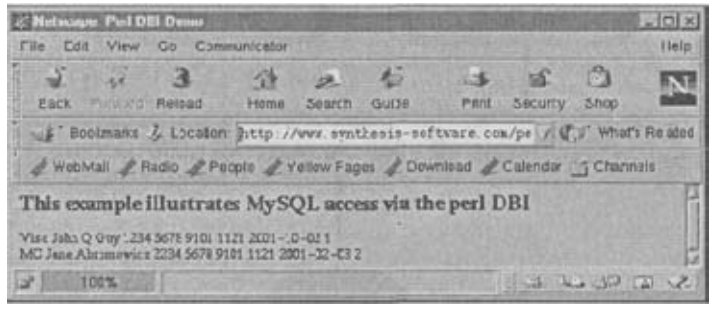

Рис. 14.2. Демонстрационный пример интерфейса Perl DBI

14.3.4. Среда разработки приложений ColdFusion

Среда разработки ColdFusion — это коммерческий продукт, созданный для взаимодействия между броузерами пользователей и любым количеством серверов баз данных, почтовых серверов и средств программирования.

Среда разработки ColdFusion продается вместе со средой визуальной разработки ColdFusion Studio, которой Web-дизайнеры сейчас просто очарованы, что с лихвой может оправдать покупку этого программного продукта по не очень низкой цене. Он позволяет производить разработку программного обеспечения с помощью интерфейса типа "перетаскивание" аналогичного тому, что есть у Visual Basic. Его интерфейс показан на рис. 14.3.

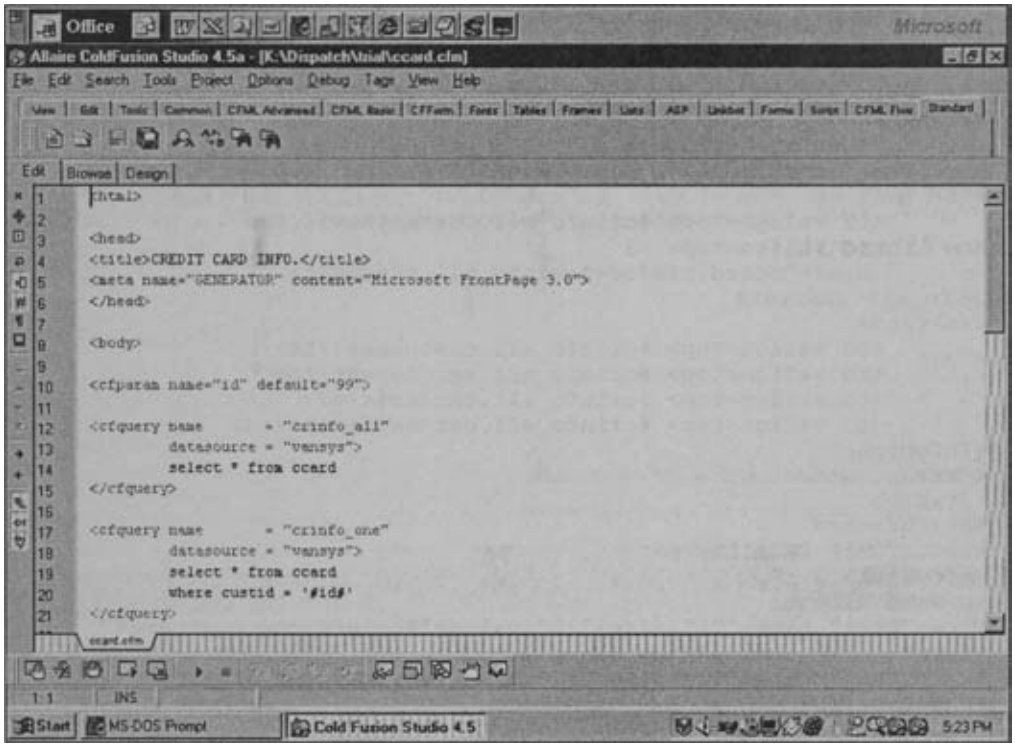

Рис. 14.3. ColdFusion Studio

Система разработки приложений ColdFusion позволяет программировать сложное взаимодействие между пользователем, сервером и базой данных без рутинного кодирования программ. Как видно из следующего примера, конечный продукт имеет вполне приличный вид (разработан всего за 20 минут с помощью ColdFusion Studio на платформе Windows):

```
<html>
<head>
<title>CREDIT CARD INFO.</title>
<meta name="GENERATOR" content="Microsoft Frontpage 3.0">
</head>
<body>
<cfparam name="id" default="99">
<cfquery name= "crinfo_all"
      datasource = "vansys">
      select * from ccard
</cfquery>
<cfquery name= "crinfo_one"
      datasource = "vansys">
      select * from ccard
      where custid = '#id#'
</cfguery>
<cfform name="crinfo" method="post " action="ccard.cfm">
<table border=1 cellpadding=1 cellspacing=1>
<tr>
      <td bgcolor=f0f0f0> <b><i>S.No</i></b></td>
      <td bgcolor=f0f0f0> <b><i>Customer Id</i></b></td>
```

```
        <td bgcolor=f0f0f0> <bxi>Customer Name</i></b></td>
        <td bgcolor=f0f0f0> <bxi>Card Type</i></b></td>
        <td bgcolor=f0f0f0> <bxi>Card Number</i></b></td>
        <td bgcolor=f0f0f0> <bxi>Expiry Date</i></b></td>
   </tr>
   <cfoutput query="crinfo_all " startrow="l"
   maxrows= "#crinfo_all . RecordCountt" >
   <tr>
          <td valign=top> #crinfo_all.CurrentRow#</td>
          <td valign=top> <a
           href="ccard.cfm?id=#crinfo_all.custid#">
Icrinfo_all.custidtl
   </a><7td>
          <td valign=top> #crinfo_all.custname#</td>
          <td valign=top> fcrinfo_all.cardtype#</td>
          <td valign=top> fcrinfo_all.cardno#</td>
          <td valign=top> #crinfo_all.cardexpdatett</td>
   </cfoutput>
   </tr>
   </table>
   <brxbr><br>
   <p>CUSTOMER DETAILS</p>
   <cfoutput>
   <p>Name <input
   type="text" name="Tl" size="20" value="#crinfo_one.custname#"> ID.
   <input
   type="text" name="T2" size="6" value="#crinfo_one.custid#"></p>
   <p> </p>
   <p>CREDIT CARD DETAILS</p>
   <p>Type <input
   type="text" name="T3" size="20" value="#crinfo_one.cardtype#"><br>
   Num. <input
   type="text"  name="T4"  size="20"  value=  "#crinfo_one.cardnol"  >
Exp<input
   type="text"  name="T5"  size="20"  value="  #crinfo_one.cardexpdatet
"></p>
   </cfoutput>
   </cfform>
   </body>
   </html>
```

Как видно из рисунка, этот вариант ColdFusion разработан для платформы Windows. Единственной официально поддерживаемой бесплатной платформой, на которой работает ColdFusion, является ОС Red Hat Linux. Имеющаяся документация содержит множество неявных предположений о типе платформы, которые попросту несправедливы по отношению к ОС Unix. Однако вполне возможно настроить приложения ColdFusion для работы под управлением ОС Linux, в частности Red Hat. Пробный дистрибутив и условия покупки ColdFusion можно получить по адресу *http://www.allaire.com*.

14.4 Язык PHP

Аббревиатура PHP означает Personal Hypertext Preprocessor — персональный гипертекстовый препроцессор. Он представляет собой язык написания сценариев, которые будут размещаться на сервере. Среди его многочисленных преимуществ уместно будет упомянуть возможность работы с базами данных. История его создания такова: он был разработан *Расмусом Лердорфом (Rasmus Lerdorf)* для ввода посетителями узла, имя которого уже мало кто помнит, своих резюме.

14.4.1. Получение и инсталляция

В данный момент есть новый релиз PHP версии 4, его можно получить по адресу *http://www.php.com*.

После загрузки дистрибутив распаковывается обычным для ОС Unix образом:

```
tar xvzf php-4.0.1p12.tar.gz
```

Как и сервер Apache, дистрибутив PHP снабжен конфигурационным сценарием, предназначенным для создания makefile. В зависимости от опций, переданных этому сценарию, он может быть статически прикомпилирован к серверу Apache.

```
./configure --with-apache=/opt/apache
```

При необходимости он может быть скомпилирован как разделяемый объектный файл.

```
./configure --with-apxs=/opt/apache/bin/apxs
```

Если число модулей сторонних разработчиков (mod_perl, mod_ssl) слишком велико или имеет тенденцию к росту, может оказаться более предпочтительным инсталлировать их как разделяемые объектные файлы. Те, кто работает с СУБД MySQL, могут воспользоваться опцией --with-mysql, которая значительно упрощает соединение с базой данных на стадии разработки.

```
--with-mysql=/path/to/mysql
```

После выполнения конфигурационного файла с помощью утилиты make можно строить и инсталлировать модуль (или программу httpd).

```
make
make install
```

Часть make install также модифицирует файл httpd.conf для того, чтобы активизировать разделяемые объектные файлы в момент запуска. Однако перед перезапуском сервера необходимо будет связать расширения php (в последующем примере это .php и .phps) с соответствующим MIME-типом.

```
Addtype application/x-httpd-php .php
Addtype application/x-httpd-php .phps
```

И последнее: проверьте конфигурационный файл и перезапустите сервер.

14.4.2. Работа с PHP

Не так давно написание CGI-сценариев можно было квалифицировать как работу средней или большой сложности. Сейчас все изменилось — независимо от своих языковых предпочтений, все сходятся во мнении, что язык PHP достаточно прост в использовании. Как видно из примера, приведенного ниже, с помощью нескольких операторов можно осуществить соединение с базой данных, выполнить оператор SQL и произвести выборку полученных результатов в файл с расширением .html для их дальнейшей обработки. Приведенный ниже пример прост в работе, но в нем не учтены ограничения, накладываемые различными СУБД на тип операторов SQL.

```
<HTML>
<HEAD>
<TITLE><B>PHP Example Page</Bx/TITLE>
</HEAD>
<BODY>
<H1> PHP </Hl>
<H2> Результаты, приведенные ниже получены динамически </H2>
<H2> с помощью PHP-запроса к базе данных MySQL. </H2>

<?
```

```
$mysql_handle = mysql_connect("localhost", "httpd", " "} or die
                ("Невозможно подключиться");
mysql_select_db ("ec_example") or die("Невозможно произвести выборку
                в базе данных");
$query = "SELECT * FROM ccard";
$result = mysql_query ($query) or die("Запрос не выполнен");
for ($i=0; $i <= mysql_num_rows($result) - 1; $i++)
     {
     if   (!mysql_data_seek($result,$i))
             {
             printf ("Невозможен поиск по строке %d\n", $i);
             continue;
             }
      if (! ($row=mysql_fetch_object($result) ) )
            continue;
      printf ("%s %s %s %s <BR>\n", $row->type, $row->name, $row-
              >number, $row->expires);
      }
      mysql_free_result($result);
?>
</BODY> </HTML>
```

Первая операция — с помощью функции mysql_connect () производится подключение к базе данных, размещенной на локальном узле. В качестве имени пользователя передается имя httpd. В этом случае пароля не требуется. После этого программа готова выбирать строки в цикле до тех пор, пока не будет достигнуто значение mysql_num_rows($result). Выборка строк производится с помощью функции mysql_fetch_object(). Полученный результат можно увидеть на рис. 14.4.

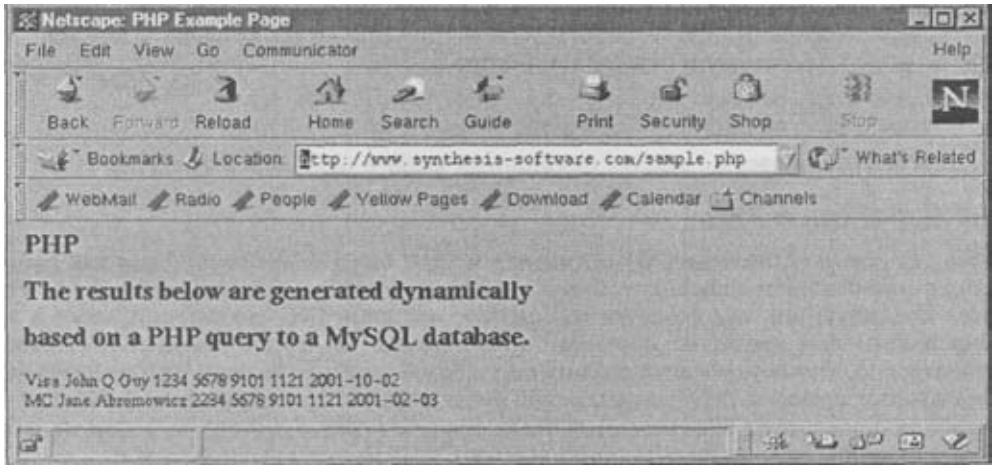

Рис. 14.4. Динамическая база данных

14.4.3. Вставка данных с помощью PHP

Вставка данных в языке PHP достаточно проста. Для этого текстовая строка запроса, содержащая оператор INSERT, сохраняется в текстовой переменной (см. ниже):

```
$query = "INSERT INTO some_table VALUES ('a', 'b', 'c', 'd', 'd')";
```

Будьте внимательны, если не хватает значений или они не соответствуют ограничениям, наложенным на таблицу, тогда ваш запрос не будет выполнен. Если вас все удовлетворяет, с помощью функции mysql_query () выполните запрос.

```
$order = mysql_query($query);
```

Очень трудно добиться выполнения запроса с первого раза и достаточно часто при выполнении запроса происходит ошибка. Эту ситуацию всегда нужно предусматривать для того, чтобы сценарий правильно отрабатывал в ситуации возникновения ошибок в процессе выполнения сценария.

```
if (!(mysql_query ($query))) die ("Ошибка в запросе.");
```

14.4.4. Выбор средства взаимодействия с базами данных

Без сомнения сильной стороной языка написания сценариев Perl является то, что это язык общего назначения. При работе как с ColdFusion, так и с PHP нужен определенный опыт в работе с операторами общего назначения. Кроме того, Perl имеет большую популярность, а PHP и ColdFusion распространены не так широко.

Пакет ColdFusion относительно дорог, но эти деньги быстро окупаются. Этот пакет тоже постепенно приобретает популярность. Даже если ваш узел находится на чужом сервере, скорее всего он может работать с ColdFusion. Напротив, о языке PHP многие разработчики вообще ничего не слышали.

Ближайшим конкурентом пакета ColdFusion является язык PHP. Оба работают приблизительно одинаково, т.е. методом включения своих кодов в HTML-код. Наибольшим преимуществом пакета ColdFusion является наличие графического интерфейса для создания Web-страниц. Наибольшим преимуществом языка PHP над пакетом Cold-Fusion является его цена (совершенно бесплатно в сравнении с одной тысячей долларов или даже больше). Обучение работе с PHP и ColdFusion процесс достаточно сложный. Сложнее, чем в случае с Perl, и в ситуации, когда необходимо проводить обучение персонала, это может стать решающим фактором.

Глава

15

ПРИМЕР КОММЕРЧЕСКОГО УЗЛА

В этой главе...

15.1. Введение

В данной главе будет построена модель коммерческого узла с помощью методов и средств разработки, о которых шла речь во всех предыдущих главах. Эта книга прежде всего направлена на создание собственных Internet-продуктов при наличии у читателя самого разного опыта. До этого книга описывала установку и обслуживание сервера Apache. В этой главе будет показано, что можно сделать с его помощью.

Этот узел можно рассматривать как демонстрацию концепции в действии длятех, кто собирается создать свой Internet-магазин. Узел создан с применением только общедоступного программного обеспечения, большую часть которого можно найти на сопровождающем компакт-диске. Кроме того, все имеющиеся здесь исходные тексты PHP и SQL-сценарии, можно дорабатывать для решения своих задач.

Этот материал в известной мере выходит за рамки темы администрирования сервера. Он уже затрагивает область Web-разработок. Авторский замысел заключается в следующем:

- Продемонстрировать гибкость и возможности общедоступного программного обеспечения.
- Предоставить читателю заготовку коммерческого узла, которую можно доработать для решения задач, стоящих перед читателем.
- Обратить внимание на самые характерные проблемы, с которыми обычно сталкиваются новички при разработке приложений.
- Обозначить новые задачи.

Заметьте, что в этой главе рассмотрение проблем графического дизайна и программирования HTML-кодов проводиться не будет. Предполагается, что по мере необходимости отлично спроектированные графические статические Web-страницы будут появляться по мановению волшебной палочки откуда-то извне. Основной упор здесь делается на создание динамического HTML-кода с помощью сервера Apache, модуля mod_php и СУБД MySQL. Инфраструктура, необходимая для осуществления этого задания, включает в себя:

- Базу данных для хранения информации о ваших товарах, покупателях и статистики о том, что эти покупатели покупают.
- Механизм поиска вбазе данных и возвращения результатов в виде Web-страниц.
- Историю покупок для прогнозирования последующих поставок.
- Структуру заказа.

Некоторые из продемонстрированных в предыдущих главах инструментов разработки были коммерческими (например Stronghold, ColdFusion). В этой же главе мы ограничимся только теми программными пакетами, которые можно получить бесплатно. На компакт-диске, сопровождающем это издание, обнаруживаем следующее программное обеспечение:

- Сервер Apache.
- Модуль mod_php.
- СУБД MySQL.

Здесь отсутствуют библиотека openSSL и модуль mod_ssl. Их совсем несложно получить (см. главу 8, "Безопасность"), но вследствие того, что вместе с данной книгой они могут быть проданы за пределы США, их нельзя было включать в этот CD- ROM.

15.2. Проектирование

Web-узел максимально упрощен. Страницы имеют связи, позволяющие осуществлять переход между ними, не заботясь особенно о передаче данных.

Существуют ситуации, в которых это не имеет значения. Например, можно просматривать записи в вашей продуктовой корзинке сколько угодно раз, но с экрана в корзинку за один раз может быть *добавлена* только одна покупка. Аналогично нельзя отобразить экран с товаром, не обратившись к каталогу или экрану поиска. Наконец, процедуры проверки линейны и достаточно просты.

15.2.1. Продуктовая корзинка

Продуктовая корзинка реализована в виде таблицы базы данных. Первичный ключ таблицы является сложным и состоит из полей cart_num и line_num (тип этих полей описан в этой главе в разделе "Проектирование базы данных"). Для целей ведения истории посещений каждый посетитель Web-узла будет иметь как минимум одну запись в таблице shopping_cart (запись создается при обращении к главной странице, назовем ее "нулевой записью") независимо от того, пуста корзинка или нет. После добавления первого товара в корзинку, нулевая запись удаляется. Нулевые записи можно отличить от обычных записей по 1) line_num = 0 и 2) inventory_num = 0.

15.2.2. Заказы

В этом проекте информация о заказе состоит из двух типов записей. В таблице order_master хранится по одной записи на каждый заказ. Каждая запись хранит ссылки на соответствующие записи в таблицах покупателей (customer) и адресов (address) вместе с датой заказа и доставки.

Детальная информация о названии и цене заказанного товара хранится в одной или нескольких записях таблицы orde r_detail. Это означает, что для того, чтобы просмотреть заказ, необходимо скомпоновать записи из таблиц order_detail и order_master в одну общую смысловую форму. Решение может показаться несколько сложным, но оно позволяет решать проблему ограничения количества детальной информации в заказе. Кроме того, это общепринятый подход, применяемый при решении таких задач.

15.3. Проектирование базы данных

В этом разделе мы обсудим две наиболее общие и одновременно влекущие за собой тяжелые последствия ошибки, обычно допускаемые в процессе проектирования базы данных: это отсутствие *первичного ключа* и отсутствие *нормализации*. Но для того, чтобы понимать, о чем здесь пойдет речь, необходимо по крайней мере иметь поверхностное знакомство с концепцией реляционных баз данных. К счастью, она достаточно проста.

15.3.1. Реляционные базы данных

В общем смысле, база данных — это метод хранения данных в компьютере, вплоть до набора слов в файле. Однако в последние десятилетия термин *база данных* стал применяться исключительно по отношению к наборам данных, в которых реализована реляционная модель. Реляционная база данных — это база данных, которая рассматривается пользователями как набор отдельных таблиц. Таблицы состоят из *строк* и *столбцов*. Каждая строка является *записью, а* каждый столбец — *полем*.

Рассмотрим таблицу Address. Она содержит пять столбцов:

```
addrt attn: street city state zip
```

Во многих случаях желательно, чтобы данные имели относительно свободную форму (например, столбец `street` может содержать буквы и цифры, заданные в совершенно произвольном порядке), но в других случаях данные можно ограничить определенным типом (целые числа или календарная дата). СУБД позволяет задавать тип данных, хранящихся в определенном столбце во время создания таблицы:

```
CREATE TABLE address
(
  addr_id INT UNSIGNED NOT NUL AUTO_INCREMENT PRIMARY KEY,
  attn VARCHAR(30),                # имя покупателя
  street1 VARCHAR(30) NOT NULL,  # адрес
  street2 VARCHAR(30),             # номер квартиры или еще что-то
  city VARCHAR(20) NOT NULL,       # город
  state CHAR(2) NOT NULL,          # две буквы аббревиатуры штата
  zip VARCHAR(10) NOT NULL,        # почтовый индекс
}
```

15.3.2. Ошибка №1: отсутствует первичный ключ

Эта ошибка тесно связана с первой, допускаемой новичками при разработке базы данных, — отсутствием первичного ключа. *Первичным ключом* называется поле или группа полей, служащих для уникальной идентификации данных в заданной таблице. В приведенном выше примере первичными ключами являются поля order# и addr# соответственно. Обратите внимание, что эти ключи совсем необязательно *показывать* конечному пользователю Web-узла, *они существуют как первичные ключи только на сервере*. Технически совсем нетрудно создать таблицу или даже таблицы, которые не имеют первичного ключа, можно даже создать вполне работающую систему, использующую такие таблицы. Но это чрезвычайно грубая ошибка проектирования.

В некоторых случаях отличным кандидатом на роль первичного ключа являются данные, хранящиеся в самой таблице. Например, номера ISBN[1] являются уникальными и при создании таблицы, хранящей информацию о книгах, отлично могут служить первичным ключом. Нет причины, которая не позволила бы использовать хранящиеся данные в качестве первичного ключа.

[1] Такой номер можно найти на обложке этой книги.

Кроме того, есть возможность создавать первичные ключи, состоящие из двух и более полей. Например, при проектировании таблицы, хранящей данные об аудиотеке, в этих целях можно использовать вместе наименование производителя ("Sony", "Pioneer") и товарный номер (VX132, QZX820). В результате получится *составной первичный ключ*. Очевидно, что нет никакой гарантии того, что товарные номера "Sony", "Pioneer" не будут совпадать. Вероятность такого совпадения мала, но мы отлично знаем, что человек предполагает, а Бог располагает.

Еще один метод создания первичного ключа выходит на арену тогда, когда среди хранимых данных нет очевидных кандидатов на роль первичного ключа. В таких случаях можно просто использовать номер. Так, например, для таблицы заказов order поле order# служит первичным ключом. Номер может быть сгенерирован СУБД автоматически или на уровне приложения. При этом необходимо позаботится только о том, чтобы значения не повторялись.

15.3.3. Внешние ключи

Каждая строка главной таблицы order включает следующие столбцы (поля):

```
cust# shipto billto
```

Они все имеют целый тип и все являются первичными ключами *некой другой таблицы*. Поле cust# имеет отношение к первичному ключу таблицы, хранящей информацию о покупателях (разговор о ней впереди), а поля shipto и billto содержат числа, связанные с записями в таблице Address. Поля в таблице A, относящиеся к первичному ключу таблицы B, называются *внешними ключами*. С помощью значения cust#, которое хранится в таблице order, можно просмотреть соответствующую запись в таблице customer, т.е. определить, что заказ 12345 поставляется Бобу Джонсу. С помощью полей shipto и billto в таблице address можно определить, что заказ будет доставлен Билу Джонсу по адресу 123 4th St. (ул. 4, д. 123), а счет предъявлен Сью Смит по адресу 345 6th St (ул. 6, д. 345) и т.д.

В общем, проекты, использующие внешние ключи для ссылки на данные, хранящиеся в других таблицах, а не копии этих данных, можно охарактеризовать как удачные.

15.3.4. Ошибка № 2: база данных не поддается нормализации

Но безудержное и последовательное использование первичных ключей, кроме всего прочего, еще имеет большое значение и для предотвращения другой ошибки, которая может быть допущена при проектировании баз данных, невозможности ее *нормализовать*. Нормализация данных — это в известной мере формальный процесс усовершенствования формата данных, в результате которого формат хранения данных соответствует строго определенным точным ограничениям. Во времена моей учебы в школе насчитывалось 7 признанных форм нормализации (INF, 2NF, 3NF, Boyce-Codd NF, 4NF, PJ/NF). Нет никаких сомнений, что за время, прошедшее с тех пор, были изобретены новые формы нормализации. Три первые формы нормализации имеют непосредственное и очевидное преимущество для разрабатываемых приложений. Я подозреваю, что оставшиеся формы были придуманы вследствие того, что у кого-то было слишком много свободного времени.

Интуитивно можно сказать, что нормализация баз данных состоит в разбиении данных на элементарные цепочки и сохранении этих данных в отдельных таблицах. Таблица заказов (order), о которой шла речь выше, может служить хорошим примером этого — использование вышеупомянутых внешних ключей позволяет хранить массу информации в двух полях. Для новичков всегда существует соблазн создать одну большую таблицу, которая содержит все необходимые данные. Например, таблица, хранящая информацию о заказах, должна содержать пять или шесть полей с адресом плательщика (имя, адрес, город,

штат, индекс), еще пять или шесть полей, содержащих адреса поставки, и море из двадцати или тридцати полей самого разного содержания (номер, описание товара, цена на товар, налоги на продажу, итоговые данные и т.д.).

Основной проблемой, возникающей при таком решении, является то, что база данных и операционная система будут вынуждены отдать вам под использование все это пространство, даже если при этом будут сохраняться данные в одном или двух полях. Все пространство быстро заполнится записями, которые в большинстве своем будут пустыми. Это снижает производительность и очень быстро приводит к тому, что при попытке создать программное приложение на основе столь запутанной структуры таблицы значения данных будут разрушены и потеряют смысл.

Гораздо лучше иметь большой набор относительно маленьких таблиц, каждая строка которых уникально идентифицирована первичным ключом. Значения данных, хранящиеся в таблицах, не должны зависеть одна от другой так, чтобы не возникала необходимость создавать программное приложение, которое будет вносить изменения в одну таблицу при внесении изменения в другую. Все внесенные изменения нужно держать в уме в то время, когда другой сотрудник, которому, вероятно, придется дорабатывать вашу программу, может этого просто не знать. Обязательное разбиение на столь элементарные фрагменты сначала может показаться излишним, но методом проб и ошибок к этому решению предстоит прийти. Это позволит создать программу, которая сможет отображать данные в более привлекательном формате. На практике ваша база данных должна обладать следующими свойствами:

- Должен присутствовать первичный ключ. Очень хорошо, если он будет состоять из одного поля, предпочтительно это должно быть число, но при необходимости может состоять из двух или более полей. По возможности создавайте в базе данных первичные ключи.
- Данные, хранящиеся в полях должны быть *атомарными;* т.е. не должно возникать ситуации, в которой необходимо разбивать поля на части. Если, например, у вас есть одно поле под названием address, в котором в длинной строке хранится имя, адрес, город, штат и индекс, ваши данные *никак* нельзя назвать атомарными, так как может потребоваться выбрать из этого поля почтовый индекс. Это неудобно.
- В базе данных не должно быть полей, значение которых определяется значением, хранящимся в других полях базы данных (это принцип *взаимной независимости).* Наличие таких полей — грубая ошибка, допущенная на этапе проектирования структуры базы данных, т.к. трудно обеспечить выполнение ограничения, когда поле В должно изменяться в то время, когда меняется поле А. Например, если запись о заказе состоит из нескольких строк, полная стоимость заказа должна пересчитываться при каждом отображении заказа, а не храниться как поле таблицы.

Желающим узнать побольше о теории баз данных можно порекомендовать книгу *"Введение в системы баз данных",* том 7, Си. Дж. Дейта, которая сейчас считается классикой.

Сценарии, приведенные ниже (они имеются на прилагаемом компакт-диске), предназначены для создания базы данных ее от:

```
CREATE TABLE ccard
 (
card_id   INT UNSIGNED NOT NULL AUTO_INCREMENT PRIMARY KEY,
cust_id   INT NOT NULL,# Ссылка на соответствующего покупателя
type      VARCHAR(20)  NOT  NULL,# Тип  карточки Visa,  MasterCard,  ...
name      VARCHAR(40)  NOT  NULL,# Имя владельца карточки
number    VARCHAR(20)  NOT  NULL,# Номер карточки
```

```
expires date NOT NULL# Срок действия карточки
)
# Отметим, что номер счета и чека могут быть вместе.
# Использованы в качестве первичного ключа, но, с моей точки зрения,
        это немного тяжеловесное решение.
CREATE TABLE check
 (
check_id  INT UNSIGNED NOT NULL AUTO_INCREMENT PRIMARY KEY,
cust_id      INT,# Покупатель
check_num    SMALLINT NOT NULL,# Номер чека
name         VARCHAR (20)  , # Имя на чеке
routing_num CHAR(12),# Код банка  (что-то вроде МФО)
acct_num     CHAR(12)# Номер счета
)

CREATE TABLE customer
 (
cust_id      INT UNSIGNED NOT NULL AUTO_INCREMENT PRIMARY KEY,
first_name VARCHAR (15) NOT NULL, I Имя
middlell    VARCHAR (15),# Отчество
last_name  VARCHAR(IS) NOT NULL,! Фамилия
title   VARCHAR(4),# Титул Mr, Mrs, Dr.
suffix   VARCHAR(5)# Суффиксы Ph.D., Jr., Ill,
)

# order_num + line_num --> Первичный ключ
CREATE TABLE order_line
 (
order_num  INT UNSIGNED NOT NULL,# Связь с главной записью заказа
line_num   SMALLINT NOT NULL,# Номер строки -
quantity     SMALLINT NOT NULL,#
inventory_num INT UNSIGNED NOT NULL,# Указатель на таблицу товаров
price        DOUBLE NOT NULL# Цена
)

CREATE TABLE order_naster
 (
order_num   INT UNSIGNED NOT NULL AUTO_INCREMENT PRIMARY KEY,
cust_id     INT UNSIGNED NOT NULL,# Указатель на таблицу customer
shipto      INT UNSIGNED NOT NULL,# Указатель на таблицу address
billto       INT UNSIGNED NOT NULL,# Указатель на таблицу address i
             (Вероятно то же, что и shipto)
ordered DATE NOT NULL,  # Дата заказа
paid DATE NOT NULL,     # Дата оплаты заказа
shipped DATE NOT NULL,  # Дата доставки
prototype  SMALLINT  NOT  NULL,  I  0  =  Не  уплачено,  1  =  кредитная
          карточка, 2
# = Оплата по чеку,    ... (?)
check  INT,  # Указатель на таблицу check
ccard  INT   # Указатель на таблицу ccard
)

CREATE TABLE product
 (
inventory_num  INT UNSIGNED NOT NULL AUTO_INCREMENT PRIMARY KEY,
product_num  VARCHAR(20)  NOT  NULL,#  Товарный номер производителя
manufacturer VARCHAR(20),  # Производитель (Sony,  Ford,  ...)
name         VARCHAR (20)  NOT  NULL,  # Название товара  (для
```

```
# использования в описании)
description VARCHAR(80) NOT NULL, # Английское описание в свободной
# форме не больше одной строки
category   VARCHAR(IO) NOT NULL, # Категория учета. Требуется,
subcat1   VARCHAR (10), # Подкатегория 1—ого уровня. Факультативно.
subcat2   VARCHAR(IO), # Подкатегория 2—ого уровня. Факультативно.
the_hype   TEXT, # Краткая форма для использования в рекламе
# ads
image      VARCHAR(255)  , # Абсолютный путь к  файлу с рисунком
in_stock     INT UNSIGNED NOT NULL, # Количество на складе
price       DOUBLE # Цена товара
)
```

15.3.5. Ввод данных

Предположим, что реальные данные для базы данных уже созданы добрыми гномами, и настал момент запроса данных из базы данных, имеющей идеальную структуру. Однако, те из чи тателей, кто любит реальность, могут извлечь полезный урок из следующего совета.

Насколько это возможно, необходимо лишать пользователей возможности вводить данные непосредственно с клавиатуры. Никогда два человека не опишут один и тот же объект одинаково. Если процесс ввода данных не будет ограниченным и формализованным, потом может оказаться невозможным получить доступ к имеющимся данным. Предположим, что вы организовали электронный магазин по продаже упряжи и продаете двадцать видов седел. Предположим также, что для решения задачи ввода информации был нанят персонал, которому вы предоставили возможность ввода данных в свободном виде. Существует большая вероятность того, что введенные данные будут иметь вид: "Прикрасные коные седла английского стиля. Из кожи кримового цвта, размера среднего размера."

Проблемы описания данных очень затруднены изменчивостью современного английского языка. Только крошечный процент населения сможет напечатать одно или сразу несколько слов безошибочно. Пока не будет убедительно доказано, что ваши сотрудники и ваши покупатели делают одинаковые ошибки, вам придется осуществлять ввод информации с помощью ниспадающих меню. Каждый раз, когда вы уменьшаете объем рутинной работы по вводу данных с клавиатуры, вы освобождаете себя от необходимости внесения исправлений во вводимые данные.

15.4. Пример узла

В этом разделе мы рассмотрим структуру коммерческого узла на конкретном примере. Поскольку большая часть кода (особенно кнопки) повторяется повсеместно, во всей своей полноте сценарии обсуждаться не будут. Если в озник интерес или есть необходимость модифицировать этот программный код для своих собственных нужд, его можно найти на компакт-диске, прилагаемом к этой книге. Приведенный здесь Web-узел состоит из семи сценариев, приведенных в табл. 15.1.

Таблица 15.1. Сценарии, задействованные в примере Web-узла

Сценарий	*Описание*
index.php	Главная страница.
catalog.php	Автоматически генерирует страницу с каталогом, содержащую все продаваемые в Internet-магазине товары, отсортированные по категориям.

Окончание табл. 15.1

Сценарий	***Описание***
result.php	Отображает результаты поиска.
item.php	Отображает информацию об отдельных товарах и их изображения (если имеются в наличии рисунки).
cart.php	Отображает и/или добавляет содержимое в продуктовую корзинку.
checkout. php	Принимает информацию, необходимую для создания заказа.
mk_order. php	Создание записи о заказе.

Следует обратить внимание читателя на то, что представленный здесь программный код является общим наброском. Чтобы этот код можно было использовать на создаваемом вами Web-узле, его необходимо немного доработать. Одни изменения очевидны, другие — не очень. Может быть доработана любая страница. Читатель может использовать исходный текст в своей работе, вносить изменения, распространять его по своему усмотрению. Если у вас получилась удачная доработка и есть желание поделиться ею со мной, пришлите ее, пожалуйста, по адресу: *shawkins@synthesis-software.com.* Я с удовольствием размещу ее на Web-узле. Последнюю версию можно найти на узле *http://www.synthesis-sofrware.com/apachejbook/updates.*

15.4.1. Сценарий index.php

Сценарий index. php генерирует небольшую страницу с пятью кнопками: Search, Catalog, View/Modify, Cart и Checkout. Она изображена на рис. 15.1.

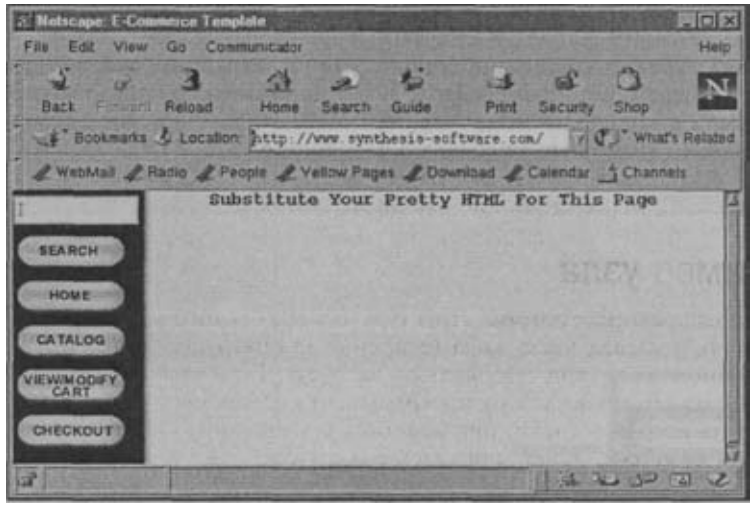

Рис. 15.1. Сценарий index.php

Первая часть представляет собой сегмент, обрабатывающий поле cart_num. Переменная cart_num используется в качестве индекса таблицы shopping_cart. Она передается параметром в сценарии в момент его запуска. Представленный ниже программный код генерирует новый номер корзинки, если таковой отсутствует.

```
if($cart_num == "")
{
```

```
$mysql_handle = mysql_pconnect("localhost", "httpd", "") or
                die("Соединение невозможно");
mysql_select_db("ecom") or die("Невозможно выбрать базу данных");
$result = mysql_query("SELECT MAX(cart_num) AS maximum FROM cart")
          or die("Невозможно запросить cart_num"};
if(!($row=mysql_fetch_object($result))) die("Невозможно выбрать
          cart_num");
$cart_num = $row->maximum + 1;
mysql_free_result($result);
}
```

Обратите внимание, что вместо функции mysql_connect здесь применяется функция mysql_pconnect. Устойчивое соединение помогает сэкономить немного времени на повторном подключении к базе данных.

Номер cart_num после создания будет использоваться во всех последующих модификациях сценария:

```
print("<TD><A  HREF=\"catalog.php?cart_num=$cart_num\");
```

Выражение cart_num=$cart_num в тэге HREF позволяет передавать значение, хранящееся в переменной $cart_num, вызванному сценарию (в данном случае это сценарий catalog, php) как переменную cart_num. Более того, пары variable=$value можно добавлять, конкатенируя их символом "&".

Этим можно воспользоваться, если вам потребуется добавить какое-нибудь содержание в правую часть экрана. Кроме того, метод генерации нового номера продуктовой корзинки, описанный выше, вполне подходит для узлов с низким трафиком, и не сможет гарантировать создание уникального cart_num в случае одновременного доступа к узлу нескольких пользователей.

15.4.2. Сценарий catalog.php

Каталожный экран создает единственную HTML-страницу, содержащую данные из базы данных product. Как видно из рис. 15.2, наименования товаров отображаются в виде гиперсвязей к сценарию item.php, что обеспечивает полную информацию о товаре и позволяет положить нужный товар в корзинку.

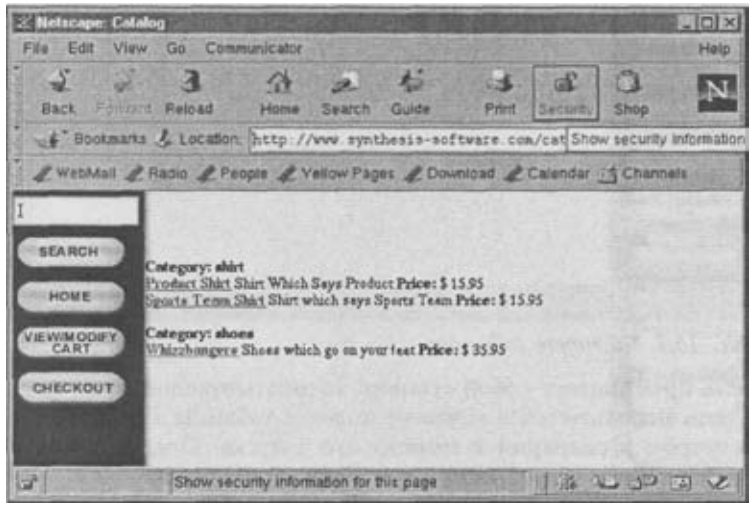

Рис. 15.2. Страница с каталогом

Основная функция страницы реализована в виде единственного цикла FOR, считывающего все записи из таблицы product, сортируя их по категории товара. Распечатывается наименование, описание и цена (имя представляет собой ссылку на сценарий item). Когда сценарий ищет новую категорию, он делает несколько пробелов и выводит на экран наименование категории:

```
# Получить все товары, входящие в определенную категорию.
$query = " SELECT inventory_num, product_num, name, category,
          description, price FROM product ORDER BY category";
$result = mysql_query ($query) or die("Ошибка в запросе");
$cnt = mysql_num_rows ($result);
for ($j=0; $j <=mysql_num_rows ($result) - 1 ; $j++ )
        {
        if ( !mysql_data_seek($result, $j))
        {
        printf("Невозможен поиск по строке %d\n", $j);
        continue;
        }
        if ( ! ($row=mysql_fetch_object ($result)))
        continue;
        if ( $current_category != $row->category)
        {
        $current_category=$row->category;
        printf (  "<BR><B>Категория:  %s   </BXBR>\n" ,
                  $row->category);
        }
        print("<A  HREF=\"  item.php?inv_num=$row->inventory_num&cart_num=
                $cart_num\ "><B>$row->name</Bx/A>");
        printf(" %s  <B>Цена: </B>   $%4.2f  <BR> " ,
                  $row->description,   $row->price);
        }
```

15.4.3. Сценарий result.php

Сценарий result.php может быть вызван практически из любой точки на узле. Для этого необходимо ввести запрос и щелкнуть по клавише Search. В результате получим экран, представленный на рис. 15.3.

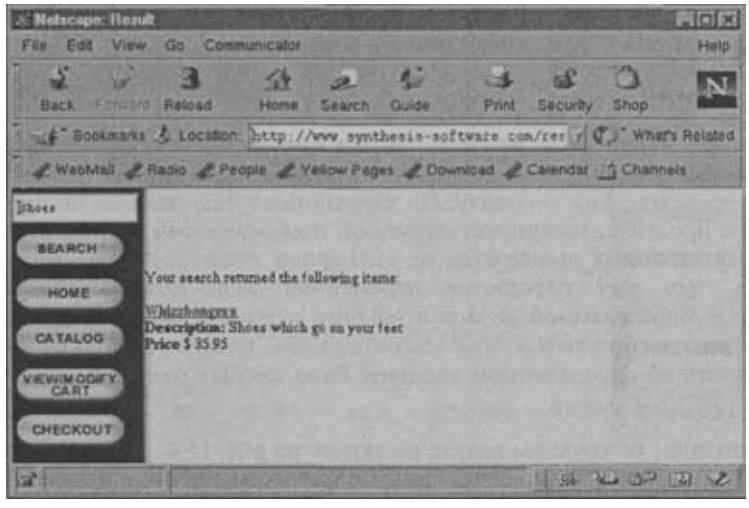

Рис. 15.3. Результаты поиска

Дополнительно к переменной $cart_num (которая есть везде) этот сценарий также принимает параметр $srch_str, в котором хранится все, что было введено в соответствующую область до того, как пользователь щелкнул по кнопке Search.

Для анализа строки сервер использует функцию strtok() для расчленения значения $srch_str на отдельные слова, рассматривая символ пробела как разделитель.

```
$token=strtok($srch_str,"  ");
```

При первоначальном вызове функции strtok () аргумент передавать не требуется.

```
$token=strtok("  ");  # Пустые запросы не возвращают результатов,
if($srch_str=="")
    {
    $where_clause="  WHERE  1=0";
    }
```

Все возможные лексемы составляются в одно длинное выражение where, которое производит поиск наименования, категории, подкатегорий.

```
while ($token)
    {
    if($where_clause  =="")#  1—ая  лексема
        {
        $where_clause=" WHERE name LIKE  '%$token%'  ";
        }
    else # Поиск более, чем одной лексемы
        {
        $where_clause=" $where_clause OR name LIKE '%$token%'  ";
        }
    $where_clause=" $where_clause OR category LIKE '%$token%' ";
    $where_clause=" $where_clause OR subcat1 LIKE '%$token%' ";
    $where_clause=" $where_clause OR subcat2 LIKE '%$token%' ";
    $where_clause=" $where_clause OR the_hype LIKE '%$token%' ";
    $token=strtok ( "  " };
    }
```

В зависимости от формата данных конструкцию выражения where_clause **можно сделать привлекательной. В настоящем виде это выражение ищет в базе данных все, что удовлетворяет как минимум одной лексеме из строки поиска. Очень важно, чтобы пользователь аккуратно вводил запросы, так как слово the в запросе будет соответствовать всем полям** the_hype, **имеющимся в базе данных.**

15.4.4. Сценарий item.php

Очевидно, что экран item является самым популярным экраном на узле. Он вызывается из любого экрана каталога или результирующего экрана, когда пользователь щелкает по определенному элементу. Он запрашивает базу данных на предмет полной информации о продукте, генерирует сценарий, изображенный на рис. 15.4.

Так как эта страница вызывается из сценариев result.php и catalog.php, мы предполагаем, что ему передается переменная $inv_num, связанная с полем inventory_num определенной записи в таблице product. Именно это значение применяется для построения выражения where_clause, которое используется для выбора всей информации об определенном элементе базы данных product.

```
$where_clause=" WHERE inventory_num = '$inv_num'  ";
```

Вся информация, которую вы видите на экране на рис. 15.4, за исключением рисунка, получена из базы данных. Размещение файла с рисунком хранится в поле. При создании

страницы item, сценарий сначала проверяет существование файла, в котором хранится рисунок, а потом строит страницу с использованием рисунка либо без него.

```
if(file_exists("$row->image"))
        {
print("<IMG SRC=\"$row->image\" ALIGN=\"LEFT\ " HSPACE=\"30\"
VSPACE=\"40\"  >");
        }
```

Более того, это единственный экран, позволяющий добавить продукты в корзинку.

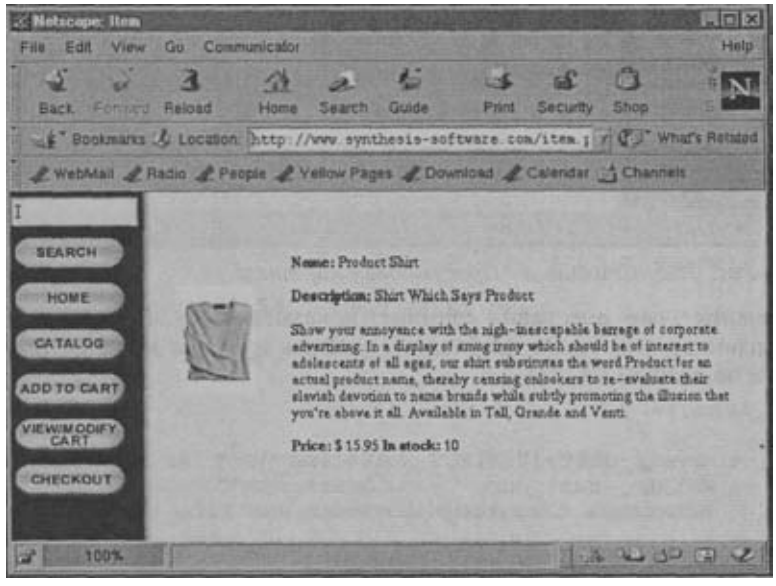

Рис. 15.4. Страница "Item"

15.4.5. Сценарий cart.php

Сценарий cart.php предназначен для добавления нового товара в продуктовую корзинку или отображения того, что в ней уже имеется. На первый взгляд в нем нет ничего особенного. Он изображен на рис. 15.5.

Но это только на первый взгляд. Сценарий экрана выполняет множество функций. Во-первых, он проверяет была ли в качестве параметра передана переменная $new_item. Если была, то необходимо добавить в корзинку определенный товар.

Товары, находящиеся внутри корзинки, последовательно пронумерованы. При добавлении новых продуктов, сценарий берет наибольший последовательный номер из уже находящихся в продуктовой корзинке товаров, прибавляет к нему единицу и сохраняет полученный результат в переменной $line_num. В таблицу cart добавлены столбцы cart_num, line_num и new_item (напомню, что new_item является индексом базы данных product).

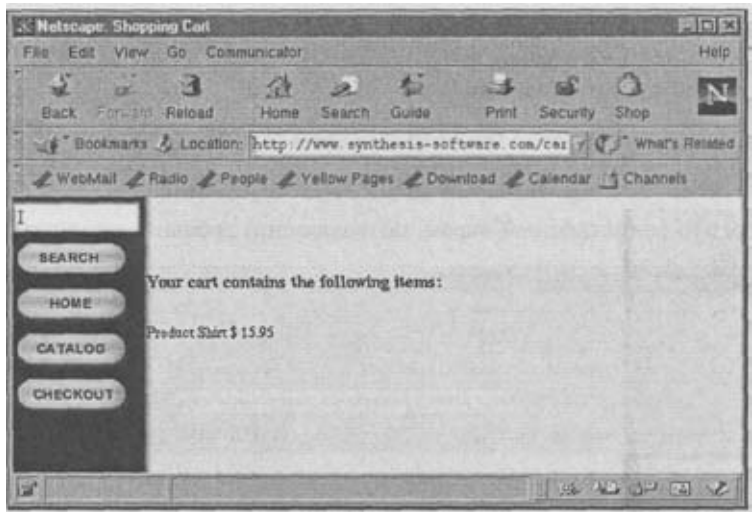

Рис. 15.5. Страница "Продуктовая корзинка"

Если значение line_num равно единице (показывая, что это первый товар в продуктовой корзинке), нам также необходимо удалить нулевую запись о новом покупателе, впервые зашедшем на узел.

```
if($new_item != " "}
        {
$result = mysql_query("SELECT MAX(line_num) AS maximum FROM cart
          WHERE cart_num = '$cart_num'") or die("Невозможно
          вычислить следующий максимальный line_num");

if(!($row=mysql_fetch_object($result)))
die("Невозможно выбрать line_num");

      $line_num=$row->maximum + 1;
$query = "INSERT INTO cart VALUES
 ('$cart_num','$line_num','','$new_item','')";
$result = mysql_query($query)
or die("Невозможно добавить в корзинку.");
      if($line_num==1)
      {
      result = mysql_query("DELETE FROM cart WHERE
               cart_num='$cart_num' AND line_num='0'")
or print("Error: Невозможно удалить пустую строку. <BR>\n");
      }
}
```

Независимо от цели, с которой был вызван сценарий (чтобы добавить новое значение или просто отобразить содержимое корзинки), он обязательно должен отображать содержимое корзинки.

15.4.6. Сценарий checkout.php

Этот сценарий вызывается из любой точки узла щелчком по клавише chec kout. Как видно на рис. 15.6, это большой сценарий с большим количеством вводимой информации.

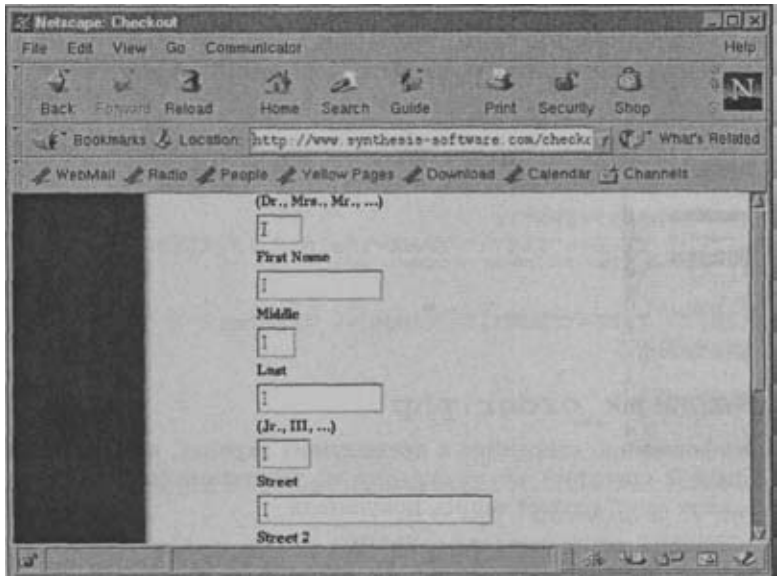

Рис. 15.6. Страница "Checkout"

Основной задачей сценария checkout.php является сбор информации, используемой при создании заказа. У нас уже есть набор товаров для создания заказа, находящихся в таблице cart, номер корзинки задается переменной $cart_num. После этого необходимо получить информацию о покупателе.

На этом экране покупатель вводит обычную информацию о себе (имя, адрес, город, штат, почтовый индекс), после этого щелкает по кнопке Send, находящейся вверху экрана. Вся введенная информация запоминается в отдельных переменных ($name, $streetl, $street2) и передается в сценарий mk_order.

```
print("<FORM ACTION=\ "mk_order .php?cart_num=$cart_num\ " METHOD=\ "
       POST\ ">"  );
print("<B> Title <B> (Dr., Mrs., Mr., ...) <BR>");
print  ("<INPUT TYPE=\"TEXT\" NAME=\"title\" MAXLENGTH=\ "4\"
        SIZE=\"4\"
><BR>");
print  ("<B> First Name <BXBR>");
print  ( "<INPUT TYPE=\"TEXT\" NAME=\ " first_name\ "
         MAXLENGTH=\ " 15\ "
SIZE=\"15\"  ><BR>"  );
print  ("<B> Middle Initial <BXBR>"};
print  ("<INPUT TYPE=\"TEXT\" NAME=\ "middle\ " MAXLENGTH=\ " 15\ "
SIZE=\"15\"  ><BR>"  );
print  ("<B> Last Name <BXBR>");
print  ("<INPUT TYPE=\"TEXT\" NAME=\ " last_name\ " MAXLENGTH=\ " 15\ "
SIZE=\"15\"   ><BR>");
print("<B> Suffix <B> (Jr., Ill, ...) <BR>");
print  ("<INPUT TYPE=\"TEXT\" NAME=\ " suffixN " MAXLENGTH=\ " 5\ "
SIZE=\"5\"   ><BR>");
print   ("<B>  Street  <BXBR>");
print  ("<INPUT TYPE=\"TEXT\" NAME=\ " streetlV "  MAXLENGTH= \ " 30 \ "
SIZE=\"30\"   ><BR>");
print  ("<B> Street 2 <B><BR>");
print  ("<INPUT TYPE=\"TEXT\" NAME=\ " street2\ "  MAXLENGTH=\ " 30\ "
```

```
SIZE=\"30\"  ><BR>");
print  ("<B> City <BXBR>");
print ("<INPUT TYPE=\"TEXT\" NAME=\ " city\ "  MAXLENGTH=\ " 20\ "
SIZE=\"20\"   ><BR>");
print  ("<B> State <B><BR>");
print ("<INPUT TYPE=\"TEXT\" NAME=\ " state\ " MAXLENGTH=\ "2\ "
        SIZE=\"2\"
><BR>");
print  ("<B> Zip <BXBR>");
print ("<INPUT TYPE=\"TEXT\" NAME=\"zip\"  MAXLENGTH=\ " 10\ "
        SIZE=\"10\"
><BR>");
print ("<INPUT TYPE=\"SUBMIT\" NAME=\ " Submit \ " VALUE=\"Send
        It\">");
```

15.4.7. Сценарий mk_order.php

Наконец, информация, собранная в предыдущих экранах, используется для создания заказа. Сначала сценарий mk_order.php на основании данных, полученных из сценария checkout .php, создает запись покупателя.

```
$query = "INSERT INTO customer VALUES
(' ' , '$first_name', '$middle', '$last_name', '$title', '$suffix')";
$customer = mysql_query($query) or die ("Нельзя создать запись о
            покупателе");
$cust_id = mysql_insert_id();
```

Так как поле cust_id при создании таблицы customer было объявлено первичным ключом с автоинкрементом, для возвращения идентификатора нового покупателя можно использовать функцию mysql_insert_id(), в которой новое значение идентификатора cust_id создается оператором insert.

Поле cust_id используется при создании записи об адресе.

```
$full_name = "$first_name $last_name";
$query = "INSERT INTO address VALUES
(' ', '$full_name', '$street1', '$street2', '$city', '$state',
      '$zip' )";
$address = mysql_query($query) or die ("Нельзя создать
           идентификатор адреса");
$addr_id = mysql_insert_id();
```

Переменная $addr_id используется при создании записи заказа.

```
$today = date("Ymd", time());
$query = "INSERT INTO orderjnaster VALUES
(' ', '$cust_id', '$addr_id', '$addr_id', '$today', '$today',
      '$today', '1', ' '
, ' ')";
$order = mysql_query($query) or die ("Fatal: mk_order .php:
         Невозможно создать order_master.");
$order_num = mysql_insert_id();
```

Наконец, самое интересное, что поле $order_num используется при создании деталировки заказа. Напомню, что одной записи из таблицы order_master может соответствовать произвольное количество записей о деталях заказа, по одной на каждый покупаемый товар. В настоящее время количество товара, задаваемое каждой строкой, равно единице, хотя вполне допустимо указание в структуре таблицы order_detail определенного количества товара одного наименования.

Сначала мы ищем все товары в продуктовой корзинке. Каждая из них будет использована для создания одной записи в таблице order_detail. Так как цены не хранятся в корзинке, их необходимо подтянуть для того, чтобы сохранить их в записи деталировки. Заметим, что это не нарушит правила функциональной независимости потому, что цена в будущем может изменяться, а мы *не* хотим модифицировать цену, по которой был продан товар. Наконец, необходимо подчеркнуть, что первичный ключ записей в таблице order_detail является сложным, состоящим из полей order_num и line_num.

```
$cart = mysgl_query("SELECT * FROM cart WHERE cart_num
        '$cart_num'") or die("Fatal: checkout.php: Невозможно
        получить информацию о корзинке.");
$cnt = mysql_num_rows($cart);
for ($j=0; $j <= mysql_num_rows($cart) - 1; $j++ )
    if (!mysql_data_seek($cart, $j))
        {
        printf("Невозможно выбрать строку %d\n", $j); continue;
        }
    if(!($cart_row=mysql_fetch_object($cart))) continue;
$query = " SELECT price AS line_price FROM product WHERE
        inventory_num = '$cart_row->inventory_num' ";
$price_rslt = mysql_query($query) or die("Fatal: mk_order.php:
        Невозможно создать таблицу order_master.");
if(!($row=mysql_fetch_object($price_rslt))) die("Невозможно
        выбрать цену.");
$line_price = $row->line_price;
$query = "INSERT INTO "order_line VALUES ('$order_num','$cart_row-
        >line_num','!','$cart_row->inventory_num','$line_price')";
$line_rslt = mysql_query($query) or die("Fatal: mk_order.php:
        Невозможно создать запись о заказе.");
```

Если все проходит отлично, сценарий выводит сообщение "Ваш заказ принят". На практике неплохо вывести на экран сам заказ для того, чтобы дать возможность покупателю еще раз проверить его и нажать клавишу <l Accept> для связи с защищенным экраном (здесь не включен по причине лицензионных обязательств), на который выводится информация с кредитной карточки.

Часть IV

Приложения

В *этой части...*

Приложение

A

ОСНОВНЫЕ ДИРЕКТИВЫ

В этом *приложении…*

A.1. Введение

Этот раздел содержит сведения о синтаксисе и использовании основных директив сервера Apache. Основные концепции и использование директив можно найти в предметном указателе.

A.2. Основные директивы

A.2.1. Директива AccessConfig

Синтаксис: AccessConfig *имя-файла*
Умолчание: AccessConfig conf/access.conf
Контекст: конфигурирование сервера, виртуальный узел

Эта директива используется для объявления файла, содержащего дополнительные директивы. Сервер считывает этот файл после завершения чтения файла ResourceConfig. Имя файла задается относительно каталога ServerRoot.

Чтобы сервер Apache считывал некоторое количество дополнительных директив из файла site_specific, хранящегося в каталоге conf, включите в файл Resource_Config следующую команду:

```
AccessConfig conf/site specific
```

Чтобы отключить эту возможность, введите команду:

```
AccessConfig /dev/null
```

A.2.2. Директива AccessFileName

Синтаксис: AccessFileName *имя-файла*
Умолчание: AccessFileName .htaccess
Контекст: конфигурирование сервера, виртуальный узел

Эта директива задает имя файла, в котором хранится список управления доступом. Сервер будет искать файл с таким именем в каждом каталоге, находящемся на пути к документу, предполагая, что файлы управления доступом для этого каталога запущены.

Чтобы задать имя файла, хранящего спецификации доступа Apache как .htaccess, необходимо задать директиву следующего вида:

```
AccessFileName .htaccess
```

' Например, когда сервер получает запрос файлу /usr/local/w
тиве, то чтобы проверить права и вернуть требующийся файл, сервер будет проверять последовательно директивы, имеющиеся в следующих файлах:

```
/usr/.htaccess
/usr/local/.htaccess
/usr/local/web/.htaccess
```

Очевидно, что последовательная проверка всех этих каталогов влечет за собой деградацию производительности. Эту возможность можно глобально отключить следующей директивой:

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

A.2.3. Директива AddModule

Синтаксис: AddModule *модуль модуль . . .*
Контекст: конфигурирование сервера
Совместимость: директива AddModule имеется только для сервера Apache версии 1.2 и выше

Эта директива может быть использована для активизации модулей, которые были скомпилированы, но в данный момент не используются. Чтобы очистить список активных модулей сервера, принятый по умолчанию, воспользуйтесь директивой ClearModuleList.

Пример

Для активизации использования модуля mod_peri задайте следующую директиву:

```
AddModulemod_perl
```

A.2.4. Директива AllowOverride

Синтаксис: AllowOverride *override override . . .*
Умолчание: AllowOverride All
Контекст: каталог

Эта директива влияет на действие файла .htaccess или любого другого файла управления доступом, определенного директивой AccessFileName. В зависимости от значения переменной override эта директива будет влиять на управление доступом одним из перечисленных в табл. АЛ способов.

Таблица А.1. Влияние на управление доступом значения переменной override

Значение переменной	*Override*	*Действие*
None		Сервер не считывает файл.
All		Сервер будет реагировать на все директивы.
AuthConfig		Активизирует использование директив санкционирования доступа (AuthDBMGroupFile, AuthDBMUserFile, AuthGroupFile, AuthName, AuthType, AuthUserFile, require и Т.Д.).
FileInfo		Активизирует использование директив, контролирующих ТИПЫ документов (AddEncoding, AddLanguage, AddType, DefaultType, ErrorDocument, LanguagePriority).
Indexes		Активизирует использование директив, контролирующих индексирование каталогов (AddDescription, AddIcon, AddIconByEncoding, AddIconByType, DefaultIcon, DirectoryIndex, FancyIndexing, HeaderName, IndexIgnore, IndexOptions, ReadmeName).
Limit		Активизирует использование директив, контролирующих доступ к узлу (allow, deny и order).
Options		Активизирует использование директив, контролирующих возможности каталогов (Options, XBitHack).

A.2.5. Директива AuthName

Синтаксис: AuthName *область санкций*
Контекст: каталог, .htaccess
Перекрытие: AuthConfig

Эта директива предназначена для установки имен области санкций для каталога. Область санкций предоставляется клиенту таким образом, что пользователи будут знать, какие вводить имя пользователя и пароль. Директива всегда используется совместно с директивами AuthConfig, require и комбинацией директив AuthUserFile и AuthGroupFile.

Пример

Для спецификации области санкций business_a необходимо задать следующую директиву:

```
AuthName business_a
```

A.2.6. Директива AuthType

Синтаксис: AuthType *тип*
Контекст: каталог, .htaccess
Перекрытие: AuthConfmg

Эта директива устанавливает тип идентификации пользователя для каталога. В настоящее время единственным возможным значением параметра *тип* является значение Basic. Директива используется совместно с директивами AuthName, require и комбинацией директив AuthUserFile и AuthGroupFile.

Чтобы задать тип идентификации Basic, воспользуйтесь следующей директивой:

```
AuthType Basic
```

A.2.7. Директива BindAddress

Синтаксис: BindAddress *saddr*
Умолчание: BindAddress *
Контекст: конфигурирование сервера

Unix-сервер может быть сконфигурирован для прослушивания соединений по всем IP-адресам сервера сразу или только по одному определенному IP-адресу. Возможные значения параметра saddr перечислены в табл. А.2.

Таблица А.2. Значения параметра saddr

Параметр	*Значение*
	Прослушивать соединения по всем IP-адресам.
<IP-адрес>	Прослушивать соединения только по указанному IP-адресу.
<полное имя домена>	Прослушивать соединения только по указанному домену.

Примечание

С помощью этой директивы реализуется один из методов виртуального хостинга. Детально виртуальный хостинг описан в главе 5, "Хостинг нескольких Web-узлов".

A.2.8. Директива ClearModuleList

Синтаксис: ClearModuleList
Контекст: конфигурирование сервера
Совместимость: директива ClearModuleList присутствует, начиная с версии сервера Apache 1.2

Сервер имеет предопределенный список активных модулей. Для того, чтобы очистить этот список, и предназначена директива ClearM oduleList. После использования этой директивы необходимо с помощью директивы AddModule составить альтернативный список модулей.

Для очистки встроенного списка активных модулей, задайте следующую директиву:

```
ClearModuleList
```

A.2.9. Директива DefaultType

Синтаксис: DefaultType *mime-mun*
Умолчание: DefaultType text/html
Контекст: конфигурирование сервера, виртуальный узел, каталоги, файл .htaccess
Перекрытие: FileInfo

Сервер Apache должен иметь возможность сообщать клиентам MIME-тип документов, которые он рассылает даже тогда, когда тип документа из его расширения не ясен. Эта ди-

ректива позволяет пользователю задавать тип документа по умолчанию, который будет использоваться тогда, когда сервер не сможет найти более подходящее решение.

Чтобы задать в качестве типа по умолчанию тип txt/html, воспользуемся директивой:

```
DefaultType text/html
```

A.2.10. Директива <Directory>

Синтаксис: <Directory *каталогу* . . . </Directory>
Контекст: конфигурирование сервера, виртуальный узел

Директивы, которые применяются для объединения других директив в группы, которые будут применяться только для определенного каталога и их подкаталогов. Обратите внимание на то, что для обозначения любого одиночного символа или группы символов, в этой директиве можно использовать групповые символы "*" и "?". Более подробно об ограничении области действия директив можно узнать в главе 1, "Основные концепции". Спецификации каталогов не могут быть вложенными. Даже в случае, когда множественные разделы каталогов совпадают с каталогом или родительским каталогом документа, директивы применяются, начиная с самого короткого соответствия. Следует напомнить также, что это имеет отношение и к директивам, находящимся в файле .htaccess.

Пример

Чтобы область действия директивы DirectiveA (совершенно абстрактной) распространялась на каталог /home/site2 и его подкаталоги, воспользуйтесь следующей конструкцией:

```
<Directory  /home/site2>
      DirectiveA
<Directory>
```

A.2.11. Директива DocumentRoot

Синтаксис: DocumentRoot *каталог-имя-файла*
Умолчание: DocumentRoot /usr/local/etc/httpd/htdocs
Контекст: конфигурирование сервера, виртуальный узел

Эта директива позволяет задавать корневой каталог, отличный от стандартного. Например, если в качестве DocumentRoot объявлен каталог /usr/businessa, это будет Значить, что URL *http://www.businessa.com/index.html* относится к документу /usr/businessa/index.html.

Каталог, определенный с помощью этой директивы является ключевым, так как многие другие директивы позволяют задавать путь относительно каталога DocumentRoot.

Примечание

При объявлении пути DocumentRoot указание замыкающей косой черты является ошибкой (например/somedir/).

Чтобы объявить в качестве корневого каталога вашего Web-узла каталог /usr/businessa, можно воспользоваться следующей директивой.

```
Documentroot /usr/businessa
```

A.2.12. Директива ErrorDocument

Синтаксис: ErrorDocument *код-ошибки документ*
Контекст: конфигурирование сервера, виртуальный узел, каталоги, файл .htaccess
Перекрытие: FileInfo

По умолчанию сервер Apache обрабатывает ошибки, выдавая запрограммированные сообщения о них. Эта директива позволяет изменить такое поведение одним из следующих способов:

Для вывода видоизмененного сообщения необходимо задать директиву:

```
ErrorDocument 403 "Having problems. Come back later"
```

Пример

Для перенаправления на локальный URL:

```
ErrorDocument 401 /customer.html
ErrorDocument 404 /cgi-bin/ customer.pl
```

Для перенаправления на внешний URL:

```
ErrorDocument 500 http://www.blah.com/cgi-bin/boo-boo
```

Примечание

При использовании директивы ErrorDocument 401 она должна ссылаться на локальный документ.

A.2.13. Директива ErrorLog

Синтаксис: ErrorLog *имя-файла*
Умолчание: ErrorLog logs/error_log
Контекст: конфигурирование сервера, виртуальный узел

Эта директива позволяет задать имя регистрационного файла, в который будут записываться диагностические сообщения об ошибках сервера. В нем может быть указан абсолютный путь (если спецификация пути начинается с символа "/", что означает путь определения относительно корневого каталога) или относительный путь относительно каталога ServerRoot.

Задать в качестве каталога регистрации ошибок каталог /var/adm/logs/http.log можно с помощью команды:

```
ErrorLog  /var/adm/logs/http.log
```

A.2.14. Директива <Files>

Синтаксис: <Files *имя-файла*>. . . </Files>
Контекст: конфигурирование сервера, виртуальный узел, файл .htaccess
Совместимость: версия 1.2 и выше

Эта директива позволяет осуществлять управление доступом по имени файла. Директивы, заключенные между директивами <Files>, будут иметь воздействие только на указанный файл. Есть директивы аналогичного характера и для каталогов URL. Секции <Files> обрабатываются в порядке их появления в конфигурационном файле после считывания секций <Directory> и файлов .htaccess, но до секций <Location>.

Имя файла может задаваться точным литеральным именем файла (например text/.html) или комбинацией литеральных и групповых символов (например text?/html, te*.html).

Чтобы директива DirectiveA воздействовала на все файлы, начинающиеся с последовательности символов srh, обратитесь к помощи следующей директивы:

```
<Files srh*>
      DirectiveA
<Files>
```

A.2.15. Директива Group

Синтаксис: Group *Unix-группа*
Умолчание: Group #-1
Контекст: конфигурирование сервера, виртуальный узел

Эта директива позволяет задать системную группу Unix (как видно из файла /etc/group), которой будет пользоваться сервер при обслуживании запросов.

Чтобы создать группу httpg, можно воспользоваться следующей директивой:

```
group httpg
```

Примечание

Из соображений безопасности рекомендуется создать группу специально для обслуживания запросов : сервером Apache.

A.2.16. Директива HostNameLookups

Синтаксис: HostNameLookups *on* | *off*
Умолчание: HostNameLookups on
Контекст: конфигурирование сервера, виртуальный узел

Эта директива необходима для включения режима просмотра базы DNS. В этом случае в регистрационных файлах вместо IP-адресов можно использовать имена узлов. Необходимо заметить, что процесс просмотра DNS-информации приводит к существенному снижению производительности сервера.

Чтобы немного "перекрыть кислород" вашему серверу, достаточно воспользоваться следующей директивой:

```
HostNameLookups    on
```

A.2.17. Директива IdentityCheck

Синтаксис: IdentityCheck *on \ off*
Умолчание: IdentityCheck off
Контекст: конфигурирование сервера, виртуальный узел

Эта директива позволяет регистрировать имена пользователей в соответствии со стандартом RFC 1413 при каждом подключении. Заметим, что для того, чтобы это работало эффективно, на клиентской машине должен работать сервис identd или что-то подобное с аналогичными функциями. Значение параметра boolean должно быть установлено в on или off.

Для включения режима идентификации пользователя достаточно:

```
IdentityCheck on
```

A.2.18. Директива <IfModule>

Синтаксис: <Ifmodule [!]*имя-модуля*> ... </IfModule>
Умолчание: нет
Контекст: все

Эта директива позволяет пользователю задавать или отменять выполнение директив в зависимости от наличия или отсутствия модуля. Директивы, находящиеся внутри операторной скобки <Ifmodule *имя-модуля*>, рассматриваются только тогда, когда модуль *имя-модуля* прикомпилирован к серверу, если модуль с указанным именем не прикомпилирован, директивы будут проигнорированы. Указание перед именем модуля знака восклицания инвертирует действие директивы. (Например указание *!modulea* означает, что директивы будут просматриваться только в том случае, когда модуль *modulea* отсутствует.)

A.2.19. Директива KeepAlive

Синтаксис: (Apache 1.1) KeepAlive *макс-запрос*
Умолчание: (Apache 1.1) KeepAlive 5
Синтаксис: (Apache 1.2) KeepAlive on/off
Умолчание: умолчание: (Apache 1.2) KeepAlive On
Контекст: конфигурирование сервера

В сервере Apache 1.1, эта директива позволяет задать верхний предел числа запросов KeepAlive на одного клиента. Для сервера Apache версии 1.2 и выше значение может быть также установлено в On, что позволяет делать устойчивые соединения, или Off, что отключает такую возможность. См. директиву MaxKeepAliveRequests.

Чтобы установить предел устойчивых соединений в значении 10 KeepAlive, можно воспользоваться следующей директивой:

```
KeepAlive 10
```

Чтобы отключить устойчивые соединения (в целом это очень нездоровая идея), воспользуйтесь следующей директивой:

```
KeepAlive off
```

A.2.20. Директива KeepAliveTimeout

Синтаксис: KeepAliveTimeout *секунды*
Умолчание: KeepAliveTimeout 15
Контекст: конфигурирование сервера

Эта директива позволяет задать в секундах время ожидания сервером Apache следующего запроса перед тем, как прервать соединение.

Для задания тайм-аута, равного 10 секундам, перед отключением неактивного соединения задайте директиву:

```
KeepAliveTimeout 10
```

A.2.21. Директива Listen

Синтаксис: Listen *[IP-адрес:]номер порта*
Контекст: конфигурирование сервера

Эта директива задает режим прослушивания более одного IP-адреса или порта. По умолчанию сервер прослушивает соединения по всем IP-адресам, но только по тем портам, которые указаны директивой Port. Директива Listen обычно используется в системах со многими IP-подключениям для наблюдения только за определенными сетевыми картами.

Пример

Для включения режима наблюдения запросов, поступающих на порт 443, воспользуйтесь следующей директивой:

```
Listen 192.168.100.1:443
```

A.2.22. Директива <Limit>

Синтаксис: <Limit ***метод метод*** ...> </Limit>
Контекст: **любой**

Синтаксис директивы Limit позволяет задать перечень директив управления доступом, которые будут действовать только на перечисленные методы доступа HTTP.

Для идентификации пользователей (см. главу 8, "Безопасность"), которые могут работать только с запросами DELETE, воспользуйтесь следующей директивой:

```
<Limit  DELETE>
       require  valid-user
</Limit>
```

A.2.23. Директива <Location>

Синтаксис: <Location URL> ... </Location>
Контекст: конфигурирование сервера, виртуальный узел

Эта директива позволяет управлять доступом к перечисленным URL. Действие этих директив ограничивается только URL, перечисленными между директивами <Location URL> И </Location>.

Чтобы директива DirectiveA (абстрактная) имела отношение только к локальному узлу /server-info, задайте следующую директиву:

```
<Location  /server-info>
       DirectiveA
<Location>
```

A.2.24. Директива LockFile

Синтаксис: LockFile *имя-файла*
Умолчание: LockFile logs/accept.lock
Контекст: конфигурирование сервера

Эта директива предназначена для спецификации пути к файлу LockFile, который используется при компиляции сервера Apache с ключами USE_FCNTL_SERIALIZED_ACCEPT или USE_FLOCK_SERIALIZED_ACCEPT. При этом место размещения файла LockFile не может быть смонтированным каталогом сетевой файловой системы (NFS).

Чтобы файл LockFile был размещен в каталоге /var/httpd/lock, воспользуйтесь следующей директивой:

```
LockFile  /var/httpd/lock
```

A.2.25. Директива MaxClients

Синтаксис: MaxClients *число*
Умолчание: MaxClients 256
Контекст: конфигурирование сервера

Эта директива позволяет задать верхний предел количества порожденных процессов сервера, эффективно ограничивая количество одновременно обрабатываемых запросов.

Для ограничения количества одновременных соединений цифрой 100 воспользуйтесь следующей директивой:

```
MaxClients  100
```

A.2.26. Директива MaxKeepAliveRequests

Синтаксис: MaxKeepAliveRequests *число*
Умолчание: MaxKeepAliveRequests 100
Контекст: конфигурирование сервера

Эта директива устанавливает верхний предел количества запросов на одно соединение, когда директива KeepAlive установлена в on. При установке значения, отличного от 0 (неограниченно), значение должно быть установлено как можно более высоким, чтобы увеличить производительность сервера.

Чтобы ограничить количество запросов на соединение числом запросов 200, достаточно воспользоваться директивой:

```
MaxKeepAliveRequests 200
```

A.2.27. Директива MaxRequestsPerChild

Синтаксис: MaxRequestsPerChild *число*
Умолчание: MaxRequestsPerChild 0
Контекст: конфигурирование сервера

Эта директива позволяет задавать верхний предел количества запросов, которые сможет обработать определенный порожденный процесс сервера. При достижении этого числа запросов порожденный процесс будет остановлен. При установке значения в 0 такой предел не устанавливается. Есть две причины для применения этой директивы:

- Она позволяет ограничить объем вероятного повреждения из-за потерь памяти.
- Она уменьшает количество простаивающих процессов, оставшихся после пиковых нагрузок.

Пример

Чтобы порожденные процессы завершались после обработки 50 запросов пользователей, можно воспользоваться следующей директивой:

```
MaxRequestsPerChild 50
```

A.2.28. Директива MaxSpareServers

Синтаксис: MaxSpareServers *число*
Умолчание: MaxSpareServers 0
Контекст: конфигурирование сервера

Эта директива устанавливает верхний предел простаивающих порожденных процессов на сервере. По достижению указанного числа сервер будет уничтожать любой новый процесс.

Чтобы сервер Apache завершил работу простаивающих серверов, когда их становится больше 20, воспользуйтесь следующей директивой:

```
MaxSpareServers 20
```

A.2.29. Директива MinSpareServers

Синтаксис: MinSpareServers *число*
Умолчание: MinSpareServers 5
Контекст: конфигурирование сервера

Эта директива задает нижний предел простаивающих порожденных процессов. Когда количество таких процессов становится ниже предельного числа, сервер Apache начинает создавать новые процессы со скоростью одного процесса в секунду.

Чтобы сервер Apache начал порождать процессы, когда их становится меньше 10, воспользуйтесь следующей директивой:

```
MinSpareServers 10
```

A.2.30. Директива Options

Синтаксис: Options *[+\-]опция [+\-]опция* . . .
Контекст: конфигурирование сервера, виртуальный узел, каталоги, файл .htaccess
Перекрытие: Options

Эта директива позволяет управлять возможностями, присущими определенному каталогу. Директива Options принимает значение "попе" или одно из значений, приведенных в табл. А.3.

Таблица А.3. Значения директивы options

Опции	*Назначение*
All	Активизировать все опции за исключением опции MultiViews.
ExecCGI	Активизировать выполнение CGI-сценариев.
FollowSymLinks	Сервер просматривает символические связи в этом каталоге. Рекомендуется в случае, даже если сервер просматривает символические ссылки, он не изменяет пути, используемого для поиска соответствия в разделах <Directory>.
includes	Активизировать использование вставок на стороне сервера.
IncludesNOEXEC	Вставки на стороне сервера разрешены, но команды #exec и #include в CGI-сценариях отключены.
Indexes	При этой опции, если URL ссылается на каталог, где отсутствует файл DirectoryIndex (чаще всего это файл index.html), сервер возвращает отформатированную распечатку содержимого этого каталога.
Multiviews	Разрешить вывод содержимого в формате MultiViews.
SymLinksIfOwnerMatch	Просматривать символические связи только в случае, когда владелец файла или каталога совпадает с владельцем ссылки.

В случае применения нескольких опций к определенному каталогу будет действовать наиболее специфическая опция. Действия опций не суммируются. Однако, если опции предшествует символ "+" или "-", действие опций программируется следующим образом: те опции, которым предшествует символ "+", прибавляются к списку уже действующих опций, а те, которым предшествует символ "-", удаляются из активного в данный момент списка.

Чтобы добавить возможность выполнения CGI-программ к списку уже действующих для каталога , /home/sample, необходимо добавить директиву:

```
<Directory /home/sample>
    Options +ExecCGI
</Directory>
```

Пример

Для создания нового списка опций, устанавливающих возможность выполнения CGI-программ только для каталога /home/sample, воспользуйтесь следующей директивой:

```
<Directory /home/sample>
    Options ExecCGI
</Directory>
```

A.2.31. Директива PidFile

Синтаксис: PidFile *имя-файла*
Умолчание: PidFile logs/httpd.pld
Контекст: конфигурирование сервера

Эта директива позволяет задать имя файла, в котором будет записываться идентификатор текущего процесса. Если имя файла не начинается символом "/", то он считается не абсолютным путем, а относительным относительно каталога ServerRoot.

Чтобы сервер Apache разместил PID-файл в файле /some/secure/location, воспользуйтесь следующей директивой:

```
PidFile /some/secure/location
```

A.2.32. Директива Port

Синтаксис: Port *номер*
Умолчание: Port 80
Контекст: конфигурирование сервера

Стандартным портом, посредством которого осуществляются все HTTP-соединения, является порт с номером 80. Если по каким-либо причинам вы и ваш клиент договорились работать с другим портом, его необходимо с помощью этой директивы объявить. Максимально допустимое значение составляет 65535. Многие порты (в частности, порты, имеющие номер меньше, чем 1024) уже зарезервированы для других протоколов.

Чтобы сервер Apache начал прослушивать порт 443(такой порт необходим для протокола SSL), можно воспользоваться следующей директивой:

```
Port 443
```

A.2.33. Директива require

Синтаксис: require *имя-сущности имя-сущности . . .*
Контекст: каталог, файл .htaccess
Перекрытие: AuthConfig

Эта директива позволяет предоставлять пользователю или группе пользователей доступ к определенному каталогу. *Имя-сущности* является именем пользователя, группы или зарегистрировавшимся пользователем. В последнем случае право доступа к каталогу будет иметь любой зарегистрировавшийся пользователь. В противном случае доступ ограничен только определенными пользователями или группами пользователей.

Чтобы ограничить доступ только тремя пользователями bob, timmy и susie, можно воспользоваться следующей директивой:

```
require user bob timmy susie
```

A.2.34. Директива ResourceConfig

Синтаксис: ResourceConfig *имя-файла*
Умолчание: ResourceConfig conf/srm.conf
Контекст: конфигурирование сервера, виртуальный узел

Эта директива определяет файл, содержащий дополнительные директивы, которые считываются сервером после файла httpd.conf. Имя файла почти всегда задается относительно каталога ServerRoot. Чтобы отключить эту возможность, достаточно вместо имени файла указать устройство /dev/null.

Чтобы задать размещение файла ResourceConfig в каталоге some/secure/directory, воспользуйтесь директивой:

```
ResourceConfig some/secure/directory
```

A.2.35. Директива RLimitCPU

Синтаксис: RLimitCPU *номер или 'max' [номер или 'max']*
Умолчание: отмена стандартных установок операционной системы
Контекст: конфигурирование сервера, виртуальный узел

Первый (необходимый) параметр устанавливает мягкий предел для всех процессов. Второй (может быть пропущен) параметр устанавливает максимальный предел для всех процессов.

Эти параметры могут быть представлены числом или ключевым словом "max", которое устанавливает максимальное значение для операционной системы. Ограничения для центрального процессора задаются в долях секунд на процесс.

Чтобы задать верхний предел равным 20 секу ндам на процесс, воспользуемся директивой:

```
RLimitCPU 20
```

A.2.36. Директива RLimitMEM

Синтаксис: RLimitMEM *номер или 'max' [номер или 'max']*
Контекст: конфигурирование сервера, виртуальный узел

Эта директива принимает один из двух параметров. Первый параметр используется для установки мягкого ограничения ресурсов для всех процессов, второй — для установки максимального ограничения ресурсов. Эти параметры могут быть числами, показывающими количество байтов, отводящихся на каждый процесс, или словом "max", которое будет обозначать максимально возможное значение для операционной системы. Значения задаются в байтах.

Пример

Чтобы задать верхний предел в 20 Мбайт на процесс, воспользуемся командой:

```
RLimitMEM 20000000
```

A.2.37. Директива RLimitNPROC

Синтаксис: RLimitNPROC *номер или 'max' [номер или 'max']*
Умолчание: отмена стандартных установок операционной системы
Контекст: конфигурирование сервера, виртуальный узел

Эта директива используется для ограничения количества порожденных процессов (например CGI-сценариев), которые порождаются процессом сервера Apache. По проекту эта директива никак не влияет на количество работающих процессов сервера Apache. Однако при этом предполагается, что порожденные процессы работают под другим идентификатором пользователя.

Чтобы ограничить количество порождаемых процессов количеством 100, воспользуйтесь следующей директивой:

```
RLimitNPROC100
```

A.2.38. Директива Satisfy

Синтаксис: Satisfy *'any' или 'all'*
Умолчание: Satisfy all
Контекст: каталог, файл .htaccess

Эта директива необходима для согласования требований по санкционированию доступа как с точки зрения идентификации "клиент/узел", так и с точки зрения идентификации "имя-пользователя/пароль". Альтернатива 'any' обеспечивает пользователю доступ к ресурсу в случае удовлетворения одному (или сразу двум) ограничениям по доступу.

Чтобы потенциальные пользователи каталога /home/sample удовлетворяли как ограничениям доступа "имя-пользователя/пароль", так и ограничениям доступа "клиент/узел", можно воспользоваться следующей директивой:

```
</Directory  /home/sample>
      Satisfy  all
</Directory>
```

A.2.39. Директива ScoreBoardFile

Синтаксис: ScoreBoardFile *имя-файла*
Умолчание: ScoreBoardFile logs/apache_status
Контекст: конфигурирование сервера

Эта директива необходима в некоторых архитектурах для определения размещения файла, который используется сервером для обмена данными между процессами-родителями и порожденными процессами.

Чтобы задать в качестве каталога размещения ScoreBoardFile каталога /etc/httpd/logs/sbf, воспользуйтесь следующей директивой:

```
ScoreBoardFile  /etc/httpd/logs/sbf
```

A.2.40. Директива SendBufferSize

Синтаксис: SendBuffersize *байт*
Контекст: конфигурирование сервера, виртуальный узел

Параметр *байт* предназначен для определения размера буфера TCP в байтах. Эта директива используется для модификации старых установок операционной системы при установке более быстрой реакции сервера.

Для установки размера буфера TCP примерно равного 2 Мбайтам примените следующую директиву:

```
SendBufferSize      2000000
```

A.2.41. Директива ServerAdmin

Синтаксис: ServerAdmin *почтовый адрес*
Контекст: конфигурирование сервера, виртуальный узел

Эта директива предназначена для объявления почтового адреса, на который будут посылаться сообщения при возникновении ошибок в работе сервера.

Для указания почтового адреса администратора *admin@yourste.org,* воспользуйтесь следующей директивой:

```
ServerAdmin   admin@yoursite.org
```

A.2.42. Директива ServerAlias

Синтаксис: ServerAlias *узел1 узел2 ...*
Контекст: виртуальный узел

Эта директива может использоваться для задания альтернативных имен узла при виртуальном хостинге на основании Host-заголовка.

Для определения псевдонима site2, воспользуйтесь следующей директивой:

```
ServerAlias  site2
```

A.2.43. Директива ServerName

Синтаксис: ServerName *полное имя домена*
Контекст: конфигурирование сервера, виртуальный узел

Эту директиву можно использовать для указания имени узла сервера при назначении URL для пересылки. Если эта директива не используется, сервер будет использовать имя сервера, связанное с его IP-адресом.

Чтобы присвоить серверу имя www.example.com, воспользуйтесь следующей директивой:

```
<VirtualHost 192.168.100.1>
      ServerName  www.example.com
<VirtualHost>
```

A.2.44. Директива ServerPath

Синтаксис: ServerPath *полный-путь*
Контекст: виртуальный узел

При создании виртуального узла на основании Host-заголовка воспользуйтесь следующей директивой для установки имени пути URL к узлу.

Чтобы установить путь /user/oldsite, воспользуйтесь следующей директивой:

```
ServerPath /user/oldsite
```

A.2.45. Директива ServerRoot

Синтаксис: ServerRoot *каталог-имя_файла*
Умолчание: ServerRoot /usr/local/etc/httpd

Это корень, относительно которого задается путь ко всем конфигурационным файлам. Он очень важен тогда, когда этот путь является относительным путем для всех файлов, которыми пользуется сервер. Аналогичного эффекта можно также добиться с помощью опции -d в командной строке при запуске.

•Чтобы корневым каталогом сервера Apache стал каталог /usr/apache, воспользуйтесь следующей директивой:

```
ServerRoot  /usr/apache
```

A.2.46. Директива ServerType

Синтаксис: ServerType *тип*
Умолчание: ServerType standalone
Контекст: конфигурирование сервера, виртуальный узел

Эта директива предназначена для определения порядка запуска сервера системой. Может задаваться *тип* inetd или standalone. При значении inetd процесс запускается процессом inetd в соответствии с данными из конфигурационного файла inetd.conf. Процесс запускается как standalone, когда он запускается одним из сценариев загрузки.

Следует напомнить, что при использовании метода inetd при каждом запросе соединения HTTP создается новый экземпляр процесса сервера. Этот метод не очень эффективен, но значительно более безопасен.

При средней загруженности сервера, вариант standalone является единственным приемлемым вариантом.

Для запуска сервера Apache в режиме standalone, воспользуйтесь следующей директивой:

```
ServerType standalone
```

A.2.47. Директива StartServers

Синтаксис: StartServers *число*
Умолчание: StartServers 5
Контекст: конфигурирование сервера, виртуальный узел

Эта директива необходима для задания количества порожденных процессов, создаваемых в момент запуска сервера. Напомним, что количество порожденных процессов, работающих в конкретный момент времени, управляется динамически (см. директивы MaxSpareServers, MinSpareServers и т.д.), таким образом, этот параметр действует только непосредственно после запуска.

Для запуска сервера Apache с уже работающими 20 серверами:

```
StartServers 20
```

A.2.48. Директива TimeOut

Синтаксис: TimeOut *число*
Умолчание: TimeOut 300
Контекст: конфигурирование сервера, виртуальный узел

Сервер Apache будет выдерживать тайм-аут при:

1. получении запроса GET,
2. ожидании следующего пакета после запроса POST или Put,
3. ожидании подтверждения при посылке пакетов TCP.

Чтобы установить значение тайм-аута равным 3 минутам, воспользуйтесь следующей директивой:

```
TimeOut 180
```

A.2.49. Директива User

Синтаксис: User *пользователь Unix*
Умолчание: User #-1
Контекст: конфигурирование сервера, виртуальный узел

Эта директива предназначена для определения идентификатора пользователя userid, по которому сервер будет отвечать на запросы. Идентификатор пользователя задается именем пользователя или символом "#", предваряющим номер пользователя. Оба эти значения хранятся в файле /etc/passwd. Из соображений безопасности необходимо создать абсолютно нового пользователя, под которым должен работать сервер Apache.

Пример

Чтобы процесс httpd работал под управлением пользователя apache, воспользуйтесь следующей директивой:

```
User apache
```

A.2.50. Директива VirtualHost

Синтаксис: <VirtualHost *адрес[:порт]* ...> ... </VirtualHost>
Контекст: конфигурирование сервера, виртуальный узел

Эти парные директивы можно использовать для выделения группы директив, которые относятся только к указанному виртуальному узлу. Заданный адрес может быть реальным IP-адресом или полностью определенным именем домена. Специальное имя _default_ используется для определения директив, которые будут использованы при отсутствии каких-либо других спецификаций.

Примечание

Виртуальный хостинг — это достаточно сложный процесс. С деталями этого процесса можно познакомиться в главе 5, "Хостинг нескольких Web-узлов".

Для прослушивания запросов, поступающих к узлу по имени www.site2.com IP-адреса 1192.168.100.1, воспользуйтесь следующей директивой:

```
NameVirtualHost 192.168.100.1
<VirtualHost 192.168.100.1>
      ServerName  www.site2.com
</VirtualHost>
```

Пример

Для прослушивания IP-адреса 192.168.100.20 на предмет поступления запросов на виртуальный узел www.site2.com воспользуйтесь следующей директивой:

```
<VirtualHost  192.168.100.20>
      ServerName  www.site2.com
</VirtualHost>
```

Приложение

Б

ПРОЧИЕ ДИРЕКТИВЫ

в этом *приложении...*

Б. 1. Модуль mod_access

Этот модуль находится в файле mod_access.c, который компилируется по умолчанию. Он осуществляет контроль доступа на основании имени узла клиента или его IP-адреса.

Б. 1.1. Директива allow

Синтаксис: allow from *узел узел* . . .
Контекст: каталог, файл .htaccess
Перекрытие: Limit
Статус: Base
Модуль: mod_access

Эта директива предназначена для объявления узлов, которым разрешен доступ к определенной директиве. Допустимые значения параметра *узел* перечислены в табл. **Б.1.**

Таблица Б.1. Значения параметра директивы allow

Значение параметра	*Действие параметра*
all	Доступ разрешен для всех узлов.
имя домена	Имя узла, которому разрешен доступ.
полный ip-адрес	IP-адрес, которому разрешен доступ.
частичный ip-адрес	Первые от 1 до 3 байтов IP-адреса для определения подсети, которой разрешен доступ.

Чтобы разрешить доступ для всех узлов в заданном домене, воспользуемся директивой:

```
allow from .ncsa.uiuc.edu
```

Примечание

Требуется указание всех элементов имени: blah. edu не совпадает с blahblah.edu.

Б. 1.2. Директива allow from env

Синтаксис: allow from *env=имя-переменной*
Контекст: каталог, файл .htaccess
Перекрытие: Limit
Статус: Base
Модуль: mod_access
Совместимость: сервер Apache версии 1.2 и выше

Директива allow from env управляет доступом к каталогу по факту существования или отсутствия переменной окружения. Это очень удобно при настройке Web-страницы, заданной в соответствии с особенностями броузеров.

Пример

Пусть переменная окружения **brwsrl** установлена директивой **BrowserMatch** до вызова оператора **allow from env**. Следующий пример позволит пользователю **brwsrl** получить доступ к каталогу `for_brwsrl`.

```
BrowserMatch ~somebrowser/2.O brwsrl
      <Directory /for_brwsrl>
      order allow, deny
      allow from env=brwsrl
      deny from all
</Directory>
```

Б.1.3. Директива deny

Синтаксис: deny *узел узел . . .*
Контекст: каталог, файл .htaccess
Перекрытие: Limit
Статус: Base
Модуль: mod_access

Эта директива позволяет ограничить доступ определенным узлам к определенным каталогам. Параметр *узел* может принимать одно из значений, показанных в табл. Б.2.

Таблица Б.2. Значения параметра директивы **deny**

Значение параметра	*Действие параметра*
all	Доступ запрещен для всех узлов.
имя домена	Имя узла, которому запрещен доступ.
полный ip-адрес	IP-адрес, которому запрещен доступ.
частичный ip-адрес	Первые от 1 до 3 байтов IP-адреса для определения подсети, которой запрещен доступ.

Б.1.4. Директива deny from env

Синтаксис: deny from env=*имя-переменной*
Контекст: каталог, файл .htaccess
Перекрытие: Limit
Статус: Base
Модуль: mod_access
Совместимость: сервер Apache версии 1.2 и выше

Пример

Эта директива предназначена для того, чтобы запретить доступ к определенному каталогу на основании существования определенной переменной окружения. См. также директиву **allow from env**.

```
BrowserMatch ^somebrowser/O.9 brwsr2
      <Directory /for_brwsr1>
      order allow, deny
      deny from env=brwsr2
      allow from all
</Directory>
```

Б.1.5. Директива order

Синтаксис: order *порядок-сортировки*
Умолчание: order deny, allow
Контекст: каталог, файл .htaccess
Перекрытие: Limit
Статус: Base
Модуль: mod_access

Директива order предназначена для того, чтобы сообщать серверу Apache о приоритетности запрета и разрешения. Возможные значения, которые может принимать параметр директивы, и их действия перечислены в табл. Б.3.

Таблица Б.3. Значение и действие параметров директивы order

Значение параметра	*Действие параметра*
deny, allow	Сначала оцениваются директивы deny, потом директивы allow. Начальное состояние OK.
allow, deny	Сначала оцениваются директивы allow, затем директивы deny. Начальное состояние FORBIDDEN.
mutual-failure	Чтобы иметь гарантированный доступ, узел должен присутствовать в списке разрешенных узлов и отсутствовать в любом из запрещающих списков.

Пример

Чтобы задать самый высокий из возможных уровней ограничения доступа, необходимо воспользоваться директивой:

```
order mutual-failure
```

Б.2. Модуль mod_actions

Этот модуль позволяет запускать CGI-сценарии, когда пользователь получает доступ к файлу определенного типа, запуская тем самым выполнение сценариев, производящих обработку файлов. Этот модуль имеется только в сервере Apache версии 1.1 или выше.

Б.2.1. Директива Action

Синтаксис: Action *cgi-сценарий mime-тип*
Контекст: конфигурация сервера, виртуальный узел, каталог, файл .htaccess
Перекрытие: FileInfo
Статус: Base
Модуль: mod_actions
Совместимость: директива Action имеется только в сервере Apache версии 1.1 и выше

Эта директива вызывает определенный CGI-сценарий, когда поступает запрос к определенному MIME-типу. Он высылает URL и путь к файлу запрошенного документа с помощью стандартных переменных окружения CGI PATH_INFO и PATH_TRANSLATED.

Предположим, что у вас имеется MIME-тип mt1, который должен быть обработан сценарием hndl_mt1. Чтобы связать сценарий hndl_mt1.pl с файлом MIME-типа mt1, необходимо сделать следующее:

```
Action mtl hndl_mt1.pl
```

Б.2.2. Директива Script

Синтаксис: Script *метод CGI-сценарий*
Контекст: конфигурация сервера, виртуальный узел, каталог
Статус: Base
Модуль: mod_actions
Совместимость: директива Script имеется только в сервере Apache версии 1.1 и выше

Эта директива добавляет операцию, которая активизирует сценарий *CGI-сценарий,* когда файл запрашивается с помощью метода *метод.* Это может быть метод GET, POST, PUT или DELETE. С помощью стандартных переменных окружения PATH_INFO и PATH_TRANSLATED он передает URL и путь к файлу, в котором хранится запрошенный документ. Необходимо напомнить, что директива Script определяет только действия по умолчанию. Если вызывается CGI-сценарий или какой-либо другой ресурс, который может обрабатывать запрошенный метод самостоятельно, он это сделает. Заметим также, что сценарий с методом GET вызывается только при наличии аргументов запроса (например, foo.html?hi). В противном случае запрос будет обработан в обычном порядке.

Пример

Чтобы связать метод PUT со сценарием script /cgi-bin/put.pl, воспользуйтесь следующей директивой:

```
</DIV>
script PUT /cgi-bin/put.pl
```

Б.3. Модуль mod alias

Директивы из этого модуля позволяют осуществлять доступ к частям файловой системы узла с помощью заданных псевдонимов.

Б.3.1. Директива Alias

Синтаксис: Alias *url-путь каталог_имя-файла.*
Контекст: конфигурация сервера, виртуальный узел
Статус: Base
Модуль: mod_alias

Эта директива позволяет хранить документы, которые находятся вне каталога DocumentRoot.

Чтобы URL, заканчивающийся символической последовательностью cgi-bin, ассоциировался с каталогом /var/secure/cgi-bin, который предположительно расположен в защищенном месте файловой системы, воспользуемся директивой:

```
Alias cgi-bin /var/secure/cgi-bin
```

Б.3.2. Директива Redirect

Синтаксис: Redirect *[статус] url-путь url*
Контекст: конфигурация сервера, виртуальный узел, каталог, файл .htaccess
Статус: Base
Модуль: mod_alias
Совместимость: контексты каталога и файла .htaccess имеются только на сервере Apache 1.1 и выше. Аргумент *статус* имеется только на сервере Apache 1.2 и выше

Директива redirect необходима для трансляции старого URL в новый. Новый URL автоматически возвращается клиенту, который затем повторяет попытку обратиться по новому адресу. Эта директива всегда предшествует директивам Alias и Script Alias. Необходимо также обратить внимание, что задается только абсолютный, а не относительный путь к DirectoryRoot. Действие аргумента *статус* (имеющегося в Apache 1.2 или выше) показано в табл. Б.4.

Таблица Б.4. Действие аргумента status

Значение аргумента	*Действие*
none	Временное переназначение (HTTP статус 302).
permanent	Постоянное переназначение (HTTP статус 301).
temporary	Временное переназначение (HTTP статус 302).
seeother	Возвращает переназначение "см. другое" ("see other" переназначение HTTP статус 303).
gone	Возвращает переназначение "прошло" ("gone" HTTP статус 410).

Примечание

Есть возможность задать статус, отличный от тех, что перечислены выше, но для этого необходимо, чтобы сервер предварительно узнал о нем с помощью функции send_error_response из модуля http_protocol.c.

Чтобы переназначить все запросы, содержащие путь /cgi-bin, на www.cgiserv.com/cgi-bin необходимо:

```
Redirect /cgi-bin www.cgiserv.com/cgi-bin
```

Б.3.3. Директива RedirectTemp

Синтаксис: RedirectTemp *url-путь url*
Контекст: конфигурация сервера, виртуальный узел, каталог, файл .htaccess
Статус: Base
Модуль: mod_alias
Совместимость: эта директива имеется только на сервере Apache **1.2**

Эта директива определяет (как нетрудно догадаться из названия) временное переназначение при HTTP статусе 302. Директива взаимозаменяема с парой Redirect temp directive/argument.

Для временного переназначения всех запросов, содержащих путь /cgi-bin на URL www.cgiserv.com/cgi-bin, необходимо:

```
RedirectTemp /cgi-bin www.cgiserv.com/cgi-bin
```

Б.3.4. Директива RedirectPermanent

Синтаксис: RedirectPermanent *url-путь url*
Контекст: конфигурация сервера, виртуальный узел, каталог, файл .htaccess
Статус: Base
Модуль: mod_alias
Совместимость: директива присутствует только в версии 1.2

Эта директива объявляет постоянную перемаршрутизацию при HTTP статусе 301. Директива взаимозаменяема с парой Redirect permanent directive/argument.

Пример

Для постоянного переназначения всех запросов, содержащих путь /cgi-bin на URL www.cgiserv.com/cgi-bin, необходимо:

```
RedirectPermanent /cgi-bin www.cgiserv.com/cgi-bin
```

Б.3.5. Директива ScriptAlias

Синтаксис: ScriptAlias *url-путь каталог_имя-файла*
Контекст: конфигурация сервера, виртуальный узел
Статус: Base
Модуль: mod_alias

Эта директива во многом аналогична директиве Alias за исключением того, что в ней объявляется целевой каталог, который содержит CGI-сценарии. URL, начинающиеся с указанного *url-путь,* будут связываться со сценариями, которые начинаются с *каталог_имя-файла.*

Для перемаршрутизации всех запросов, которые содержат в своем имени /cgi-bin/, на каталог /sbin/cgi-bin (который обозначен, как каталог с программами), воспользуйтесь следующей директивой:

```
ScriptAlias /cgi-bin/ /sbin/cgi-bin
```

Б.4. Модуль mod_auth

Директивы этого модуля связаны с процедурой идентификации пользователей с помощью текстовых файлов. Заметим, что эти директивы в действительности не могут обеспечить достаточного уровня безопасности. Для этого необходимо применение протокола SSL. Подробно проблема безопасности рассматривается в главе 8, "Безопасность".

Б.4.1. Директива AuthGroupFile

Синтаксис: AuthGroupFile *имя-файла*
Контекст: каталог, файл .htaccess
Перекрытие: AuthConfig
Статус: Base
Модуль: mod_alias

Эта директива предназначена для того, чтобы задать абсолютный путь к текстовому файлу, содержащему список пользовательских групп для идентификации пользователей. Этот файл имеет следующий формат: строка начинается с цифрового имени группы, затем следует двоеточие, список пользователей, разделенных пробелами. Просмотра больших файлов пользователей следует избегать, так как это влечет за собой перерасход ресурсов. Старайтесь пользоваться возможностями директивы AuthDBMGroupFile, а небольшими файлами групп. Для соблюдения правил безопасности AuthGroupFile должен храниться в месте, недоступном для пользователей.

Пример

Чтобы задать размещение файла authgroupfile в каталоге /etc/secure/groupfile, необходимо сделать следующее:

```
AuthGroupFile/etc/secure/groupfile
```

Б.4.2. Директива AuthUserFile

Синтаксис: AuthUserFile *имя-файла*
Контекст: каталог, файл .htaccess
Перекрытие: AuthConfig
Статус: Base
Модуль: mod_auth

Эта директива предназначена для определения абсолютного пути к текстовому файлу, содержащему перечень имен пользователей для идентификации пользователей. Файл пользователей имеет следующий формат: строка начинается с цифрового кода пользователя, затем следует двоеточие, затем пароль пользователя, закодированный с помощью функции crypt(). Для повышения эффективности при разрастании этого файла воспользуйтесь директивой AuthDBMUserFile.

Пример

Чтобы задать в качестве каталога расположения файла пользователей каталог /etc/secure/groupfile, воспользуйтесь директивой следующего вида:

```
AuthUserFile  /etc/secure/groupfile
```

5.4.3. Директива AuthAuthoritative

Синтаксис: AuthAuthoritative *<On (умолчание) | Off>*
Контекст: каталог, файл .htaccess
Перекрытие: AuthConfig
Статус: Base
Модуль: mod_auth

Значение, установленное директивой AuthAuthoritative, свидетельствует о том, можно или нет другим модулям идентификации производить идентификацию пользователей, которые не прошли идентификацию с помощью модуля mod_auth. По умол-

чанию устанавливается значение "Да" — то есть модуль mod_auth является последней инстанцией в процессе идентификации. Однако повсеместно принята практика, позволяющая производить идентификацию на основании одного из модулей идентификации mod_auth_db.c, mod_auth_dbm.c и т.д. Эти модули оперируют данными из специализированных баз данных.

Пусть имеется два модуля идентификации mod_auth.c и mod_auth_anon.c. Чтобы модуль mod_auth.c знал, что в случае отсутствия положительного результата при идентификации с помощью этого модуля идентификация должна производиться модулем mod_auth_anon.c, необходимо прибегнуть к директиве:

```
AuthAuthoritative Off
```

Б.5. Модуль mod_auth_anon

Директивы, находящиеся в этом модуле, позволяют осуществить анонимный доступ, аналогичный доступу anonymous ftp; т.е. существует пользователь с идентификатором "anonymous", пароль которого является адресом его электронной почты. При использовании этого метода в совокупности с модулями, управляющими доступом на основании хэшированных баз данных, можно отслеживать доступ пользователей, не перекрывая при этом доступ внешнему миру. Обсуждение этой проблемы можно найти в главе 8, "Безопасность".

Б.5.1. Директива Anonymous

Синтаксис: Anonymous *пользователь пользователь* . . .
Умолчание: нет
Контекст: каталог, файл .htaccess
Перекрытие: AuthConfig
Статус: Base
Модуль: mod_auth_anon

Эта директива предназначена для объявления одного или более пользователей, которым разрешен доступ без проверки пароля. По соглашению, в случае, когда открыт доступ широкому кругу пользователей, в списке пользователей должен присутствовать пользователь "anonymous".

Чтобы разрешить доступ псевдопользователям "anonymous" и "guest", воспользуйтесь командой:

```
Anonymous guest guest
```

Б.5.2. Директива Anonymous_Authoritative

Синтаксис: Anonymous_Authoritative ***on*** | ***off***
Умолчание Anonymous_Authoritative off
Контекст: **каталог, файл** .htaccess
Перекрытие: AuthConfig
Статус: Extension
Модуль: mod_auth_anon

При установке директивы Anonymous_Authoritative в on это будет единственным методом идентификации пользователя. Поэтому, если не получается анонимная

идентификация, доступ предоставлен не будет. Если директива установлена в off, доступ будет возможен с помощью различных методов.

Пример

Чтобы задать модуль mod_auth_anon в качестве последней и окончательной инстанции в управлении доступом, задайте следующую директиву:

```
Anonymous_Authoritative on
```

Б.5.3. Директива Anonymous_LogEmail

Синтаксис: Anonymous_LogEmail *on* | *off*
Умолчание off
Контекст: каталог, файл .htaccess
Перекрытие: AuthConfig
Статус: Extension
Модуль: mod_auth_anon

Когда эта директива установлена в on, при регистрации анонимного пользователя вводится псевдопароль, зарегистрированный в файле httpd-log. По соглашению этот псевдопароль имеет реальный адрес электронной почты.

Пример

Чтобы задать режим задания адреса электронной почты вместо пароля во время процесса анонимной регистрации, необходимо задать команду:

```
Anonymous_LogEmail on
```

Б.5.4. Директива Anonymous_MustGiveEmail

Синтаксис: Anonymous_MustGiveEmail *on|off*
Умолчание off
Контекст: каталог, файл .htaccess
Перекрытие: AuthConfig
Статус: Extension
Модуль: mod_auth_anon

При установке в on, вводимый псевдопароль не может быть пустым. Эта директива предназначена для того, чтобы предотвратить появление мусора.

Чтобы заставить пользователя во время анонимного доступа вводить хотя бы немного произвольных символов в качестве пароля, воспользуйтесь следующей командой:

```
Anonyraous_MustGiveEmail on
```

Б.5.5. Директива Anonymous_NoUserID

Синтаксис: Anonymous_NouserID *on* | *off*
Умолчание Anonymous_NouserID off
Контекст: каталог, файл .htaccess
Перекрытие: AuthConfig
Статус: Extension
Модуль: mod_auth_anon

Когда эта директива установлена в on, пользователи могут осуществлять доступ, вводя произвольную информацию в полях для ввода идентификатора пользователя и пароля. Очевидно, что это самая опасная конфигурация.

Чтобы разрешить анонимный доступ пользователю без идентификатора пользователя (в сущности, это равносильно полному отсутствию управления доступом), можно воспользоваться следующей директивой:

```
Anonymous_NoOseirID on
```

Б.5.6. Директива Anonymous_VerifyEmail

Синтаксис: Anonymous_VerifyEmail *on* | *off*
Умолчание Anonymous VerifyEmail off
Контекст: каталог, файл .htaccess
Перекрытие: AuthConfig
Статус: Extension
Модуль: mod_auth_anon

Эта директива несет в себе черты чрезмерной самозначительности. При установке ее в on анонимным пользователем вводится псевдопароль, который должен содержать по крайней мере символы "@" и ".", обычно содержащиеся в настоящем адресе электронной почты. К сожалению, это не позволяет избежать ввода адреса наподобие *eat@my.shorts* и определенной неловкости, которую могут вызвать творчество отдельных создателей таких паролей.

Чтобы пользователи вводили псевдопароль, содержащий по крайней мере символы "@" и ".", воспользуйтесь директивой:

```
Anonymous_VerifyEmail on
```

Б.6. Модуль mod auth db

Директивы этого модуля обеспечивают процесс идентификации с помощью файла базы данных Беркли.

Б.6.1. Директива AuthDBGroupFile

Синтаксис: AuthDBGroupFile *имя-файла*
Контекст: каталог, файл .htaccess
Перекрытие: AuthConfig
Статус: Extension
Модуль: mod_auth_db

Эта директива задает абсолютный путь к файлу, содержащему список пользователей и групп, к которым эти пользователи принадлежат. Ключом этого файла является имя пользователя. Запись файла состоит из кода пользователя, за ним следует разделенный запятыми список групп, к которому этот пользователь относится. Явное указание пробела или двоеточия запрещено. Этот файл должен храниться вне файловой иерархии, которую он защищает, и быть недоступным для внешнего мира.

Чтобы задать размещение файла группы mod_auth_db в каталоге /var/secure/dbgroup, необходимо:

```
AuthDBGroupFile    /var/secure/dbgroup
```

Б.6.2. Директива AuthDBUserFile

Синтаксис: AuthDBUserFile *имя-файла*
Контекст: каталог, файл .htaccess
Перекрытие: AuthConfig
Статус: Extension
Модуль: mod_auth_db

Эта директива задает абсолютный путь к файлу, содержащему список пользователей и паролей, необходимых для их идентификации. Ключом этого файла является имя пользователя, за которым следует зашифрованный с помощью функции crypt () пароль. За этой парой следует двоеточие и далее произвольная информация. Все, что следует за двоеточием, игнорируется. Этот файл должен храниться таким образом, чтобы быть недоступным внешнему миру.

Чтобы задать размещение файла пользователей mod_auth_db в каталоге /var/secure/userdb, необходимо:

```
AuthDBUserFile  /var/secure/userdb
```

Б.6.3. Директива AuthDBAuthoritative

Синтаксис: AuthDBAuthoritative *on|off*
Контекст: каталог, файл .htaccess
Перекрытие: AuthConfig
Статус: Base
Модуль: mod_auth_db

Если эта директива установлена в on, в доступе на этом уровне будет отказано полностью. Если задано значение off, то к ресурсу можно получить доступ с помощью механизма идентификации какого-то модуля нижнего уровня.

Если необходимо, чтобы модуль mod_auth_db был последней инстанцией *е* процедуре идентифика-, ции, директива AuthDBAuthoritative должна быть установлена в on.

```
AuthDBAuthoritative on
```

Б.7. Модуль mod_auth_dbm

Директивы, имеющиеся в этом модуле, предназначены для идентификации с помощью файлов базы данных DBM. Обсуждение этого метода иденитификации пользователей можно найти в главе 8, "Безопасность".

Б.7.1. Директива AuthDbmGroupFile

Синтаксис: AuthDbmGroupFile *имя-файла*
Контекст: каталог, файл .htaccess
Перекрытие: AuthConfig
Статус: Extension
Модуль: mod_auth_dbm

Эта директива задает абсолютный путь кфайлу базы данных DBM, содержащему перечень пользовательских групп в целях их идентификации. Файл групп проиндексирован по имени пользователя. За этим значением следует перечень групп, разделенных запятыми, к которым принадлежит этот пользователь. Пробелы и двоеточия здесь применять нельзя. Этот файл должен храниться вне файловой иерархии, которую он защищает, и не должен быть доступен внешнему миру.

Чтобы задать групповой файл в формате DBM, который размещается в файле /etc/secure/dbmgroup, можно воспользоваться следующей директивой:

```
AuthDbmGroupFile   /etc/secure/dbmgroup
```

Б.7.2. Директива AuthDBMUserFile

Синтаксис: AuthDBMUserFile *имя-файла*
Контекст: каталог, файл .htaccess
Перекрытие: AuthConfig
Статус: Extension
Модуль: mod_auth_dbm

Эта директива определяет абсолютный путь к файлу, содержащему список пользователей и пароли, необходимые для их идентификации. Ключом этого файла является имя пользователя, за которым следует зашифрованный с помощью функции crypt () пароль. За этой парой следует двоеточие и произвольная информация. Все, что следует за двоеточием, игнорируется. Этот файл должен храниться таким образом, чтобы быть недоступным внешнему миру.

Пример

Чтобы задать размещение файла пользователей DBM в каталоге /etc/secure/dbmuser, необходимо:

```
AuthDBMUserFile   /etc/secure/dbmuser
```

Б.7.3. Директива AuthDBMAuthoritative

Синтаксис: AuthDBMAuthoritative < *on(умолчание)* | *off* >
Контекст: каталог, файл .htaccess
Перекрытие: AuthConfig
Статус: Base
Модуль: mod_auth_dbm

Если задано значение off, то к ресурсу можно получить доступ с помощью механизма идентификации какого-то модуля нижнего уровня (это определяется в конфигурационном файле и файле modules.c) в том случае, если не указывается идентификатор пользователя или правила соответствия идентификатора пользователя, представленного клиентом.

Чтобы сделать модуль mod_auth_dbm последней инстанцией в идентификации пользователя, воспользуйтесь следующей директивой:

```
AuthDBMAuthoritative on
```

Б.8. Модуль mod_browser

Директивы, имеющиеся в этом модуле, предназначены для включения возможности установки переменных окружения на основании данных о броузере. Идея заключается в том, чтобы сопровождать различных пользователей по Web-страницам в режиме, оптимальном для данного броузера.

Б.8.1. Директива BrowserMatch

Синтаксис: BrowserMatch *рег_выр amp1 amp2. . .*
Контекст: конфигурация сервера
Статус: Base
Модуль: browser
Совместимость: сервер Apache 1.2 и выше

Эта директива позволяет устанавливать локальные переменные окружения, базируясь на данных из заголовка User-Agent, который передается подключившимся броузером. Первый аргумент является регулярным выражением, которое в случае совпадения будет состоять из оставшихся аргументов, установленных как переменные окружения.

Пример

Чтобы присвоить переменной окружения browser значение "netscape", когда заголовок User-Agent начинается с символической последовательности "Mozilla...", воспользуйтесь следующей директивой:

```
BrowserMatch ^Mozilla forms jpeg=yes browser=netscape
```

Б.8.2. Директива BrowserMatchNoCase

Синтаксис: BrowserMatchNoCase *регулярное_выражение*
Контекст: конфигурация сервера
Статус: Base
Модуль: mod_cern_meta
Совместимость: сервер Apache 1.2 и выше

Директива BrowserMatchNoCase семантически идентична директиве BrowserMatch.

Предлагаемая вашему вниманию директива установит переменную окружения platform в значение "windows" в том случае, если строка User-Agent содержит символическую последовательность "win":

```
BrowserMatchNoCase win platform=windows
```

Б.9. Модуль mod_cern_meta

Директивы, имеющиеся в этом модуле, позволяют добавлять в заголовки, обычно выводимые в файле, заголовок метафайла CERN HTTPD.

Б.9.1. Директива MetaDir

Синтаксис: MetaDir *каталог*
Умолчание: MetaDir .web
Контекст: конфигурация сервера
Статус: Base
Модуль: mod_cern_meta
Совместимость: сервер Apache 1.1 и выше

Эта директива предназначена для указания места в файловой системе, где будут храниться метафайлы, которые могут использоваться в качестве заголовков. Обычно это бывает подкаталог каталога, хранящего файлы, к которым осуществляется доступ.

Чтобы указать, что метафайлы должны храниться в том же самом каталоге, что и файлы, к которым осуществляется доступ, необходимо задать директиву следующего вида:

```
Metadir .
```

Б.9.2. Директива MetaSuffix

Синтаксис: MetaSuffix *suffix*
Умолчание: MetaSuffix .meta
Контекст: конфигурация сервера
Статус: Base
Модуль: mod_cern_meta
Совместимость: сервер Apache 1.1 и выше

Эта директива задает суффикс файла, содержащего метаданные.

Если это значение установлено в значение ".meta", сервер Apache при обслуживании файла index.html возвратит в качестве заголовка содержимое файла index.html.meta. Чтобы активизировать такое поведение сервера, можно воспользоваться следующей директивой:

```
MetaSuffix .meta
```

Б.10. Модуль mod_cgi

Модуль mod_cgi позволяет возвращать динамически генерируемое содержимое, которое создается исполняемыми файлами или компилированными программами (см. главу 9, "Динамические Web-страницы).

Б.10.1. Директива scriptLog

Синтаксис: ScriptLog *имя-файла*
Умолчание: нет
Контекст: конфигурация ресурса
Модуль: mod_cgi

Эта директива предназначена для задания файла, в котором будут регистрироваться ошибки работы CGI-сценариев. ЕСЛИ ЭТУ ПЕРЕМЕННУЮ ОСТАВИТЬ НЕНАЗНАЧЕННОЙ, ТО ОШИБКИ НЕ БУДУТ РЕГИСТРИРОВАТЬСЯ ВООБЩЕ. Журнал регистрации ошибок должен быть доступен для записи с помощью CGI-процессов. Эта директива больше подходит для применения в качестве временного мероприятия для помощи в процессе отладки.

Чтобы сервер Apache начал регистрировать ошибки, возникающие при работе CGI-сценариев, в файл /var/logs/error.txt, воспользуйтесь следующей директивой:

```
ScriptLog  /var/logs/error.txt
```

Б.10.2. Директива ScriptLogLength

Синтаксис: ScriptLogLength *размер*
Умолчание: 10385760
Контекст: конфигурация ресурса
Модуль: mod_cgi

Эта директива предназначена для задания наибольшего размера (в байтах) регистрационного журнала. После того как файл достигнет этого размера, информация записываться не будет.

Чтобы задать размер журнала ScriptLog равным приблизительно 20 Мбайт, воспользуйтесь следующей директивой:

```
ScriptLogLength  20000000
```

Б.10.3. Директива ScriptLogBuffer

Синтаксис: ScriptLogBuffer *размер*
Умолчание: 1024
Контекст: конфигурация ресурса
Модуль: mod_cgi

Эта директива предназначена для задания верхнего предела длины тел зарегистрированных директив PUT или POST. Это позволяет предотвратить слишком быстрое увеличение размера файла.

Чтобы увеличить стандартное значение на 2048 байт, воспользуйтесь директивой:

```
ScriptLogBuffer  2048
```

Б.11. Модуль mod_digest

Этот модуль позволяет производить идентификацию пользователей с помощью алгоритма цифровой идентификации MD5. Он не компилируется по умолчанию. Более подробную информацию об этом типе идентификации можно найти в главе 8, "Безопасность".

Б.11.1. Директива AuthDigestFile

Синтаксис: AuthDigestFile *имя-файла*
Контекст: каталог, файл .htaccess
Перекрытие: AuthConfig
Статус: Base
Модуль: mod_digest
Совместимость: сервер Apache 1.1 и выше

Эта директива используется для указания абсолютного пути к файлу, содержащему список пользователей, отформатированный для идентификации MD5, и зашифрованных паролей. Следует заметить, что директива AuthType должна быть установлена в значение "Digest", а вместо директивы AuthUserFile придется воспользоваться директивой AuthDigestFile.

Пример

Чтобы задать каталог /etc/secure/authdigest для размещения файла auth_digest:

```
AuthDigestFile /etc/secure/authdigest
```

Б.12. Модуль mod_dir

Директивы, содержащиеся в этом модуле, предназначены для управления методами, с помощью которых создаются и отображаются индексы каталогов. Стандартным местом, откуда они берутся, является файл index.html. Однако имя этого файла можно изменить. С другой стороны, сервер Apache можно сконфигурировать таким образом, чтобы он генерировал и форматировал листинг файлов прямо в каталоге.

Б.12.1. Директива AddAlt

Синтаксис: AddAlt *строка файл файл. . .*
Контекст: конфигурация сервера, виртуальный узел, каталог, файл .htaccess
Перекрытие: Indexes
Статус: Base
Модуль: mod_dir

Эта директива может быть использована для того, чтобы рядом с файлом отображалась текстовая строка, а не рисунок. Для этого значение директивы FancyIndexing должно быть установлено в on.

Б.12.2. Директива AddAltByEncoding

Синтаксис: AddAltByEncoding *строка MIME-кодирование. . .*
Контекст: конфигурация сервера, виртуальный узел, каталог, файл .htaccess
Перекрытие: Indexes
Статус: Base
Модуль: mod_dir

Эта директива может быть использована для того, чтобы рядом с файлом вместо стандартной пиктограммы для файлов с заданным MIME-кодированием отображалась текстовая строка. Для этого значение директивы FancyIndexing должно быть установлено в on.

Пример

Чтобы рядом с файлами MIME-типа x-compress отображалась строка "compress", достаточно:

```
AddAltByEncoding "compress" x-compress
```

Б.12.3. Директива AddAltByType

Синтаксис: AddAltByType *строка MIME-тип MIME-тип. . .*
Контекст: конфигурация сервера, виртуальный узел, каталог, файл .htaccess
Перекрытие: Indexes
Статус: Base
Модуль: mod_dir

Эта директива используется для того, чтобы рядом с файлом вместо стандартной пиктограммы для файлов с заданным значением *MIME-тип* отображалась текстовая строка. Для этого значение директивы FancyIndexing должно быть установлено в on.

Пример

Следующая директива будет отображать строку "text или html" для файлов MIME-типа text/html:

```
AddAltByType "text or html" text/html
```

Б.12.4. Директива AddDescription

Синтаксис: AddDescription *строка файл файл . . .*
Контекст: конфигурация сервера, виртуальный узел, каталог, файл .htaccess
Перекрытие: Indexes
Статус: Base
Модуль: mod_dir

Эта директива используется для привязки описания изображения к файлу. Параметр *файл* может быть как полным, так и неполным именем файла, групповым символом или расширением описываемого файла.

Чтобы подключить описание "Foolish Young Dogs" к файлу изображения /web/pics/housepets.gif, необходимо:

```
AddDescription "Foolish Young Dogs" /web/pics/housepets.gif
```

Б.12.5. Директива Addicon

Синтаксис: Addicon *пиктограмма файл файл . . .*
Контекст: конфигурация сервера, виртуальный узел, каталог, файл .htaccess
Перекрытие: Indexes
Статус: Base
Модуль: mod_dir

Эта директива используется для указания пути или URL к файлу, содержимое которого будет отображаться рядом с указанным файлом. Имя может быть определено ^^DIRECTORY, ^^BLANKICON, расширением файла, групповым символом, неполным или полным именем файла. Значение *пиктограмма* может быть задано относительным URL или в формате (alttext, url), где alttext является текстовой строкой, которая будет использоваться для отображения неграфическими броузерами.

Для отображения пиктограммы, сохраненной в сценарии script.xbm, соответствующей файлам с расширением .pl или .ksh, воспользуемся директивой:

```
Addicon /icons/script .xbm .pl .ksh
```

Б.12.6. Директива AddIconByEncoding

Синтаксис: AddIconByEncoding *пиктограмма MIME-кодирование MIME-кодирование*
Контекст: конфигурация сервера, виртуальный узел, каталог, файл .htaccess
Перекрытие: Indexes
Статус: Base
Модуль: mod_dir

Эта директива используется для задания пиктограммы, которая будет отображаться рядом с файлами, имеющими определенное MIME-кодирование. Значение *пиктограмма* задается обычно URL с символом "%" относительно пиктограмммы, или в формате (alttext, url), где alttext является текстовым тегом пиктограммы для неграфических броузеров.

Для отображения пиктограммы comprees.xbm для файлов MIME-типа x-compress, воспользуйтесь следующей директивой:

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

Б.12.7. Директива AddIconByType

Синтаксис: AddIconByType *пиктограмма MIME-кодирование MIME-кодирование ...*
Контекст: конфигурация сервера, виртуальный узел, каталог, файл .htaccess
Перекрытие: Indexes
Статус: Base
Модуль: mod_dir

Эта директива используется для задания пиктограммы, которая будет отображаться рядом с файлами, имеющими определенный MIME-тип. Значение *пиктограмма* может быть задано URL с символом "%" относительно пиктограммы, или в формате (строка, url), где строка является текстовым тегом пиктограммы для неграфических броузеров. MIME-тип является групповым выражением, соответствующим требуемым MIME-типам.

Пример

Для отображения текстовой строки IMG вместо файла изображений /icons/image.xbm, который соответствует MIME-типу image/*, воспользуйтесь следующей директивой:

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

Б.12.8. Директива DefaultIcon

Синтаксис: DefaultIcon *url*
Контекст: конфигурация сервера, виртуальный узел, каталог, файл .htaccess
Перекрытие: Indexes
Статус: Base
Модуль: mod_dir

Эта директива предназначена для пиктограммы, которая будет отображаться в процессе FancyIndexing, если нет специальных пиктограммы.

Чтобы задать отображение пиктограммы, находящейся по адресу /icon/unknown.xbm при отсутствии соответствующей директивы, воспользуйтесь следующей директивой:

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

Б. 12.9. Директива DirectoryIndex

Синтаксис: DirectoryIndex *локальный-urlлокальный-url*
Умолчание: DirectoryIndex index.html
Контекст: конфигурация сервера, виртуальный узел, каталог, файл .htaccess
Перекрытие: Indexes
Статус: Base
Модуль: mod_dir

Обычно эта директива задает имя первого файла, который будет отображаться при доступе к каталогу (например index.hmtl). Этим можно воспользоваться для задания альтернативного отображаемого или исполняемого файла.

Следующая директива будет выполнять CGI-сценарий index.pl, если в каталоге не существует ни файла index.html, НИ файла alt-index, html:

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

Б. 12.10. Директива FancyIndexing

Синтаксис: FancyIndexing < *on* | *off* >
Контекст: конфигурация сервера, виртуальный узел, каталог, файл .htaccess
Перекрытие: Indexes
Статус: Base
Модуль: mod_dir

Эта директива предназначена для включения/отключения режима FancyIndexing.

Пример

Чтобы включить режим FancyIndexing, воспользуйтесь следующей директивой:

```
FancyIndexing on
```

Б.12.11. Директива HeaderName

Синтаксис: HeaderName *имя-файла*
Контекст: конфигурация сервера, виртуальный узел, каталог, файл .htaccess
Перекрытие: Indexes
Статус: Base
Модуль: mod_dir

Эта директива предназначена для определения файлов, которые будут вставляться в заголовок листингов каталогов. Имя файла задается относительно индексируемого каталога.

Чтобы вставить файл site-banner.html в начало индексного листинга, воспользуйтесь следующей директивой:

```
HeaderName    site-banner.html
```

Б.12.12. Директива IndexIgnore

Синтаксис: Indexlgnore *файл файл . . .*
Контекст: конфигурация сервера, виртуальный узел, каталог, файл .htaccess
Перекрытие: Indexes
Статус: Base
Модуль: mod_dir

Эта директива предназначена для определения файлов (таких как исполняемые сценарии), которые не будут включены в генерируемые листинги каталогов. Файлы могут задаваться как с полным, так и с относительным именем файла, групповыми выражениями или расширениями файлов.

Чтобы сервер Apache проигнорировал установки, сделанные в файле .htaccess и во всех файлах, завершающихся расширением . pi во время создания листингов каталогов, воспользуйтесь следующей директивой:

```
Indexlgnore  README  .htaccess  .pl
```

Б.12.13. Директива IndexOptions

Синтаксис: AuthDigestFile *имя-файла*
Контекст: конфигурация сервера, виртуальный узел, каталог, файл .htaccess
Перекрытие: Indexes
Статус: Base
Модуль: mod_dir

Эта директива предназначена для управления процессом индексирования. Перечень опций можно найти в табл. Б.5.

Таблица Б.5. Опции директивы IndexOptions

Опция	*Действие*
FancyIndexing	Включает фиктивное индексирование каталогов.
IconsAreLinks	Делает пиктограммы частью привязки имени файла для фиктивного индексирования.
ScanHTMLTitles	Выбирает титул документа из HTML-документа при генерировании листингов фиксированных индексов. Предупреждение: вызывает сильное падение производительности.
SuppressLastModifled	SuppSDo не включает дату последней модификации при генерировании листингов фиктивного индексирования.
SuppressSize	Не указывает размер файлов в листингах фиктивного индексирования.
Suppress Description	Не указывает описание файлов в листингах фиктивного индексирования.

По умолчанию опции неактивны.

Для активизации фиктивного индексирования и отключения описания документов в сгенерированном описании необходимо воспользоваться следующей директивой:

```
<Directory /home/site2/>
    IndexOptions FancyIndexing SuppressDescription
<Directory>
```

Б.12.14. Директива ReadmeName

Синтаксис: ReadmeName *имя-файла*
Контекст: конфигурация сервера, виртуальный узел, каталог, файл .htaccess
Перекрытие: Indexes
Статус: Base
Модуль: mod_dir
Совместимость: сервер Apache 1.1 и выше

Эта директива предназначается для задания имени файла, который будет добавлен в конец листинга каталога.

Чтобы добавить файл readme в конец каждого генерируемого листинга каталога, можно воспользоваться следующей директивой:

```
ReadmeName README
```

Б.13. Модуль mod_env

Директивы этого модуля используются в основном для передачи переменных окружения CGI-сценариям, для получения значений переменных окружения отсистемных процессов, которые вызвали сервер Apache.

Б.13.1. Директива PassEnv

Синтаксис: PassEnv *переменная переменная . . .*
Контекст: конфигурация сервера, виртуальный узел
Статус: Base
Модуль: mod_env
Совместимость: директива PassEnv имеется только на сервере Apache 1.1 и выше

Этой директивой можно воспользоваться для установки переменных окружения сервера, которые будут переданы CGI-сценариям прямо из окружения сервера.

Чтобы передать переменную окружения INFORMIXDIR порожденному процессу, воспользуйтесь следующей директивой:

```
PassEnv INFORMIXDIR
```

Б.13.2. Директива SetEnv

Синтаксис: SetEnv *переменная переменная . . .*
Контекст: конфигурация сервера, виртуальный узел
Статус: Base
Модуль: mod_env
Совместимость: директива SetEnv имеется только на сервере Apache 1.1 и выше

Эта директива предназначена для установки переменных окружения, которые будут переданы CGI-сценариям.

Чтобы установить переменную окружения INFORMIXDIR с целью ее передачи порожденному процессу, воспользуйтесь следующей директивой:

```
SetEnv INFORMIXDIR /usr/local/informix
```

Б.13.3. Директива UnsetEnv

Синтаксис: UnsetEnv
Контекст: конфигурация сервера, виртуальный узел
Статус: Base
Модуль: mod_env
Совместимость: директива UnsetEnv имеется только на сервере Apache 1.1 и выше.

Этой директивой можно воспользоваться для удаления переменных окружения из набора переменных окружения, которые передаются CGI-сценариям.

Чтобы удалить переменную INFORMIXDIR из набора переменных окружения, которые передаются порожденным процессам, воспользуйтесь следующей директивой:

```
UnsetEnv INFORMIXDIR
```

Б.14. Модуль mod_expires

Директивы из этого модуля используются для управления значением Expires заголовка HTTP, которое передается клиенту.

Б.14.1. Директива ExpiresActive

Синтаксис: ExpiresActive < *on* | *off* >
Контекст: конфигурация сервера, виртуальный узел
Перекрытие: Indexes
Статус: Extension
Модуль: mod_expires

Эта директива используется для включения или отключения режима установки значения Expires заголовка HTTP.

Чтобы включался заголовок HTTP Expires, воспользуйтесь следующей директивой:

```
ExpiresActive on
```

Б.14.2. Директива ExpiresByType

Синтаксис: ExpiresByType *MIME-тип <код> секунд*
Контекст: конфигурация сервера, виртуальный узел, каталог, файл .htaccess
Перекрытие: Indexes
Статус: Extension
Модуль: mod_expires

Эта директива используется для задания значения Expires в заголовке HTTP, возвращаемого клиентскому процессу. Значением *<код>* может быть "M", которая показывает, что значение задано относительно времени последней модификации, или "A", что показывает, что значение задано относительно времени последнего доступа.

Чтобы указать, что срок хранения документов html истекает через неделю после последнего доступа к ним, воспользуйтесь директивой:

```
ExpiresByType text/html A604800
```

Б.14.3. Директива ExpiresDefault

Синтаксис: ExpiresDefault *<код> секунд*
Контекст: конфигурация сервера, виртуальный узел, каталог, файл .htaccess
Перекрытие: Indexes
Статус: Extension
Модуль: mod_expires

Эта директива предназначена для определения алгоритма по умолчанию, который будет использоваться для указания срока хранения всех документов заданной области. Значения кодов аналогичны значениям, которые используются в директиве `ExpiresByType`.

Пример

Чтобы задать стандартное значение срока хранения равным одному месяцу с момента последней модификации документа, необходимо задать директиву:

```
ExpiresDefault M2592000
```

Б.15. Модуль mod_headers

Этот модуль имеет только одну директиву, директиву Header, которая может использоваться для расшифровки стандартных заголовков HTTP.

Б.15.1. Директива Header

Синтаксис: Header [set | append | add | unset] *значение-заголовка*
Умолчание: Header unset header
Перекрытие: Indexes
Статус: Optional
Модуль: mod_header

Эта директива предназначена для замены, слияния или удаления заголовков ответа HTTP. Допустимые действия перечислены в табл. Б.6.

Таблица Б.6. Действия директивы Header

Действие	*Результат*
set	Заменить заданный заголовок (если он существует) новым заголовком.
append	Добавить заданный заголовок к любому существующему одноименному заголовку.
add	Добавить заданный заголовок к существующему списку заголовков, даже если таковой уже существует.
unset	Удалить первый попавшийся заголовок с заданным именем.

Чтобы задать значение заголовка сервера "Fenris", перекрыв любое существующее значение, воспользуемся директивой:

```
Header set Server "Fenris"
```

Б.16. Модуль mod_imap

Этот модуль предназначен для использования функций программы отображения CGI. Она предоставляет дескриптор файла размещения, который включен в стандартный дистрибутив.

Imagemaps -- это изображения с четко заданными компонентами, по которым можно щелкать, чтобы получить индивидуальную или покомпонентную реакцию.

Б.16.1. Директива ImapMenu

Синтаксис: ImapMenu *option*
Контекст: конфигурация сервера, виртуальный узел, каталог, файл .htaccess
Перекрытие: Indexes
Модуль: mod_map
Совместимость: директива ImapMenu имеется только на сервере Apache 1.1 и выше

Этой директивой можно воспользоваться для определения предпринимаемой реакции, когда imap-файл вызывается без координат. Опции перечислены в табл. Б.7.

Таблица Б.7. Опции директивы ImapMenu

Опция	*Действие*
none	Меню не генерируется, предпринимаются стандартные действия.
formatted	Комментарии в файле imagemap игнорируются. Печатается заголовок первого уровня, затем печатается hrule, затем связи, по одной на каждую строку. Меню имеет обычный вид.
semi formatted	Комментарии печатаются везде, где они встречаются в файле. Пустые строки преобразуются в HTML-разрывы. Заголовок или hrule не печатаются, в противном случае меню имеет тот же вид, что и меню с опцией formatted.

Окончание табл. Б. 7

Опция	*Действие*
unformatted	Комментарии печатаются, пустые строки игнорируются. То, что отсутствует в файле imagemap, не печатается. Все разрывы и заголовки включаются в файл imagemap как комментарии. Это предоставляет больше гибкости в виде меню, но заставит рассматривать ваши imap-файлы как HTML-файлы, а не как простые текстовые файлы.

Б.16.2. Директива ImapDefault

Синтаксис: ImapDefault *[error, nocontem, referer, URL]*
Контекст: конфигурация сервера, виртуальный узел, каталог, файл .htaccess
Перекрытие: Indexes
Модуль: mod_map
Совместимость: директива ImapDefault имеется только на сервере Apache 1.1 и выше

Эту директиву необходимо использовать для задания стандартных установок в файлах imagemap. Если они остаются незаданными, то по умолчанию принимается nocontent, что означает: клиенту отсылается сообщение "204 No Content".

Чтобы по умолчанию файлом imagemap был mysite.html, воспользуемся директивой:

```
ImapDefault mysite.html
```

Б.16.3. Директива ImapBase

Синтаксис: ImapBase *[map, referer, URL]*
Контекст: конфигурация сервера, виртуальный узел, каталог, файл .htaccess
Перекрытие: Indexes
Модуль: mod_map
Совместимость: директива ImapBase имеется только на сервере Apache 1.1 и выше

Эта директива устанавливает базу, которая будет использоваться в файле imagemap по умолчанию. Эта установка отменяется директивой base в файле imagemap.

Чтобы задать каталог /images/ в качестве каталога базы image, можно воспользоваться следующей директивой:

```
ImapBase /images/
```

Б.17. Модуль mod_include

Модуль mod_include обеспечивает динамическое изменение HTML-документов на стороне сервера. Обсуждение этой темы можно найти в главе 9, "Динамические Web-страницы.

Б.17.1. Директива XBitHack

Синтаксис: XBitHack *статус*
Умолчание: XBitHack off
Контекст: конфигурация сервера, виртуальный узел, каталог, файл .htaccess
Перекрытие: Options
Модуль: mod_include

При установке в on или full директива XBitHack позволяет обработку обычных HTML-документов (имеющих MIME-тип text/html). Значения, которые может принимать параметр *статус,* перечислены в табл. Б.8.

Таблица Б.8. Значение параметра статус директивы XbitHack

Значение параметра	*Действие*
off	Нет.
on	**Модулем** mod_include **будут проанализированы все файлы, у которых установлен** бит "user-execute".
full	**Модулем** mod_include **будут проанализированы все файлы, у которых установлены биты** "user-execute" и "group-execute".

Б.18. Модуль mod_info

Модуль mod_info позволяет администратору (а потенциально — всем желающим) получить доступ к информации о сервере в виде Web-страницы. Подробно эта проблема обсуждается в главе 7, "Регистрация и мониторинг".

Б.18.1. Директива AddModuleInfo

Синтаксис: AddModuleInfo *имя-модуля строка*
Контекст: конфигурация сервера, виртуальный узел
Статус: Base
Модуль: mod_browser
Совместимость: директива AddModuleInfo имеется только на сервере Apache 1.3 и выше

Эта директива позволяет при выводе включить дополнительную информации об имени заданного модуля.

Чтобы задать гипертекстовую связь на информацию о модуле, когда отображается экран данных, mod_info, воспользуйтесь следующей директивой:

```
AddModuleInfo mod_auth.c 'См. <A HREF="http://www.apache.org/docs/
mod/mod_info.html">http://www.apache.org/docs/mod/mod_info.html</A>'
```

Б.19. Модуль mod_isapi

Модуль mod_isapi обеспечивает поддержку расширений интерфейса программирования приложений для интернет-сервера (Internet Server Application Programming Interface — ISAPI) при работе сервера Apache в средах Win32. Чтобы запустить работу

этого модуля, необходимо связать дескриптор isapi-isa с файлами, имеющими расширение dll:

```
AddHandler isapi-isa dll
```

Б.20. Модуль mod_log_agent

Модуль mod_log_agent обеспечивает регистрацию клиентских пользовательских агентов в соответствии с заголовками UserAgent поступающих запросов.

Б.20.1. Директива AgentLog

Синтаксис: AgentLog *файл-конвейер*
Умолчание: AgentLog logs/agent_log
Контекст: конфигурация сервера, виртуальный узел
Статус: Extension
Модуль: mod_log_agent

Эта директива задает файл, в котором сервер Apache регистрирует входящие запросы пользовательских агентов. Обратите внимание на то, что в дополнение к обычным именам файлов (относительные или абсолютные пути), эта директива принимает также значение *конвейер* с последующей системной командой или именем исполняемой программы (предположительно для получения этого типа данных со стандартного устройства ввода).

Чтобы переслать полученные заголовки user_Agent на локальную счетную программу countagent.pl, можно воспользоваться следующей директивой:

```
AgentLog | countagent.pl
```

Б.21. Модуль mod_log_common

Начиная с версии Apache 1.2, модуль mod_log_common **был заменен модулем** mod_log_config.

Б.22. Модуль mod_log_config

Модуль mod_log_config обеспечивает регистрацию запросов к серверу в формате Common Log Format. Кроме того, с помощью директивы LogFormat этот формат может быть настроен любым образом. Более подробную информацию о модуле можно найти в главе 7, "Регистрация и мониторинг".

Б.22.1. Директива CookieLog

Синтаксис: CookieLog *имя-файла*
Контекст: конфигурация сервера, виртуальный узел
Модуль: mod_cookies
Совместимость: имеется только на сервере Apache 1.2 и выше

Эта директива используется для регистрации файлов cookies.

Для регистрации файлов cookies в файле /var/iog/cookies, можно воспользоваться следующейдирективой:

```
CookieLog  /var/log/cookies
```

Б.22.2. Директива Custom_Log

Синтаксис: CustomLog *файл-конвейер формат-или-псевдоним* env= [!] *переменная-окружения*
Контекст: конфигурация сервера, виртуальный узел
Статус: Base
Модуль: mod_log_config
Совместимость: имеется только на сервере Apache 1.3 и выше

Эта директива используется для задания файла (с абсолютным или относительным путем), в который будет записываться регистрационная информация. Если формат содержит пробелы, пробел должен быть заключен в двойные кавычки. Допустимо использование псевдонима формата, определенного директивой LogFormat. При необходимости регистрацию можно сделать условной в зависимости от значения переменной окружения, включив в качестве аргумента имя этой переменной и требующееся значение переменной.

Чтобы зарегистрировать дату, узел, клиента и количество байт, возвращаемых в файл /var/log/bytelog, воспользуйтесь следующей директивой:

```
CustomLog /var/log/bytelog "date host  ident byte"
Browser=netscape
```

Пример

Чтобы зарегистрировать дату, узел, клиента и количество байт, возвращаемых броузеру Netscape в файл /var/log/netscape, воспользуйтесь следующей директивой (здесь предполагается, что переменная окружения броузера устанавливается на основании типа броузера):

```
CustomLog /var/log/bytelog "date host  ident byte"
Browser=netscape
```

Б.22.3. Директива LogFormat

Синтаксис: LogFormat *формат [псевдоним]*
Умолчание: LogFormat "%h %l %u %t "\^r\" %s %b"
Контекст: конфигурация сервера, виртуальный узел
Статус: Base
Модуль: mod_log_config
Совместимость: имеется только на сервере Apache 1.3 и выше

Эта директива используется для задания формата журнала, созданного директивой TransferLog. Отметим, что при подключении псевдонима к формату, определенному с помощью этой директивы, псевдонимом можно будет пользоваться и в других директивах. Значения переменных, которые используются в этом формате, приводятся в главе 7, "Регистрация и мониторинг".

Директива LogFormat, задающая псевдоним, не несет никаких других нагрузок. Псевдоним можно указать в любом месте.

Чтобы задать псевдоним standard для форматной строки "%h %l %u %t \"%r\" %s %b" воспользуйтесь следующей директивой:

```
LogFormat "%h %l %u %t \"%r\" %s %b" standard
```

Б.22.4. Директива TransferLog

Синтаксис: TransferLog *файл-конвейер*
Умолчание: отсутствует
Контекст: конфигурация сервера, виртуальный узел
Статус: Base
Модуль: mod_log_config

Эта директива задает регистрационный файл, в который будет записываться информация из TransferLog. С другой стороны, с помощью конвейера Unix вывод можно послать на другую программу.

Для передачи данных, полученных директивой TransferLog, программе wc, воспользуемся директивой:

```
TransferLog                    |                    wc
```

Б.23. Модуль mod_log_referer

Этот модуль существовал и использовался в сервере Apache вплоть до версии 1.3.5. Для регистрации документов, ссылающихся на документы, находящиеся на сервере, рекомендуется пользоваться условной директивой CustomLog.

Б.24. Модуль mod_mime

Этот модуль обеспечивает механизм определения MIME-типов на основании расширений файлов. Тема дескрипторов нашла свое освещение в главе 1, "Основные концепции".

Б.24.1. Директива AddEncoding

Синтаксис: AddEncoding *mime-кодирование расширение расширение. . .*
Умолчание: отсутствует
Контекст: конфигурация сервера, виртуальный узел, каталог, файл .htaccess
Перекрытие: FileInfo
Статус: Base
Модуль: mod_mime

Эта директива используется для того, чтобы сообщить серверу Apache, какое расширение соответствует определенному *MIME-типу*.

Чтобы связать MIME-тип x-gzip с расширением .gz, можно воспользоваться следующей директивой:

```
AddEncoding x-gzip .gz
```

Б.24.2. Директива AddHandler

Синтаксис: AddHandler *имя-дескриптора расширение расширение...*
Контекст: конфигурация сервера, виртуальный узел, каталог, файл .htaccess
Статус: Base
Модуль: mod_mime
Совместимость: имеется только на сервере Apache 1.1 и выше

Эта директива используется для того, чтобы сообщить серверу Apache, что файлы с указанным расширением будут переданы определенному дескриптору.

Чтобы связать дескриптор `cgi-script` с файлом, имеющим расширение .pl, можно воспользоваться следующей директивой:

```
AddHandler cgi-script cgi
```

Б.24.3. Директива AddLanguage

Синтаксис: AddLanguage *mime-lang расширение расширение. . .*
Контекст: конфигурация сервера, виртуальный узел, каталог, файл .htaccess
Перекрытие: FileInfo
Статус: Base
Модуль: mod_mime

Эта директива используется длятого, чтобы связать расширения файлов (например .en, .fr) с внутренними языковыми представлениями (например en, fr) модуля mod_mime. Обычно используется при согласовании содержимого.

Чтобы связать расширение .en с английским языком, воспользуемся директивой:

```
AddLanguage en .en
```

Б.24.4. Директива AddType

Синтаксис: AddType *mime-тип расширение расширение...*
Контекст: конфигурация сервера, виртуальный узел, каталог, файл .htaccess
Перекрытие: FileInfo
Статус: Base
Модуль: mod_mime

Директива AddType используется для добавления новых расширений файлов к списку расширений, которыми завершаются имена файлов, являясь при этом файлами определенного MIME-типа.

Чтобы указать, что файлы с расширением GIF имеют MIME-тип `image/gif`, воспользуемся директивой:

```
AddType image/gif GIF
```

Б.24.5. Директива ForceType

Синтаксис: ForceType *тип_ среды*
Контекст: каталог, файл .htaccess
Статус: Base
Модуль: mod_mime
Совместимость: имеется только на сервере Apache 1.1 и выше

Эта директива может использоваться только внутри скобок <Directory>, <Location> или в файле .htaccess для того, чтобы сервер Apache рассматривал все файлы, размещенные в определенном месте, как файлы строго определенного типа.

Чтобы сервер Apache рассматривал файлы в каталоге /usr/local/images как файлы MIME-типа image/gif, воспользуйтесь следующей директивой:

```
<Directory  /usr/local/images>
     ForceType  image/gif
</Directory>
```

Б.24.6. Директива SetHandler

Синтаксис: SetHandler ***имя-дескриптора***
Контекст: **каталог, файл** .htaccess
Статус: Base
Модуль: mod_mime
Совместимость: имеется только на сервере Apache 1.1 и выше

Эта директива используется для указания на то, что файлы в определенном каталоге (указывается внутри операторных скобок <Directory>, <Location> или в файле .htaccess) должны обрабатываться строго определенным дескриптором.

Чтобы сервер Apache применял дескриптор cgi-script ко всем файлам, расположенным в каталоге /usr/local/cgi-bin, нужно применить следующую директиву:

```
<Directory /usr/local /images>
      SetHandler cgi-script
</Directory>
```

Б.24.7. Директива TypesConfig

Синтаксис: TypesConlig *имя-файла*
Умолчание: TypesConfig conf/mime.types
Контекст: конфигурация сервера
Статус: Base
Модуль: mod_mime

Эта директива используется для того, чтобы задать размещение конфигурационных файлов MIME-типов. Это место определяется относительно каталога ServerRoot. Вносить изменения в этот файл не рекомендуется. Кроме того, для указания дополнительных MIME-типов, это лучше делать с помощью директивы AddType.

Предположим, что директива ServerRoot задана как /opt/apache, следующая директива покажет, что файл MIME-типов будет размещен в каталоге /opt/apache/conf/MIME.

```
TypesConfig  conf/MIME.types
```

Б.25. Модуль mod_mime_magic

При работающем сервере модуль mod_mime_magic определяет MIME-тип файла, просматривая первые несколько байт его содержимого аналогично тому, как это делает в ОС Unix команда file.

Б.25.1. Директива MimeMagicFile

Синтаксис: MimeMagicFile *имя-файла*
Умолчание: Отсутствует
Контекст: конфигурация сервера, виртуальный узел
Статус: Extension
Модуль: mod_mime_magic

Эта директива предназначена для:

1. активизации модуля,
2. определе ния размещения конфигурационного файла.

Предположим, что директива ServerRoot задана как /opt/apache, следующая директива включает работу модуля mod_mime_magic с конфигурационным файлом, находящимся в каталоге /apt /apache/conf/magic:

```
MimeMagicFile  conf/magic
```

Б.26. Модуль mod_mmap_static

Этот модуль активизирует использование процесса mmap() длястатических файлов, которые часто запрашиваются. См. главу 10, "Настройка рабочих характеристик сервера".

Б.26.1. Дир ектива MMapFile

Синтаксис: MMapFile *имя-файла*
Умолчание: отсутствует
Контекст: конфигурация сервера
Статус: Experimental
Модуль: mod_mime_magic
Совместимость: имеется только на сервере Apache 1.1 и выше

Эта директива предназначена для задания одного или более файлов, которые будут размещены прямо в оперативной памяти во время запуска сервера.

Чтобы разместить /opt/apache/htdocs/index.html в оперативной памяти во время загрузки, воспользуйтесь следующей директивой:

```
MMapFile    /opt/apache/htdocs/index.html
```

Б.27. Модуль mod_negotiation

Этот модуль содержится в файле mod_negotiation.c и компилируется по умолчанию. Он отвечает за процедуру согласования содержимого. Согласование содержимого — это процесс, во время которого сервер производит выбор из нескольких подобных документов и возвращает документ, наиболее полно соответствующий возможностям и потребностям клиента.

Б.27.1. Директива CacheNegotiatedDocs

Синтаксис: CacheNegotiatedDocs
Контекст: конфигурация сервера
Статус: Base
Модуль: mod_negotiation
Совместимость: имеется только на сервере Apache 1.1 и выше

Директива CachNegotiateDocs предназначена для задания режима кэширования согласованных proxy-сервером документов.

Чтобы включить этот режим, достаточно воспользоваться директивой (она не имеет аргументов):

```
CacheNegotiatedDocs
```

Б.27.2. Директива LanguagePriority

Синтаксис: LanguagePriority *MIME-язык MIME-язык. . .*
Контекст: конфигурация сервера, виртуальный узел, каталог, файл .htaccess
Перекрытие: FileInfo
Статус: Base
Модуль: mod_negotiation
Совместимость: имеется только на сервере Apache 1.1 и выше

Эта директива используется для задания приоритета языковой установки в том случае, когда предпочтение не задано.

Чтобы сервер Apache обслуживал документы на английском, французском или немецком языке именно в этом порядке, воспользуемся директивой:

```
LanguagePriority en fr de
```

Б.28. Модуль mod_proxy

Этот модуль реализует режим proxy-сервера Apache. Подробное его описание можно найти в главе 6, "Proxy-серверы и кэширование".

В.29. Модуль mod_rewrite

Б.29.1. Директива RewriteEngine

Синтаксис: RewriteEngine *<on\off>*
Умолчание: RewriteEngine off
Контекст: конфигурация сервера, виртуальный узел, каталог, файл .htaccess
Перекрытие: FileInfo

Статус: Extension
Модуль: mod_rewrite
Совместимость: Apache 1.2

Эта директива используется для включения/отключения механизма перезаписи URL. Для виртуальных узлов такая возможность должна быть объявлена явным образом.

Чтобы активизировать перезапись URL, можно записать такую директиву:

```
RewriteEngine on
```

Б.29.2. Директива RewriteOptions

Синтаксис: RewriteOptions *опция*
Умолчание: нет
Контекст: конфигурация сервера, виртуальный узел, каталог, файл .htaccess
Перекрытие: FileInfo
Статус: Extension
Модуль: mod_rewrite
Совместимость: Apache 1.2

Эта директива используется для того, чтобы установить специальные опции для текущей серверной или каталожной конфигурации. В настоящее время существует только одна опция inherit, которая позволяет осуществлять вложенную конфигурацию (например виртуальные узлы, подкаталоги) для наследования конфигурации, порождающей сущности.

Чтобы задать наследование конфигурации, можно воспользоваться следующей директивой:

```
RewriteOptions inherit
```

Б.29.3. Директива RewriteLog

Синтаксис: RewriteLog *имя-файла*
Умолчание: нет
Контекст: конфигурация сервера, виртуальный узел
Перекрытие: FileInfo
Статус: Extension
Модуль: mod_rewrite
Совместимость: Apache 1.2

Эта директива используется для того, чтобы задать файл, в котором будет регистрироваться любая перезапись. Файл может задаваться как с указанием абсолютного, так и относительного пути.

Чтобы переписать регистрационный файл, расположенный в /var/logs/rewriteiog, прибегните к помощи следующей директивы:

```
RewriteLog /var/logs/rewritelog
```

Б.29.4. Директива RewriteLogLevel

Синтаксис: RewriteLogLevel *уровень*
Умолчание: RewriteLogLevel 0
Контекст: конфигурация сервера, виртуальный узел
Перекрытие: нет
Статус: Extension
Модуль: mod_rewrite
Совместимость: Apache 1.2

Эта директива используется для определения частоты, с которой сервер будет перезаписывать статистику. Значение варьируется от 0 (регистрация не производится) до 9 (производится очень детальная регистрация). Полная регистрация снижает производительность.

Пример

Задать режим регистрации в умеренном режиме:

```
RewriteLogLevel 2
```

Б.29.5. Директива RewriteLock

Синтаксис: RewriteLock *имя-файла*
Умолчание: нет
Контекст: конфигурация сервера
Перекрытие: нет
Статус: Extension
Модуль: mod_rewrite
Совместимость: Apache 1.3

Эта директива определяет файл, который будет использоваться для блокировки при синхронизации. А файл используется для обмена данными с программами `RewriteMap`.

Чтобы задать следующий файл блокировки /var/lock/rewritelock:

```
RewriteLock   /var/lock/rewritelock
```

Б.29.6. Директива RewriteMap

Синтаксис: RewriteMap *имя-карты тип-карты: источник-карты*
Умолчание: по умолчанию не используется
Контекст: конфигурация сервера, виртуальный узел
Перекрытие: не применяется
Статус: Extension
Модуль: mod_rewrite
Совместимость: Apache 1.2 (частично), Apache 1.3

Эта директива используется для определения шаблонов перезаписи, которые используются в правилах замены строк для вставки или замены полей в процессе поиска ключей. Детальное описание этого процесса можно найти в главе 11, "Переназначение адреса".

Б.29.7. Директива RewriteBase

Синтаксис: RewriteBase *BaseURL*
Умолчание: по умолчанию принимается физический путь к каталогу
Контекст: каталог, файл .htaccess
Перекрытие: FileInfo
Статус: Extension
Модуль: mod_rewrite
Совместимость: Apache 1.2

Эта директива используется в контексте конфигурации каталога (<Directory>, файл .htaccess) задания базового URL с помощью каталога, к которому был осуществлен доступ явным образом.

Предположим, что файлы на виртуальном узле физически располагаются на сервере в каталоге /some/directory. Чтобы предупредить механизм перезаписи о том, что к нему можно осуществить доступ с помощью URL /site2, можно использовать файл .htaccess такого вида:

```
RewriteBase /some/directory
```

Б.29.8. Директива RewriteCond

Синтаксис: RewriteCond *тестовая-строка условный-шаблон*
Умолчание. нет
Контекст: конфигурация сервера, виртуальный узел, каталог, файл .htaccess
Перекрытие: FileInfo
Статус: Extension
Модуль: mod_rewrite
Совместимость: Apache 1.2 (частично), Apache 1.3

Эта директива используется для того, чтобы задать условие, при котором активизируется директива RewriteRule. Совместно с RewriteRule можно использовать одну или более директив RewriteCond. Объяснение функциональной нагрузки, которую несут параметры, приводится ниже.

TestString

Строка TestString представляет собой простую текстовую строку, которая может включать следующие конструктивы:

- $N где 0 <= N <= 9. Это обратная ссылка директивы RewriteRule, необходимая для ссылки к символической последовательности, ограниченной скобками соответствующей директивы RewriteRule.
- %N где 1 <= N <= 9. Это позволяет делать ссылку на символическую последовательность, ограниченную скобками, из шаблона последней соответствующей директивы RewriteCond в текущем наборе условий.
- %{ИМЯ_ПЕРЕМЕННОЙ} Здесь используется переменная из приведенного ниже списка. Все эти переменные соответствуют MIME-заголовкам HTTP с аналогичными именами, С-переменным сервера Apache или полям типа struct ОС Unix.

HTTP заголовок	HTTP_USER_AGENT HTTP_REFERER HTTP_COOKIE HTTP_FORWARDED HTTP_HOST HTTP_PROXY_CONNECTION HTTP_ACCEPT
Соединение и Запрос	REMOTE_ADDR REMOTE_HOST REMOTE_USER REMOTE_IDENT REQUEST_METHOD SCRIPT_FILENAME PATH_INFO QUERY_STRING AUTH_TYPE
Внутренние установки сервера	DOCUMENT_ROOT SERVER_ADMIN SERVER_NAME SERVER_ADDR SERVER_PORT SERVER_PROTOCOL SERVER_SOFTWARE
Системные установки	TIME_YEAR TIME_MON TIME_DAY TIME_HOUR TIME_MIN TIME_SEC TIME_WDAY TIME
Специальные установки	API_VERSION THE_REQUEST REQUEST_URI REQUEST_FILENAME IS_SUBREQ

CondPattern представляет собой стандартное расширенное регулярное выражение с использованием выражений, описанных в табл. Б.9.

Таблица Б.9. Дополнения в стандартные расширенные регулярные выражения

Выражение	*Действие*
	Предшествование символа "!" (отрицание) в строке задает шаблон, соответствие которому не требуется.
<CondPattern	Возвращает значение "Истина", если сравниваемая строка лексически меньше, чем заданное значение Condpattern.
>CondPattern	Возвращает значение "Истина", если сравниваемая строка лексически больше, чем заданное значение Condpattern.
=CondPattern	Посимвольное сравнение.
-d	Возвращает значение "Истина", если сравниваемая строка задает путь к существующему каталогу.

Окончание табл. Б.9

Выражение	*Действие*
-f	Возвращает значение "Истина", если сравниваемая строка задает путь к существующему файлу.
-з	Возвращает значение "Истина", если сравниваемая строка задает путь к существующему файлу, размер которого больше 0.
-1	Возвращает значение "Истина", если сравниваемая строка задает путь к символической ссылке.
- F	Возвращает значение "Истина", если сравниваемая строка задает путь к существующему файлу, доступному с сервера. Проверочный тест заключает в себе подзапрос и поэтому снижает производительность сервера в случае, когда он используется достаточно часто.
-U	Возвращает значение "Истина", если сравниваемая строка задает путь к существующему URL.

Кроме того, можно задать дополнительные флаги CondPattern:

nocase INC	Сделать нечувствительным к регистру вводимых букв.
	Скомбинированное правило, как если бы было задано логическое "Или".

Чтобы переписать любой URL, запрошенный текстовым броузером Lynx на страницу textonly.html, можно воспользоваться следующей директивой:

```
RewriteCond %{HTTP_USER_AGENT} ~Lynx.*
RewriteRule ^/$ /textonly.html [L]
```

Б.29.9. Директива RewriteRule

Синтаксис: RewriteRule *шаблон, замещение*
Умолчание: нет
Контекст: конфигурация сервера, виртуальный узел, каталог, файл .htaccess
Перекрытие: FileInfo
Статус Extension
Модуль: mod_rewrite
Совместимость: Apache 1.2 (частично), Apache 1.3 (полностью)

Эту директиву можно использовать для объявления одного правила перезаписи. Заметим, что порядок перечня правил в конфигурационном файле определяет порядок их применения. Шаблон определяется с помощью регулярного выражения, которое применяется к активному в момент вызова правила URL.

Чтобы задать прибавление заключающей косой черты в URL, который ссылается на каталог somedir (проблема замыкающей косой), воспользуйтесь следующей директивой:

```
RewriteEngine on
RewriteRule ^somedir$ somedir/ [R]
```

Б.30. Модуль mod_setenvif

Этот модуль используется для условной установки переменных окружения на основании значений, полученных из запроса клиента.

Б.30.1. Директива BrowserMatch

Синтаксис: BrowserMatch *регулярное-выражение* envar *[=значение] [...}*
Умолчание: , нет
Контекст: конфигурация сервера
Перекрытие: нет
Статус Base
Модуль: mod_setenvif
Совместимость: Apache 1.2 и выше (в Apache 1.2 эту директиву можно найти в отсутствующем сейчас модуле mod_browser)

Эта директива используется для установки некоторых переменных окружения на основании значения Use Agent HTTP из заголовка запроса.

: Чтобы установить переменную среды browser в значение "explorer", когда значение заголовка, User_Agent "MSIE", воспользуйтесь следующей директивой:

```
BrowserMatch MSIE browser=explorer
```

30.2. Директива BrowserMatchNoCase

Синтаксис: BrowserMatchNoCase *регулярное-выражение* envar*[=значение* [...]
Умолчание: нет
Контекст: конфигурация сервера
Перекрытие: нет
Статус base
Модуль: mod_setenvif
Совместимость: Apache 1.2 и выше (в Apache 1.2 эту директиву можно найти в отсутствующем сейчас модуле mod_browser)

Эта директива тождественна директиве BrowserMatch с единственным отличием, которое заключается в том, что сравнение производится без учета регистра данных.

Чтобы установить переменную среды browser в значение "explorer", когда заголовок User_Agent имеет значение "Msie", "msie" или, вероятно, "MsIe", воспользуйтесь следующей директивой: •:•

```
BrowserMatchNoCase MSIE browser=explorer
```

Б.30.3. Директива SetEnvIf

Синтаксис: SetEnvif *атрибут регулярное-выражение* envar*[=значение* [...]
Умолчание: нет
Контекст: конфигурация сервера
Статус Base

Модуль: mod_setenvif
Совместимость: Apache 1.3 и выше (ключевое слово Request_Protocol и сравнение переменной среды есть только в версии 1.3.7 и выше)

Эта директива используется для установки переменных окружения на основании значений, полученных в запросе. Несколько наиболее популярныхзначений перечислено в табл. Б.10.

Таблица Б.10. Условные значения директивы SetEnvIf

Remote_Host	Имя узла клиента, сделавшего запрос.
Remote_Addr	IP-адрес узла клиент а, сделавшего запрос.
Remote_User	Идентифицированное имя пользователя.
Request_Method	Используемый метод (например POST, GET).
Request_Protocol	Название и версия протокола, с помощью которого был произведен запрос (например HTTP/0.9, HTTP/1.1 и т.д.).

• Чтобы установить системную переменную cg i_yn в "y", когда URI получает доступ к Perl-сценарию, воспользуемся директивой:

```
SetEnvIf Request URI "\.pl$" cgi yn=y
```

Б.30.4. Директива SetEnvIfNoCase

Синтаксис: SetEnvIfNoCase
Умолчание: нет
Контекст: конфигурация сервера
Перекрытие: нет
Статус: Base
Модуль: mod_setenvif
Совместимость: Apache 1.3 и выше

Эта директива тождественна директиве SetEnvIf за исключением учета регистров символов.

Б.31. Модуль mod_so

Этот модуль используется для загрузки различных модулей в сервер Apache во время его работы. Отметим, что в отличие от ОС Unix в ОС Windows он по умолчанию установлен в on.

Б.31.1. Директива LoadFile

Синтаксис: LoadFile *имя-файла имя-файла* . . .
Контекст: конфигурация сервера
Статус: Base
Модуль: mod_so

Этот модуль используется для подключения указанного объектного файла или библиотеки в момент запуска или перезапуска сервера.

Б.31.2. Директива LoadModule

Синтаксис: LoadModule *модуль имя-файла*
Контекст: конфигурация сервера
Статус Base
Модуль: mod_so

Этот модуль используется для подключения определенного объектного файла или библиотеки и добавления структуры соответствующего модуля к списку активных файлов.

Чтобы загрузить во время работы сервера модуль mod_status, находящийся в подкаталоге module каталога ServerRoot, воспользуемся директивой:

```
LoadModule status module modules/mod status.so
```

Б.32. Модуль mod_speling

Этот модуль предназначен для исправления ошибок в URL, которые могут допустить пользователи при вводе адреса.

Б.32.1. Директива CheckSpelling

Синтаксис: CheckSpelling on/off
Умолчание: CheckSpelling on
Контекст: конфигурация сервера, виртуальный узел, каталог, файл .htaccess
Перекрытие: Options
Статус Base
Модуль: mod_speling

Эта директива включает/отключает возможность проверки URL.

Для включения режима проверки достаточно задать директиву:

```
CheckSpelling on
```

Б.33. Модуль mod status

Этот модуль предназначен для отображения статистики работы сервера в виде Web-страницы.

Б.33.1. Директива ExtendedStatus

Синтаксис: ExtendedStatus On | Off
Умолчание: ExtendedStatus Off
Контекст: конфигурация сервера
Статус Base
Модуль: mod_status
Совместимость: директива ExtendedStatus имеется только в сервере Apache 1.3.2 и выше

Эта директива предназначена для включения режима хранения детализированной статистической информации ПО каждому запросу. Обратите внимание на то, что дета-

лизированная информация может сохраняться или не сохраняться для всего сервера целиком, ее нельзя задавать на уровне виртуального узла.

Чтобы включить отслеживание детализированной статистической информации, необходимо воспользоваться директивой:

```
ExtendedStatus on
```

Б.34. Модуль mod_unique_id

Этот модуль генерирует для каждого запроса уникальный маркер и применим только на Unix-машинах. Он не имеет директив.

Б.35. Модуль mod_userdir

Этот модуль хранится в файле mod_userdir.c и компилируется по умолчанию. Детальное описание этого модуля можно найти в главе 5, "Хостинг нескольких Web-узлов".

Б.35.1. Директива UserDir

Синтаксис: UserDir *каталог/имя-файла*
Умолчание: UserDirpublic_html
Контекст: конфигурация сервера, виртуальный узел
Перекрытие: Options
Статус: Base
Модуль: mod_userdir
Совместимость: Apache 1.2 (частично), Apache 1.3

Эта директива определяет подкаталоги, находящиеся в корневом каталоге пользователя в файловой системе, в которую направляются запросы при запросе документов пользователя.

: Чтобы сервер Apache направлял запросы к документам пользователя в подкаталог htdocs, расположенный в корневом каталоге, необходимо задать следующую директиву:

```
UserDir htdocs
```

Б.36. Модуль mod_usertrack

Модуль mod_usertrack **(ранее известный как модуль** cookies) **обеспечивает работу с файлами** cookies.

Б.36.1. Директива CookieExpires

Синтаксис: CookieExpires *срок-действия*
Контекст: конфигурация сервера, виртуальный узел
Статус: Optional
Модуль: mod_use.rtrack
Совместимость: Apache 1.2 (частично), Apache 1.3

Этой директивой определяется срок хранения файлов cookie. Этотмодуль распознает следующие английские слова: *years, months, weeks, hours, minutes* и *seconds.* Если задается только число, то это подразумевает, что время задано в секундах.

Чтобы указать серверу Apache, что время хранения истекает через 1 неделю, 2 дня и 3 часа, необходимо задать следующие директивы:

```
CookieExpires "1 week 2 days 3 hours"
```

Б.36.2. Директива CookieName

Синтаксис: CookieName *символы*
Умолчание: Apache
Контекст: конфигурация сервера, виртуальный узел, каталог, файл .htaccess
Перекрытие: Optional
Статус: Base
Модуль: mod_usertrack

По умолчанию файл cookie, который этот модуль использует для отслеживания, называется Apache. Однако с помощью этой директивы его имя можно изменить.

- Чтобы изменить имя файла cookie на httpd, примените следующую директиву:

```
CookieName httpd
```

Б.36.3. Директива CookieTracking

Синтаксис: CookieTracking *on* \ *off*
Контекст: конфигурация сервера, виртуальный узел, каталог, файл .htaccess
Перекрытие: FileInfo
Статус: optional
Модуль: mod_usertrack

Эта директива используется для того, чтобы включить режим отслеживания cookie по серверному или каталожному принципу. Будучи активизированным, сервер Apache пошлет файлы cookie по всем новым запросам.

Пример

Чтобы включить отслеживание пользователей, необходимо прибегнуть к следующей директиве:

```
CookieTracking on
```

Приложение

B

КОНЦЕПЦИЯ ПРОТОКОЛА TCP/IP

В этом *приложении...*

B.1. Введение

Аббревиатура TCP/IP раскрывается, как Transport Control Protocol/Internet Protocol (протокол управления передачей/протокол Internet), что, вероятно, говорит вам не о многом. На компьютерном жаргоне *протоколом* называется опубликованный набор стандартов, который используется для определения порядка и природы обмена данными по сети. Идея, которая была заложена при создании протокола, заключалась в том, чтобы предоставить программистам правила, которые, если им точно следовать, позволяют программам, написанным совершенно независимо (например броузер Internet Explorer и Apache Web-сервер Apache), устанавливать сеанс связи, обмениваться информацией, а затем отключаться друг от друга.

Технически протокол TCP/IP является не единым протоколом, а скорее, комплексом отдельных протоколов. Часть протокола, которая обозначается TCP, занимается формированием отдельных блоков данных, которые называются пакетами, передает эти пакеты по сети, обеспечивая при этом их безопасную передачу. Часть IP отвечает за адресацию и маршрутизацию.

B.2. IP-адрес

IP-адрес представляет собой 32-битовое число, которое вместе с маской сети идентифицирует положение узла в сети. IP-адреса записываются в виде октетов, разделенных точками:

```
145.186.47.50
```

Следует отметить, что этот формат рассчитан только на удобство восприятия человеком. Компьютер работает непосредственно с 32-разрядным двоичным числом. Чтобы IP-адреса были читабельными, на границах байтов ставится точка, и значения байтов транслируются в десятичные числа, как это показано на рис. B.1.

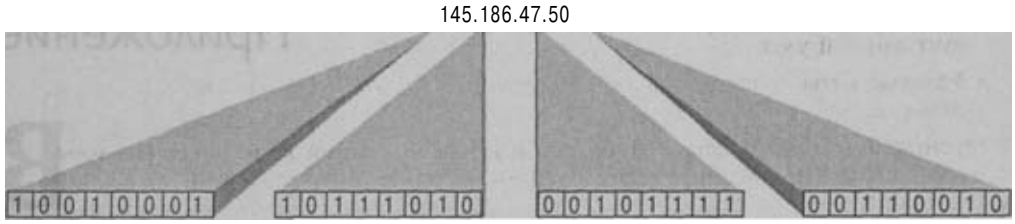

Рис. В. 1. Границы байтов

Плохо, что количество имеющихся в наличии IP-адресов ограничено 32-битовой длиной. Для сравнения можно заметить, что 3-значное число может принимать только 1000 возможных значений:

$10^3 = 1\ 000,$

где 10 — количество возможных значений, которые может принимать число, состоящее из одной цифры, а 3 — количество разрядов в числе. Аналогично количество возможных IP-адресов ограничено 2 (количество возможных значений числа, состоящего из одной цифры в двоичной системе) в степени 32 (количество бит в IP-адресе):

$2^{32} = 4\ 294\ 967\ 296$

Несмотря на то, что цифра 4294967296 выглядит достаточно внушительно, в действительности это не так уж и много. Плачевно, но факт, что побочным эффектом деления IP-адреса на классы (см. раздел "Классы" в этом приложении) является то, что в нашем распоряжении имеется значительно меньше, чем 4294967296 адресов. И очень вероятно, что скоро они будут исчерпаны. Существуют планы перевода стандарта IP-адресов с 32 на 128 бит, но в действительности, чтобы это сделать, необходимо поднять всемирный переполох, аналогичный тому, что был поднят по поводу проблемы 2000 года[1].

В.3. Маска сети

Вероятно, что наиболее сложным моментом обучения концепции TCP/IP является изучение концепции *маскирования сети.* Напомним, что разделение 32-битового адреса на четыре однобайтовых последовательности, является удобным для восприятия человеком. На практике граница может проходить в любом месте и совсем не обязательно по границе байтов.

[1] Так получилось, что как и любой дееспособный программист в мире, я посвятил определенное время работе над проблемой 2000 года. Как вы, вероятно, еще помните, эта проблема была создана близорукими программистами времен холодной войны, не пожелавших отводить соответствующего пространства в своих переменных для хранения четырех цифр, представляющих год. Чтобы близорукие программисты 90-х годов могли обрабатывать такие данные, применяя алгоритмы вроде:

```
if (2_digit_year >= 50) then
        {
        let century =19
        }
else
        {
        let century =20
        }
```

Конечно это будет отлично работать до 2049 года, начиная с которого любой такой устаревший код, в котором используются такие, с позволения сказать "решения", должен быть отловлен и перепрограммирован. Иногда закрадывается подозрение, что порочность человеческой натуры не позволит нам сделать этого.

- Сетевые биты являются частью IP-адреса, определяющей сеть, в которой существует данный узел.
- Узловые биты являются частью IP-адреса, определяющей номер узла в пределах подсети.

Место границы между сетевыми битами и узловыми битами задается сетевой маской.

Чтобы понять принципы маскирования сети, необходимо знать, каким образом октеты транслируются в соответствующие двоичные значения. Как видно из рис. В.2, в десятичной арифметике имеется знакоместо для единиц, десятков, сотен и т.д.

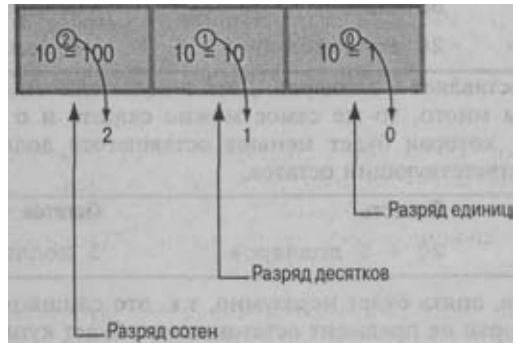

Рис. В.2. Десятичные степени

Аналогично, в двоичной арифметике, есть знакоместо для 1, 2, 4, 8, ит.д., что отражено на рис В.3.

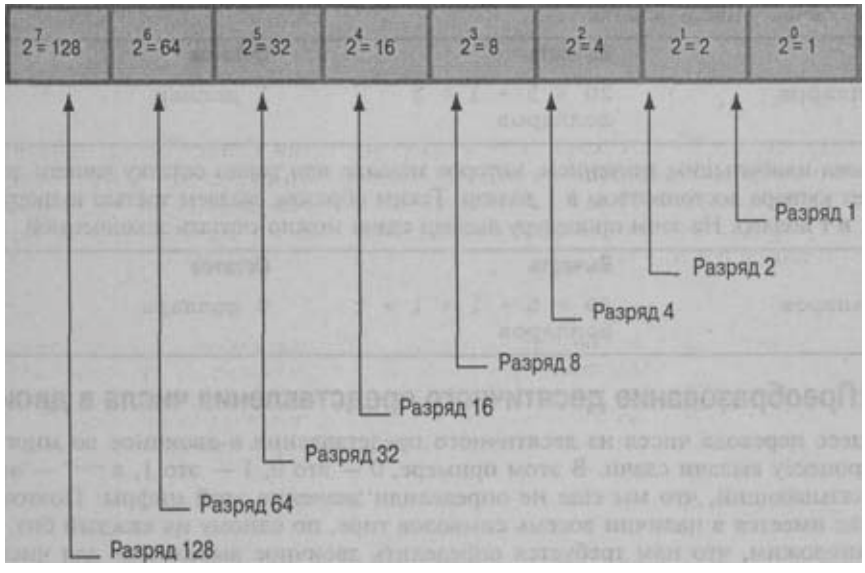

Рис. В.3. Двоичные степени

B.3.1. Пример

Если вы когда-нибудь работали кассиром, то должны знать, что способ подсчета сдачи заключается в выдаче купюр максимального достоинства (не превышающего причитающейся суммы сдачи).

Предположим, что мы должны вернуть покупателю 28 долларов сдачи. Очевидно, что вы не можете дать ему купюру достоинством в 100 или 50 долларов. Это будет значительно больше величины сдачи. Самая крупная купюра, которая не превышает долга — 20 долларов. Вот ее и можно отсчитать покупателю.

Сдача	*Вычесть*	*Остаток*
28 долларов	20 долларов	8 долларов

Теперь наш долг составляет 8 долларов. Дать покупателю еще 20 долларов нельзя, т.к. это будет слишком много, то же самое можно сказать и о купюре 10 долларов. Наибольшей купюрой, которая будет меньше оставшегося долга — это купюра в 5 долларов. Получим соответствующий остаток.

Сдача	***Вычесть***	***Остаток***
28 долларов	2 0 + 5 долларов	3 доллара

Дать еще 5 долларов, опять будет неразумно, т.к. это слишком много. Теперь наибольшей купюрой, которая не превысит остаток сдачи, будет купюра в 1 доллар.

Сдача	***Вычесть***	***Остаток***
28 долларов	2 0 + 5 + 1 долларов	2 доллара

Можно покупателю дать еще 1 доллар? Да, 1 доллар все еще меньше 2 долларов, которые сейчас у нас в остатке.

Сдача	***Вычесть***	***Остаток***
28 долларов	2 0 + 5 + 1 + 1 долларов	1 доллар

И снова наибольшим значением, которое меньше или равно остатку вашего долга все еще будет купюра достоинством в 1 доллар. Таким образом, выдаем третью купюру достоинством в 1 доллар. На этом процедуру выдачи сдачи можно считать законченной.

Сдача	***Вычесть***	***Остаток***
28 долларов	2 0 + 5 + 1 + 1 + 1 долларов	0 доллара

B.3.2. Преобразование десятичного представления числа в двоичное

Процесс перевода чисел из десятичного представления в двоичное во многом подобен процессу выдачи сдачи. В этом примере, 0 — это О, 1 — это 1, а "—" — это маркер, показывающий, что мы еще не определили значение этой цифры. Поэтому сначала у нас имеется в наличии восемь символов тире, по одному на каждый бит.

Предположим, что нам требуется определить двоичное выражение для числа 137. Посмотрев на рис. B.3, скажем, что самый левый бит в байте будет составлять 128. Потому что 128 меньше 137, этим числом можно воспользоваться. Зададим 1 в самом левом бите.

Это соответствует 128 из числа 137, преобразуемого в двоичный формат.

Десятичное число	*Вычесть*	*Остаток*
137.	128	9

Нельзя использовать 64, потому что это больше, чем 9, таким образом, поставим О на месте 64.

10------

Десятичное число	*Вычесть*	*Остаток*
1 3 7	1 2 8 + 0	, 9

Нельзя воспользоваться 32 и 16 (в соответствии с рис. В.3 — это следующие две позиции) так как они все еще больше, чем значение 9, которое у нас в остатке. Поэтому на месте 32 и 16 в этих местах поставим 0 для того, чтобы показать, что эти биты не используются.

1000----

Десятичное число	*Вычесть*	*Остаток*
137	128 + 0 + 0 + 0	9

Теперь настала очередь места 8, и действительно 8 меньше, чем 9, которая у нас по-прежнему в остатке. Таким образом, на месте 8 можно задать 1, для того, чтобы показать, что этот бит используется.

10001---

Десятичное число	*Вычесть*	*Остаток*
137	128 + 0 + 0 + 0 + 8	1

Оставшаяся часть решения очевидна, но мы пройдем процедуру получения решения до конца. Следующим разрядом, меньше, чем 8, будет 4. Так как 4 больше 1, то этот бит использоваться не будет. Ставим 0.

100010--

Десятичное число	*Вычесть*	*Остаток*
137	128 + 0 + 0 + 0 + 8 + 0	1

Следующим разрядом является разряд 2 и он также больше 1. И в этом разряде поставим 0.

1000100-

Десятичное число	*Вычесть*	*Остаток*
137	128 + 0 + 0 + 0 + 8 + 0 + 0	1

В конце концов мы пришли к логическому финалу нашей процедуры. Рассмотрим, наконец, бит 1. Значение 1 равно 1, которое у нас все еще в остатке. Итак, ставим в последнем разряде 1 и завершаем наши упражнения.

10001001

Десятичное число	*Вычесть*	*Остаток*
137	128 + 0 + 0 + 0 + 8 + 0 + 0 + 1	О

Теперь нам известно, что двоичным представлением десятичного числа 137 является 10001001. Аналогичный процесс может быть использован для определения двоичного представления любого десятичного числа меньше или равного 255, которое является самым большим 8-битовым числом. Для чисел, превышающих 255, рис. В.3 необходимо будет расширить.

В.3.3. Преобразование маски сети в двоичное представление

В соответствии с соглашением, IP-адреса и маски сети имеют представление, разделенное на 1-байтовые сегменты, и процедура, здесь описанная, будет сопровождать вас на протяжении всей вашей карьеры сетевого администратора. Так, например, маска сети

255.255.252.0

преобразуется в следующие 8-битовые сегменты:

11111111.11111111.11111100.00000000

Другая общая маска сети, 255. 255.255.0, преобразуется следующим образом:

11111111.11111111.11111111.00000000

В.3.4. Работа с масками сети

Концепция маски сети очень проста. Как было сказано ранее, маска сети задает границу между узловой и сетевой частями IP-адреса.

Например, пусть, как это показано на рис. **В.1,** мы имеем IP-адрес 145.186.47.50. Ему соответствует двоичное значение:

145.186.47.50 = 10010001.10111010.00101111.00110010

Пусть этот IP-адрес используется с маской сети 2551255.252.0, двоичное значение которой составляет:

255.255.252.0 = 11111111.11111111.11111100.00000000

Определить, какая часть вашего адреса представляет сеть, а какая — узел, можно, наложив IP-адрес поверх сетевой маски. Каждый бит IP-адреса, имеющий значение 1 в сетевой маске является сетевой частью IP-адреса. Каждый бит IP-адреса, которому соответствует значение 0 в сетевой маске, представляет собой узловую часть IP-адреса.

IP-адрес:	10010001.10111010.00101111.00110010
Маска сети:	11111111.11111111.11111100.00000000
Сетевая часть:	10010001.10111010.001011
Узловая часть:	11.00110010

В.3.5. Классы IP-адресов

IP-адреса делятся на классы на основании количества сетевых битов, которые они содержат. Это показано в табл. В.1.

Таблица В.1. Классы IP-адресов

Класс	*Значение первого байта*	*Сетевая маска*
A	< 128	255.0.0.0
B	от 128 до 191	255.255.0.0
C	от 192 до 223	255.255.255.0
Зарезервировано	>223	

B.4. IP-порты

В дополнение к 32-битовому IP-адресу протокол также обеспечивает дополнительное 16-битовое число, что и называется *портом*. Несмотря на свою относительную малоизвестность, IP-порт представляет собой достаточно важную часть информации и частично определяет IP связь[2].

В ОС Unix номера портов и соответствующие им сервисы задаются в файле /etc/services. Вот пример такого файла:

```
tftp          69/udp
gopher        70/tcp       # сервер gopher
rje           77/tcp
finger        79/tcp
http          80/tcp    #    www используется некоторыми ошибочными
www           80/tcp       # программами, http будет правильнее
link          87/tcp        ttylink
kerberos      88/udp       kdc # Kerberos udp
kerberos      88/tcp       kdc # Kerberos top
supdup        95/tcp      # BSD supdupd(8)
hostnames     101/tcp      hostname # обычно для sri-nic
iso-tsap      102/tcp
x400          103/tcp   #    ISO Mail
x400-snd      104/tcp
csnet-ns      105/tcp
pop-2         109/tcp      # PostOffice V.2
```

Обратите внимание на то, что порт 80 илипорт http объявлен явным образом. Если по какой-то причине определение порта 80 в вашем файле services было закомментировано, сервер Apache работать не будет. ОС Unix полностью полагается на список портов, которые необходимо прослушивать, приведенный в файле /etc/services, и будет игнорировать все порты, не перечисленные в этом списке. Соответствующий файл в ОС Windows тоже имеет имя services. Его можно найти в каталоге C:\Windows.

2 Технически IP-соединение создается двумя сокетами, одним локальным и другим удаленным. Каждый разъем состоит из комбинации значений IP-адрес/порт.

Приложение

Г

ПРЕОБРАЗОВАНИЕ ИМЕН В IP-АДРЕСА

Г.1. Введение

Преобразование имен — это процесс трансляции имен узлов в IP-адреса. Нужно отметить, что если IP-адреса читаются слева направо, связанные с ними имена доменов квалифицируются справа налево. В табл. Г.1 дан перечень наиболее часто используемых доменных имен высшего уровня.

Таблица Г.1. Домены высшего уровня

Домен	*Где используется*
com	Коммерческие организации.
edu	Учебные заведения.
gov	Правительственные организации.
mil	Военные организации (армия и флот).
net	Исторически этот домен был зарезервирован для организаций, представляющих сетевые услуги, но сейчас это имя присваивается любому, кто в состоянии заплатить регистрационный взнос за этот до мен.
org	Изначально зарезервированный для некоммерческих организаций домен . org сейчас может получить любой.
int	Международные организации.

Точный метод, который используется для преобразования адресов или имен доменов, зависит от конфигурации системы. Возможные источники информации включают:

- статические файлы, такие как /etc/hosts или C:\WINDOWS\hosts, на локальной машине,
- локальные информационные службы, такие как NIS,
- глобальную службу аутентификации DNS (Domain Name Service).

Г.1.1. Файл /etc/hosts

Файл /etc/hosts представляет собой статический текстовый файл, который используется для ассоциации IP-адресов с именами узлов. Строка в нем начинается с IP-адреса и одного или более узлов. Узел, имя которого указывается слева, считается официальным именем узла, все остальные считаются псевдонимами. Файл /etc/hosts используется на всех Unix-машинах и многих Windows-машинах и имеет следующий вид:

```
192.168.100.1          odin.example.com
192.168.100.80         fenris.asgard.com fenris
192.168.100.10         loki.asgard.com loki
```

Г.1.2. Утилита nslookup

Утилита nsiookup является стандартной утилитой ОС Unix для присвоения именам узлов IP-адресов. Кроме того, по ним можно узнать какие ресурсы будут опрошены для определения IP-адреса при выполнении запроса. Например, команда

```
nsiookupwww.apache.org
```

вернет что-то вроде:

```
Server: ns2.mindspring.com
Address: 207.69.188.186
Non-authoritative answer:
Name: www.apache.org
Address: 63.211.145.10
```

В наши дни большая часть утилит nsiookup берет информацию, необходимую для преобразования имен в IP-адреса (хост-файл, DNS сервер), из файла /etc/nsswitch.conf. В этом файле также определяется порядок, по которому эти ресурсы опрашиваются:

```
# /etc/nsswitch.conf
#
# Конфигурационный файл переключения имен
#
passwd: compat
shadow: compat
group: compat
hosts: files nis dns
networks: nis files dns
ethers: nis files
protocols: nis files
rpc: nis files
services: nis files
```

Чтобы предоставить доступ к вашему серверу всем желающим, вам необходимо зарегистрироваться на DNS. За определенную плату это можно сделать на узле *http://www.networksolutions.com*.

Приложение

A

РЕШЕНИЕ ПРОБЛЕМ, ВОЗНИКАЮЩИХ ПРИ РАБОТЕ СЕТИ

В этом приложении...

Д.1. Введение

Перед тем как начать разговоры о настройках сетевых карт и маршрутизации, хотелось бы расставить все точки над i. Во-первых, концепции и методы, затрагиваемые в этой главе, не являются характерными только для сервера Apache и поэтому они не совсем имеют отношение к данной книге. Они включены в это приложение потому, что опыт подсказывает, когда пользователи начинают выяснять, почему Web-сервер отказывается работать, около пятидесяти процентов времени тратится на решение сетевых проблем, которые никоим образом не связаны с самим сервером Apache.

Во-вторых, это достаточно поверхностное освещение сетевых методов и концепций. Чтобы полностью осветить проблему, нам потребуется три тома убористого текста. Вместо этого, все, что я хочу — это вкратце обозначить проблемы, которые являются причиной восьмидесяти или девяноста процентов сетевых сбоев во всем мире.

В-третьих, команды, приведенные в примерах, не обязательно будут работать на вашей системе. Чем ближе вы будете приближаться к аппаратному уровню (сетевые карты уже находятся довольно близко к этому уровню), тем больше будет наблюдаться вариаций в различных клонах ОС Unix. Насколько мне известно, все показанные здесь команды работают во всех клонах ОС Unix. С другой стороны, мне известно, что результаты работы, используемые опции и синтаксис в определенной степени различаются. Если конкретная команда в соответствии с тем, что показано в примерах не будет работать, это не беда. Всегда можно получить справку о конкретной команде с помощью общеизвестной утилиты man, а она подводит редко.

Д.2. Сбои соединения на физическом уровне

Общеизвестная доктрина гласит: пятьдесят процентов всех проблем, возникающих с сетью, имеют мало общего с конфигурацией программного обеспечения. Лично мне кажется, что это немного завышенная оценка, а вот цифра в двадцать пять процентов, вполне соответствует действительности. Вот несколько общих проблем, с которыми можно столкнуться.

1. Потеря соединения. Эта неисправность не так характерна для неэкранированной витой пары, но на старых конфигурациях с коаксиальным кабелем она случалась сплошь и рядом. Первым и самым простым решением при возникновении такой проблемы является проверка надежности соединений.
2. Плохой кабель. Кабель — это наиболее подверженный повреждениям элемент вашей сети. В обычной практике кабели подвергаются ударам, надломам, обрывам и даже надкусам.
3. Плохой порт. Если соединение и кабель у вас сомнений не вызывают, попробуйте подключиться к другому порту.
4. Не работает сетевая карта. Когда все остальное не дало результатов, замените сетевую карту. Недостатком этого метода является то, что если модель новой сетевой карты не идентична модели старой, то такая замена *может повлечь за собой* новые проблемы. И то правда, жизнь всегда преподносит сложные решения.

При отсутствии кабельного тестера решением проблемы может служить наличие проверенного хорошего кабеля.

Значительно дешевле сделать свой собственный кабель, чем покупать готовый. Хороший монтажный инструмент в любом компьютерном магазине или магазине электроники стоит около 40 долларов. Еще за 50 долларов можно купить 200 метров кабеля и полную коробку "концевиков". Для сравнения — 30 метров готового кабеля стоит 79,99 долларов.

Д.3. Диагностика программного обеспечения ОС Unix/Linux

Если у вас есть полная уверенность в том, что сетевое оборудование находится в исправности, но сеть по-прежнему не работает, остается проверить работу программного обеспечения. Успешные соединения Web-сервера зависят от двух компонентов:

- простота сетевых соединений,
- настройка сервисов (включая и httpd)

Сетевые соединения включают такие моменты, как IP-адрес и маску сети, настройку карты Ethernet и преобразование имени в IP-адрес. Естественно, что настройки сервисов делаются индивидуально и всецело зависят от задач, решаемых вашей системой.

Таблица Д.1. Сетевые команды

Что *требуется сделать*	*Как это сделать*
Определить название сетевой карты.	`netstat -i`
Определить стандартный путь.	`netstat -r`
Протестировать соединение между двумя узлами.	`ping`

Окончание табл. Д.1

Что требуется сделать	***Как это сделать***
Просмотреть или модифицировать установки интерфейса.	ifconfig
Добавить или удалить информацию о маршрутизации,	route
Задать метод преобразования имени в IP-адрес.	nslookup
Найти определенный узел или IP-адрес.	nslookup

Первое, что необходимо проверить, это связь. Предположим, мы пытаемся получить доступ к Web-серверу, находящемуся на узле loki. Для проверки наличия связи воспользуйтесь следующей командой:

```
ping loki
```

Команда ping не делает ничего, кроме того, что отсылает заданный запрос на заданный узел (или шлюз) и ожидает пакета, подтверждающего успешный прием. При благоприятном исходе будет получен следующий результат:

```
PING loki (192.251.100.10): 56 data bytes
64 bytes from 192.251.100.10: icmp_seq=0 ttl=32 time=0.8 ms
64 bytes from 192.251.100.10: icmp_seq=l ttl=32 time=0.7 ms
——loki ping statistics ——
2 packets transmitted, 2 packets received, 0% packet loss
round-trip min/avg/max = 0.7/0.7/0.8 ms
```

Если такая проверка положительных результатов не дает, можно попробовать другие варианты. Обратите внимание, что в этом примере вы пытались подключиться к узлу, указав его имя. Этот метод предполагает, что ваш компьютер знает, каким образом имя компьютера (в нашем случае loki) преобразуется в числовой IP-адрес. Это не обязательно соответствует реальному положению вещей[1] и может являться причиной возникшей проблемы.

Для проверки связи с узлом независимо от преобразования имен, можно указать непосредственно IP-адрес узла.

```
ping 192.251.100.10
```

Когда "пингование" по IP-адресу дает результат, а по доменному имени — нет, проблема (возможно частично) заключается в механизме преобразования имени. Быстрым и простым решением этой проблемы будет добавление имени и IP-адреса в файл /etc/hosts, расположенный на локальном компьютере[2]. Если служба DNS вами не используется, то этого будет вполне достаточно. В противном случае, необходимо добавить информацию о машине в базу данных DNS и/или восстановить соединение с DNS-сервером.

[1] Процедура преобразования цифровых IP-адресов в символические имена узлов и обратно и называется разрешением имен. Обычно это осуществляется с помощью статических хост-файлов (обычно на машинах Unix/Linux это файл /etc/hosts) или с помощью сервиса DNS.

[2] Хост-файлы могут использоваться и на Windows-машинах. Для этого в корневом каталоге, а это C:\WINDOWS для ОС Windows или C:\WINNT для ОС Windows NT, создается файл hosts.

Д.4. Моменты, на которые необходимо обратить внимание при ошибках "пингования"

Соединение вашего компьютера с IP-сетью определяется в первую очередь двумя моментами: IP-адресом и маской сети. Когда одно из этих значений является неправильным на все 100%, IP-соединение работать не будет.

Сначала с помощью команды netstat определите, с помощью какого интерфейса локальной сети вы пытаетесь соединиться. Как видно из имени утилиты netstat, что она предназначена для определения состояния соединений в сети. Возможности утилиты netstat в различных системах Unix различны, но не существует ни одного варианта утилиты, которая не поддерживала бы опцию -r. Эта опция предназначена для получения информации о маршрутизации. Введите следующую команду:

```
netstat -r
```

Эта команда возвращает следующую информацию:

```
Kernel IP routing table
Destination     Gateway     Genmask         Flags   MSS Window  irtt Iface
192.251.100.0   *           255.255.255.0   U       1500 0      0    eth0
135.187.35.0    ^           255.255.252.0   U       1600 0      0    ethl
127.0.0.0       *           255.0.0.0       U       3584 0      0    lo
default         odin        0.0.0.0         UG      1500 0      0    eth0
```

В самом левом столбце, обозначенном как Destination, находитсяадрес, по которому делались попытки послать пакеты. Продолжая пример, начатый в предыдущем разделе, обратим внимание на адрес 192 . 251.100 . 0. Здесь значение адреса не совпадает с IP-адресом целевой машины (сравним IP-адрес 192. 251.100.70 машины loki со значением адреса 192.251.100.0 определяющего подсеть loki, где значение О можно рассматривать как своеобразный групповой символ).

В любом случае, если вами определена подсеть, обратите внимание на столбец Iface для того, чтобы определить какой интерфейс используется. В вышеприведенном примере это будет ethO. Зная интерфейс, с помощью команды ifconfig можно определить свойства сети.

```
ifconfig eth0
```

Варианты команды ifconfig также широко варьируются, но они всегда содержат информацию об IP-адресе, маске сети и является ли эта сетевая карта активной (UP). Вот пример для ОС Red Hat Linux5.2.

```
ethO      Link encap:Ethernet  HWaddr 00:20:78:17:9A:EB
          inet addr:192.251.100.1 Beast:192.251.100.255
Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST MTU.-1500 Metric:!
          RX packets:90 errors:0 dropped:0 overruns:0 frame:0
          TX packets:1031 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0
          Interrupt:11  Base address:0xef80
```

Обратите внимани е на слово "UP", включенное в вывод результата. Это свидетельствует о том, что система не только знает о существовании сетевой карты, но и то, что эта карта ожидает ввода. Если полученный результат не содержит слова "UP", необходимо попытаться перевести его в состояние "UP". Это также входит в функции команды ifconfig.

```
ifconfig eth0 up
```

Существует ошеломляющее множество причин, которые могут воспрепятствовать активизации Ethernet-карты. Это может быть неподходящий драйвер используемой сетевой карты (это характерно для ОС Linux; в этом случае можно посетить Web-узел производителя и найти на ней соответствующий драйвер). Сам драйвер может работать со сбоями (поищите на Web-узле дополнения или усовершенствованные драйверы). Используемый IP-адрес или маска сети нарушает определенный пункт соответствующего RFC[3] (проверьте карту с помощью проверенного IP-адреса и маски сети). Сетевая карта сама по себе может быть неисправна (в этом случае замените ее).

Команда ifconfig показывает, что карта находится в состоянии "DP", попробуйте "пропинговать" свой собственный IP-адрес. Если сетевая карта "пингуется", но компьютер по-прежнему не в состоянии подключиться к внешнему миру, в этом случае проблема кроется скорее всего в отсутствии физического соединения.

Д.4,1. Переконфигурирование сетевой карты

Команда ifconfig выполняет и другие функции. Ею можно использоваться для настройки сетевой карты. Имейте в виду, что возможности команды ifconfig в различных версиях ОС Unix немного различаются, таким образом, перед тем, как ею воспользоваться, необходимо тщательно проверить ее возможности в системном руководстве или справочнике man. В качестве примера приведем команду, которая в HP-UX 11.0 установит IP-адрес карты равным 192.251.100. 32 *с* маской сети 255.255.255.0:

```
ifconfig lan1 192.251.100.32 netmask 255.255.255.0
```

Д.5. Что делать в случае успешного "пингования"

Успешное "пингование" свидетельствует о том, что между двумя машинами установлена связь. Это означает, что почти наверняка все проблемы, возникающие в этом случае, вызваны конфигурацией сервисов. Для диагностирования и исправления ошибок, возникающих в этом случае, могут пригодиться следующие методы.

Д.5.1. Проверка работы сервиса

Чтобы проверить работает сервис или нет (в нашем случае это httpd), можно воспользоваться командой ps. В данном примере опция -е генерирует расширенный листинг данных, а опция -f генерирует полный листинг всех работающих в системе процессов. Аналогичные функции в ОС Linux выполняет команда ps aux. Часть командной строки | grep httpd перенаправляет весь вывод на команду grep (global regular expression parser — анализатор регулярных выражений), задачей которой является поиск в больших текстовых цепочках определенных последовательностей символов, в данном случае httpd.

```
ps -ef I grep httpd # ОС HP-UX
ps aux i grep httpd # ОС Linux
```

Если эта команда не дает никакого результата, необходимо перезапустить процесс httpd.

[3] Стандарт RFC, аббревиатура которого обозначает "Request For Comment" ("Запрос на комментарии"), СТОЛ стандартом де-факто В самых разных отраслях компьютерной и электроктронной инженерии, включая и стандарт протоколов Ethernet (RFC ***) и набор TCP/IP (RFC ***).

Д.5.2. Прослушивается ли порт сервером?

Unix-машины должны иметь информацию о том, какие порты необходимо прослушивать. По умолчанию большинство систем конфигурируются на прослушивание порта 80[4] на предмет поступления http-запросов. Однако это создает пробел в системе защиты, таким образом, некоторые системные администраторы просто удаляют эту строку из файла /etc/services. Чтобы проверить наличие этой проблемы, запустите следующую команду:

```
cat /etc/service | grep 80
```

Результатом должно быть что-то вроде:

```
http 80/tcp
```

Отрицательным результатом будет пробел. Это будет означать, что компьютер не прослушивает порт 80 на предмет http-соединений. Добавьте в файл services строку "http 80/tcp". Присутствие символа "#" в начале строки тоже нежелательно, т.к. это будет означать, что эта строка закомментирована. Решить эту проблему можно с помощью привычного текстового редактора для того, чтобы удалить символ комментария в начале строки.

Д.5.3. Проверка работы под управлением пользователя root

При решении проблем, возникающих при работе сервиса httpd, это может не пригодиться, но может пригодиться во многих других случаях, таким образом, этот метод нужно упомянуть для полноты картины изложения. Попытайтесь получить доступ к сервису, зарегистрировавшись в качестве пользователя root. Если в этом режиме все работает, но не работает у других пользователей, проблема заключается в правах пользователя. Посмотрите права пользователя с помощью команды Is -l и при необходимости измените их с помощью команды chmod.

[4] Другие службы прослушивают другие порты. Например, протокол SMTP работает с портом 25, протокол telnet прослушивает порт 23. В этих номерах нет ничего особенного, теоретически можно назначить сервису httpd порт 1234. Пока броузеры, которые подключаются, будут знать об этом, все будет хорошо. Однако любой другой броузер, пытающийся подключиться к вашему серверу извне, будет искать стандартный порт.

Приложение

E

КОНЦЕПЦИЯ UNIX

В этом приложении...

E.1. Конфигурационные файлы

В этом разделе будет перечислено несколько стандартных конфигурационных файлов ОС Unix, которые будут необходимы вам на протяжении всего вашего пребывания в должности администратора сервера Apache. Этот перечень ни в коей мере нельзя рассматривать как окончательный. Если то, что вы ищете в этом приложении, отсутствует, можно обратиться к такой популярной справочной утилите ОС Unix, каковой является утилита man. Например, команда

```
man sendmail
```

отобразит информацию о конфигурации и работе с процессом sendmail.

/etc/hosts. Это текстовый файл, который является базой данных, связывающей **IP**-адреса с именами узлов и псевдонимами.

/etc/services. Этот файл связывает имена сервисов с номерами портов. Имя сервиса, связанное с пронумерованным портом, используется процессом `inetd` (см. раздел в этом приложении, посвященный процессам) для того, чтобы определить, какой порт необходимо просматривать на предмет входящего запроса.

/etc/passwd. Этот файл является базой данных о пользователях системы. Записи сделаны в последовательном виде. Их смысловая нагрузка показана в табл. ЕЛ.

```
name:encrypted_password:UID:GID:user    name:home_directory:shell
```

Таблица E.1. Записи базы данных о пользователях системы

Запись	*Описание*
name	Имя пользователя, которое печатается при регистрации.
UID	Цифровой идентификатор пользователя, связанный с именем пользователя. Строго говоря, и ID является единственным идентификатором пользователя в системе; имя целиком и полностью предназначается для удобства администратора системы.

Окончание табл. Е. 1

Запись	*Описание*
GID	Аналогично UID, но определяет группу, к которой принадлежит данный пользователь.
user name	Имя пользователя на английском языке, например Scott Hawkins. При необходимости его можно опустить. За исключением команды finger оно нигде не используется.
/home/directory	Абсолютный путь к корневому каталогу пользователя.
shell	Оболочка (ksh, bash и т.д.), которая загружается при запуске системы.

/etc/group. Это файл, в котором система хранит объявление групп пользователей. Файл содержит алфавитно-цифровое имя группы, цифровой идентификатор группы, список алфавитно-цифровых имен пользователей, являющихся членами групп.

Е.2. Процессы

Процесс — это программа, работающая в данный текущий момент. Процесс содержит исполняемые команды, данные программы, данные стека. Процессы уникально идентифицируются своими номерами, которые называются идентификаторами процесса или pid.

Процессы создаются из других процессов с помощью системного вызова fork(). Например, напечатав ls и щелкнув по клавише <enter>, пользователь инициализирует процедуру разветвления процесса, который является оболочкой, создавая при этом почти точную копию самого себя (все тождественно, за исключением идентификатора процесса pid). Затем копия немедленно передает управление своими системными ресурсами программе ls, запуская системный вызов exec(). Таким образом, все процессы происходят из процесса init.

Процесс, который был создан с помощью вызова fork() , является *порожденным процессом,* разветвленный процесс является родителем. В некоторых информационных распечатках можно увидеть поле PPID; это поле, в котором содержится идентификационный номер процесса-родителя.

Процессы могут взаимодействовать друг с другом посредством *сигналов,* являющихся стандартизированными цифровыми сообщениями о системных событиях (чтобы получить детальную информацию см. /usr/include/linux/signal.h).

Процессы имеют определенный *приоритет* — цифровое значение от -19 (самый высокий приоритет) до 20 (самый низкий приоритет), которое используется для выяснения того, какой процесс займет следующий интервал центрального процессора. По умолчанию процессы создаются с приоритетом 0.

Процессы могут работать как в *приоритетном режиме* (визуально выполняясь на экране), так и в *фоновом режиме* (работая без участия терминала). Можно перемещать процессы явным образом между приоритетным и фоновым режимом (см. в табл. Е.2 строки, соответствующие &, ^Z, bg, fg и jobs).

Процессы-зомби — это процессы, которые были остановлены и освободили свои системные ресурсы, но все еще не удалены из списка работающих процессов.

Некоторые команды, позволяющие непосредственно управлять такими процессами, приведены в табл. Е.2.

Таблица Е.2. Команды управления процессами

Команда	*Описание*
&	Запустить указанный процесс в фоновом режиме.
^Z	Остановить текущий процесс, работающий в приоритетном режиме.
bg	Перевести текущий процесс *в* фоновый режим.
fg	Перевести текущий процесс в приоритетный режим.
fuser	Отобразить pid любого процесса, использующего указанный файл.
jobs	Перечень фоновых процессов.
jail	Остановить выполнение процесса.
killall	Остановить все процессы, работающие с указанной командой.
nice	Изменить приоритет процесса.
nohup	Запустить процесс, который будет игнорировать все сигналы.
pidof	Определить и распечатать pid указанной программы.
ps	Получить информацию о состоянии процесса.
pstree	Отобразить дерево работающих процессов.
renice	Изменить идентификатор работающего процесса.
top	Отобразить информацию о состоянии центрального процессора.

Как использовать эти процессы, можно узнать из документации по вашей системе.

Е.3. Демоны

Демоны — это невидимые рабочие лошадки мира Linux. Демон — программа, работающая в фоновом режиме (т.е. она не подключена к конкретному терминалу), которая выполняет определенную системную задачу. Обычно это системная задача, которая призвана обслужить запрос некой пользовательской программы на выборку информации (например rwhod), для доступа к определенному системному ресурсу (например lpd) или для того, чтобы установить связь между двумя системами или частями одной и той же системы (например httpd, telnetd, talkd).

Демоны запускаются без вмешательства оператора. Это может произойти одним из трех способов:

- запуск с помощью сценария rc во время загрузки системы,
- запуск с помощью процесса init во время загрузки системы,
- запуск с помощью процесса inetd по мере необходимости (иногда с помощью процесса tcpd).

Следует заметить, что процесс inetd берет информацию о том, какие порты следует прослушивать из файла /etc/services.

Порт Internet является логическим сетевым соединением, связанным с определенным сервисом. Заметим, что этот порт разительно отличается от физических портов, к которым можно подключить физический кабель. Порт Internet появился тогда, когда

сетевые программисты мира собрались вместе и решили, что порт номер такой-то будет привязан к такому-то сервису. В номерах портов нет ничего особенного, просто существует договоренность об определенном порядке назначения этих номеров. Файл /etc/services содержит соответствия порт/сервис для вашей машины.[1]

Главной задачей конфигурирования демонов является настройка условий, при которых они запускаются и останавливаются без вмешательства оператора. Для этого была создана программа inetd (которая сама по себе тоже является демоном). Программа inetd прослушивает определенные порты на наличие запросов и по мере необходимости вызывает программы (тоже демоны), связанные с этими портами. Для того, чтобы демон управлялся процессом inetd, нужно добавить в файл /etc/inetd.conf строку в следующем формате (формат приведен в табл. Е.3):

```
service socket_type protocol wait user program arguments
```

Таблица Е.3. Как задать управление демоном процессом inetd

Запись	*Описание*
service	**Сетевой сервис объявляется в файле** /etc/services.
socket_type	**Одно из значений**(stream|dgram|raw|rdm|seqpacket).
protocol	Действующий протокол (например tcp, udp) в соответствии с заданным в файле /etc/protocols.
wait	Ожидание или немедленно.
user	Имя пользователя, под управлением которого работает данный сервис.
program	Программа, которая запускается на выполнение при получении процессом inetd запроса на соответствующий сокет.
arguments	Аргументы запускаемой программы.

Из соображений производительности, вероятно, не следует запускать программу httpd под управлением inetd. Демон inetd начинает дублироваться по мере поступления запросов на порт 80 (или 443, или любой другой в соответствии с конкретной настройкой системы). Это означает, что всякий раз, когда делается попытка Web-броузером подключиться к серверу, с диска в память копируется и запускается новая копия процесса httpd. Совершенно очевидная перегрузка, которая является следствием этого процесса, может нанести довольно серьезный урон производительности сервера.

С другой стороны, когда сервер Apache работает в автономном режиме, в памяти простаивает определенное количество порожденных процессов, ожидая следующего запроса. Такая практика уменьшает время задержек, которое является следствием запуска экземпляров httpd по мере необходимости. Частности этой проблемы можно найти в обсуждении директив MaxSpareServers и MinSpareServers.

[1] Следует обратить внимание, что длятого, чтобы иметь какой-то смысл все перечисленные в этом файле имена пользователей должны соответствовать некоторой записи в файле /etc/passwd.

Приложение

Ж

КОНЦЕПЦИЯ WINDOWS NT

Ж.1. Введение

Парадигма управления ОС Windows NT почти всецело базируется на GUI-интерфейсе. Место аргументов командных строк заняли радиокнопки, место конфигурационных файлов заняли закладки. Вероятно, это может немного озадачить тех, кто привык работать в среде ОС Unix, но после того, как станет понятно, как все работает, с таким положением дел можно вполне смириться.

Повествование этой главы потребует от вас умения переключения из окна в окно. Поэтому, когда вы встретите предписание в виде

Start=>Settings=>Control Panel,

это будет означать, что сначала необходимо щелкнуть по кнопке Start, затем в появившемся меню выбрать элемент Settings и войти в окно Control Panel.

Вот несколько подсказок, которые могут оказаться весьма полезными профессионалам Unix, начинающим свое знакомство с NT:

- Пользователь Windows NT administrator по сути аналогичен пользователю Unix root. Чтобы делать практически все в NT, необходимо получить привилегии администратора. При этом есть возможность определения привилегий с помощью утилиты User Manager for Domains. Эту утилиту можно найти, выполнив последовательность действий:

  Start=>Programs=>Administrative Tools (common)

- В отличие от обычных ОС Windows, после каждого небольшого изменения в конфигурации производить перезагрузку необязательно.
- Большая часть аппаратного оборудования и свойств настроек управляется из окна Control Panel. Для администраторов Apache наибольший интерес представляет имеющийся здесь элемент Network.
- В среде Windows ресурсы иногда определяются в соответствии с универсальным соглашением по присвоению имен. Вот синтаксис:

  \\computername\shared_resource\sudirl\...\subdirN\filename

  Вот, например, имя, которое вполне удовлетворяет этому соглашению:

  \\Loki\hplj5

Ж.1.1. Работа с сетью

ОС Windows NT поисторическим и маркетинговым причинам имеет в своем арсенале самые разнообразные сетевые протоколы. Вам пригодится только один из них — это TCP/IP. Свойства этого протокола доступны с помощью следующей последовательности команд:

Start=>Settings=>Control Panel=>Network

После этого, выбрав закладку Protocols, выделите протокол TCP/IP и щелкните по клавише Properties. В результате появится окно, аналогичное тому, что изображено на рис. Ж.1.

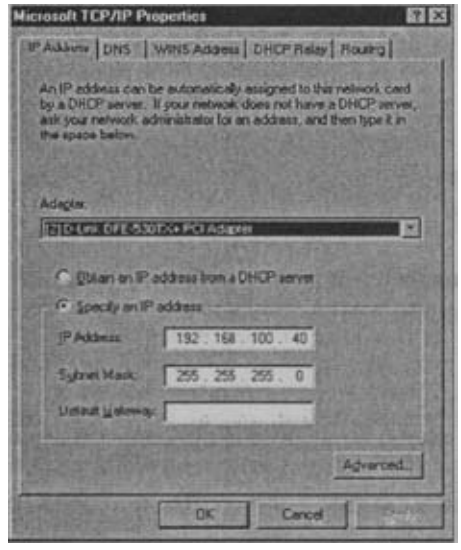

Рис. Ж.1. TCP/IP в ОС Windows NT

Теперь можно задавать и изменять свойства TCP/IP и DNS

Приложение

3

КОДЫ СОСТОЯНИЯ HTTP

В этом приложении...

3.1. Введение

В этом приложении приводится перечень кодов состояния HTTP и связанных с ними символических констант и дано краткое описание каждого из них:

- 1xx: Информационные. Запрос получен, процесспродолжается.
- 2xx: Успешно. Сигнал был успешно принят, понят и принят кобработке.
- 3xx: Перемаршрутизация. Необходимо предпринять определенные действия, чтобы выполнить запрос.
- 4xx: Ошибка клиента. Синтаксическая ошибка в запросе или запрос не может быть обработан.
- 5xx: Ошибка сервера. Сервер не смог выполнить явно безошибочный запрос.

3.2.1xx: Информационные коды

3.2.1. Код 100 Continue

Клиент может продолжать запрос.

3.2.2. Код 101 Switching Protocols

Сервер принял запрос клиента на переключение На модифицированныйпротокол.

3.3. 2xx: Успешное завершение

3.3.1. Код 200 ок: HTTP_OK

Успешный запрос.

3.3.2. КОД 201 Created: HTTP_CREATED

Запрос выполнен, в результате этого был создан новый запрос. (Типичная реакция на запрос метода PUT.)

3.3.3. КОД 202 Accepted: HTTP_ACCEPTED

Запрос был принят на обработку, но обработка не завершена.

3.3.4. Код 203 Non-Authoritative Information: HTTP_NON_AUTHORITATIVE

Возвращенная информация была собрана с копии третьей стороны.

3.3.5. КОД 204 No Content: HTTP_NO_CONTENT

Сервер обработал запрос, но в результате данные не получены.

3.3.6. Код 205 Reset Content

Пользовательский агент переустановит отображение документа, который вызвал посылку запроса.

3.3.7. Код 206 Partial Content

Сервер выполнил частичный запрос GET кдокументу.

3.4. 3xx: Перемаршрутизация

3.4.1. Код 300 Multiple Choices: HTTP_MULTIPLE_CHOICES

Этот заголовок используется для того, чтобы показать, что удовлетворять запросу может более чем один документ.

3.4.2. Код 301 Moved Permanently: HTTP_MOVED_PERMANENTLY

Запрошенный документ был перенесен на новый URI.

3.4.3. Код 302 Found: HTTP_FOUND

Запрошенный ресурс был временно перемещен на новый URI.

3.4.4. КОД 303 See Other: HTTP_SEE_OTHER

Ответ на запрос можно найти под различными URI. Он может быть выбран с помощью запроса, сделанного методом GET к этому ресурсу.

3.4.5. КОД 304 Not Modified: HTTP_NOT_MODIFIED

Сервер отвечает Этим кодом, когда клиент выполнил условный запрос GET и запрос был разрешен, но документ не модифицирован.

3.4.6. КОД 305 Use Proxy: HTTP_USE_PROXY

Доступ к запрошенному ресурсу *должен* производиться через proxy, заданный в поле Location. Поле Location задает URI для proxy.

3.4.7. Код 307 Temporary Redirect: HTTP_TEMPORARY_REDIRECT

Запрошенный ресурс временно находится под другими URI. Так как переадресация может быть отменена в любой удобный момент, для будущих запросов клиент должен использовать Request-URI.

3.5. 4xx: Ошибки на стороне клиента

3.5.1. Код 400 Bad Request

Запрос не понят сервером из-за наличия синтаксической ошибки.

3.5.2. Код 401 Unauthorized

Запрос требует идентификации пользователя.

3.5.3. Код 402 Payment Required

В данный момент не определен полностью, но будущее покажет.

3.5.4. Код 403 Forbidden

Сервер понял запрос, но он отказывается его выполнять. Идентификация тут не помогает.

3.5.5. Код 404 Not Found

Сервер не нашел соответствия по запросу Request-URI.

3.5.6. Код 405 Method Not Allowed

Метод, указанный в Request-Line, не соответствует ресурсу, заданному Request-URI.

3.5.7. Код 406 Not Acceptable

Ресурс, определенный запросом, может генерировать только ответ, характеристики которого не соответствуют заголовкам, посланным в запросе.

3.5.8. Код 407 Proxy Authentication Required

Этот код подобен коду 401 (Unauthorized), но в этомслучае клиент должен сначала идентифицировать себя с помощью proxy.

3.5.9. Код 408 Request Time-out

На протяжении периода ожидания сервера клиент не сделал запроса.

3.5.10. Код 409 Conflict

Запрос не будет завершен вследствие конфликта с текущим состоянием ресурса.

3.5.11. Код 410 Gone

Запрошенный ресурс и адрес, по которому можно сделать пересылку, на сервере отсутствуют.

3.5.12. Код 411 Length Required

Сервер отказывается принимать запрос без определенного Content-Length.

3.5.13. Код 412 Precondition Failed

При проверке на сервере одного или более полей заголовка запроса обнаружено несоответствие.

3.5.14. Код 413 Request Entity Too Large

Сервер отказывается обрабатывать запрос потому, что размер запроса больше того, что может обработать сервер.

3.5.15. Код 414 Request-URI Too Large

Сервер отказывается обрабатывать запрос потому, что Request-URI превышает размеры, которые может обработать сервер.

3.6. 5xx: Ошибки на стороне сервера

3.6.1. Код 500 Internal Server Error

Сервер начал сбоить по неизвестной причине.

3.6.2. Код 501 Not Implemented

Сервер не поддерживает возможностей, необходимых для обработки запроса.

3.6.3. Код 502 Bad Gateway

Сервер, функционирующий как шлюз или proxy, получил ошибочный ответ от подчиненного сервера, к которому он попытался получить доступ для обработки запроса.

3.6.4. Код 503 Service Unavailable

В данный момент сервер не в состоянии обработать запрос из-за того, что сервер перегружен или находится на профилактическом обслуживании.

3.6.5. Код 504 Gateway Time-out

Работая в режиме шлюза или proxy, сервер не получил вовремя ответ от сервера верхнего уровня.

3.6.6. КОД 505 HTTP Version not supported

Сервер не поддерживает или отказывается поддерживать версию протокола HTTP, которая была использована в последнем запросе.

Приложение

И

РЕГУЛЯРНЫЕ ВЫРАЖЕНИЯ

И.1. Введение

Формальное определение регулярных выражений звучит очень высоконаучно, но я обязательно включу его в качестве грозного предостережения тем, кто рассматривает возможность обучения на компьютерных курсах.

Пусть у нас есть множество символов (назовем это множество "алфавитом", но это не обязательно символы от "a" до "z"), которое обозначим "I". Множество, не имеющее членов (пустой набор) отображается символом "0" Символ "и" обозначает объединение двух множеств (например, объединение множеств {1,2,3} и {2,3,4} представляет собой множество {1,2,3,4}). Символ звездочка "*" обозначает нуль или более повторений предыдущего символа. Наконец, символы "a" и "Ь" используются в качестве переменных.

Регулярными выражениями над алфавитом "£" являются строки, содержащие следующие элементы алфавита:

1. Регулярным выражением является множество "0" и любой член множества "2".
2. Если "a" и "Ь" являются регулярными выражениями, то таковым является и выражение "ab".
3. Если "a" и "Ь" являются регулярными выражениями, то таковым является выражение "aub".
4. Если "a" является регулярным выражением, то таковым является выражение "a*".
5. Любое выражение, которое не удовлетворяет условиям, изложенным в пп. 1 - 4, является нерегулярным выражением.

Практически понятие регулярного выражения применяется как своеобразная стенография учеными, специализирующимися на вычислительной технике, лингвистами и другими учеными для обозначения последовательности символов.

Как было показано выше, *регулярные выражения* традиционно представляются загадочными греческими буквами. Однако в кодовой таблице ASCII не существует возможностей отображать всю эту чушь, поэтому представление регулярных выражений в Internet производится полностью с помощью печатаемых символов. Далее вы можете увидеть некоторые из таких символов. *При необходимости использовать литеральное выражение одного из символов, представленных в регулярном выражении* (например, использовать символ точки в конце предложения не в качестве группового символа),

этот символ необходимо предварять символом "\". Например, выражение "wor.." будет соответствовать "worry", "worst" и "words", а "wor.\." будет соответствовать как "word.", так и "work." и т.д.

И.1.1. Специальные символы

Начало строки. Например, следующая строка будет соответствовать любой строке, начинающейся с букв abed:

^abcd

$ Конец строки. Например, следующая строка будет соответствовать любой строке, которая завершается буквами abed:

abcd$

Соответствует любому символу.

И.1.2. Определение множества

Чтобы последовательность символов рассматривалась как варианты представления одного символа, символы должны быть заключены в квадратные скобки. Например,

```
[0123456789]
```

будет соответствовать одной цифре. Аналогичная функция может быть выполнена указанием диапазона символов ASCII от 0 до 9:

```
[0-9]
```

Очень могут пригодиться диапазоны [A-Z] и [a-z], с помощью которых задаются буквы в верхнем и нижнем регистрах соответственно.

Кроме того, символ "^", обозначающий начало строки при его использовании вне квадратных скобок, при использовании его внутри квадратных скобок будет инвертировать значение шаблона. Например, следующее регулярное выражение будет искать соответствие всех символов, которые *не* являются числами:

```
[^0-9]
```

И.1.3. Повторение предыдущего шаблона

Зачастую очень удобно показать, что заданный шаблон будет повторяться определенное количество раз. Для этого применяются следующие символы:

? Обозначает 0 или 1 повторение предыдущего символа или множества символов.

\+ Обозначает 1 или более повторений предыдущего символа или множества символов.

Обозначает 0 или более повторений предыдущего символа или множества символов.

Для определения шаблона, задающего 0 или 1 гласную букву, можно воспользоваться следующим регулярным выражением:

```
[aeiou]?
```

Для определения шаблона, задающего 1 или более цифр, можно воспользоваться следующим регулярным выражением:

```
[0-9]
```

Для определения шаблона строки, левый символ которой является буквой, можно воспользоваться следующим регулярным выражением:

^[A-Za-z]

И.1.4. Объявление и работа с последовательностью символов

Чтобы в строке замены задать ссылку на уже имеющийся шаблон, необходимо задать ее как последовательность символов. Последовательность символов выделяется квадратными скобками. Так, например, строка:

```
[A-Za-z]*([0-9])[A-Za-z]*
```

будет соответствовать любому количеству букв (в верхнем или нижнем регистре), за которыми следует одна цифра. За цифрой следует любое количество букв. В строках, в которых производится подстановка, это будет выглядеть следующим образом:

```
$1
```

Далее последовательностью символов будет $2, затем — $3 и т.д.

Приложение

K

ИНТЕРФЕЙС MOD_PERL API

K.1. Введение

В этой главе приведен простой перечень методов интерфейса с модулем mod_perl. Переменная $r содержит ссылку на запрошенный объект, который автоматически передается дескрипторам Perl при вызове.

K.1.1. Методы обработки клиентских запросов

```
$r = Apache->request();
$str = $r->args();
$c = $r->connection;
$str = $r->content();
$str = $r->filename($newval);
$r->finfo();
$str = $r->get_remote_host($lookup_type);
$str = $r->get_remote_logname();
$str = $r->header_in($hdr, $newval);
$bool = $r->header_only();
$href = $r->headers_in();
$str = $r->method($newval);
$num = $r->method_number($nv);
$u = $r->parsed_uri();
$str = $r->path_info($newval);
$str = $r->protocol();
$bool = $r->proxyreq($newval);
$r->read($buf, $bytes_to_read);
$s = $r->server # CM. Apache::Server
$str = $r->the_request();
$str = $r->uri($newval);
```

K.1.2. Методы ответа сервера

```
$num = $r->bytes_sent();
$r->cgi_header_out($hdr, $newval);
$str = $r->content_encoding($newval);
$aref = $r->content_languages($newval);
$str = $r->content_type($newval);
$r->custom_response($code, $uri);
```

```
$str = $r->err_header_out($hdr, $newval);
$href = $r->err_headers_out();
$str = $r->handler($newval) ;
$str = $r->header_out($hdr, $newval);
$href = $r->headers_out(};
$bool = $r->no_cache($newval);
$num = $r->request_time();
$num = $r->status($newval);
$str = $r->status_line($newval);
```

K.1.3. Посылка данных клиенту

```
$r->print(@list);
$r->printf($format, @args);
$r->rflush();
$r->send_cgi_header($str);
$len = $r->send_fd($filehandle);
$r->send_http_header($content_type);
```

K.1.4. Основные функции сервера

```
$r->chdir_file($file);
$r->child_terminate();
$r->hard_timeout($msg);
$r->internal_redirect($newplace);
$r->internal_redirect_handler($newplace);
$bool = $r->is_initial_req();
$bool = $r->is_main(};
$r->kill_timeout();
$str = $r->location ();
$req = $r->last();
$req = $r->main();
$req = $r->next();
$str = $r->notes($k, $v); # или $tab = $r->notes()
$req = $r->prev();
$r->register_cleanup($code_ref);
$r->reset_timeout();
$r->soft_timeout($msg);
$str = $r->subprocess_env($k, $v);
```

K.1.5. Методы конфигурирования сервера

```
$str = $r->dir_config($k);
$str = $r->document_root();
$str = $r->get_server_name();
$num = $r->get_server_port ();
$str = $r->server_root_relative($obj);
```

K.1.6 Класс Apache: : Log

```
$str = $r->as_string();
$r->log_reason($message, $file);
$r->log_error($message);
$r->warn($message);
$log = $r->log();
$log = $s->log();
$log->emerg($str ... $code_ref);
$log->alert($msg ... $code_ref);
$log->crit(5msg ... $code_ref);
$log->error($msg ... $code_ref);
```

```
$log->warn($msg ... $code_ref);
$log->notice($msg ... $code_ref);
$log->info($msg ... $code_ref);
$log->debug($msg ... $code_ref);
```

K.1.7. Методы управления доступом

```
$opts = $r->allow_options();
$str = $r->auth_name($newval) ;
$str = $r->auth_type();
($rc, $pw) = $r->get_basic_auth_pw();
$r->note_basic_auth_fallure();
$aref = $r->requires();
$flag = $r->satisfies();
$bool = $r->some_auth_required() ;
```

K.1.8. Специальные методы модуля mod_perl

```
$str = $r->current_callback();
$bool = $r->define($name);
Apache->exit($code);
$fh = Apache->gensyra();
$aref = $r->get_handlers($str);
Apache->httpd_conf($str);
$bool= $r->module($name);
$bool = Apache->perl_hook($name);
$r->post_connection($code_ref);
$r->push_handlers($str => $code_ref);
$r = Apache->request($r);
$r->set_handlers($str => $aref);
```

K.1.9. Класс Apache::SubRequest

```
$subr = $r->lookup_uri($uri);
$subr = $r->lookup_flie($filename);
$rc = $subr->run();
```

K.1.10. Класс Apache::Server

```
$s = Apache->server
$bool = $s->is_virtual();
$s->log_error();
$aref = $s->names();
$s = $s->next();
$num = $s->port();
$str = s->server_admin();
$str = s->server_hostname() ;
$num = s->timeout($newval);
$s->warn();
```

K.1.11. Класс Apache::Connection

```
$bool = $c->aborted();
$str = $c->auth_type();
$addr = $c->local_addr();
$addr = $c->remote_addr($addr);
$str = $c->remote_host();
$str = $c->remote_ip($ip);
$str = $c->remote_logname();
$str = $c->user($username);
```

K.1.12. Класс Apache::Table

```
Stab = Apache::Table->new($r, $tab->add($key,$str_or_aref)
$tab->clear();
$tab->do($code_ref);
$tab->merge($key, $str_or_aref);
$tab->set($key, $str);
$str = $tab->get($key);
$tab->unset($key);
```

K.1.13. Класс Apache::URI

```
$uri = Apache::URI->parse($r, $string_uri);
$str = $uri->unparse();
$str = $uri->cowponent($newval) ;
```

(где ***component*** может принимать одно из значений: fragment, hostinfo, hostname, password, path_info, path, port, query, rpath, scheme, user)

K.1.14. Класс Apache: :Util

```
$str = Apache::Util::escape_html($html);
$str = Apache::Util::escape_uri($uri) ;
$str = Apache::Util::ht_time($time, $fmt, $bool)
$secs = Apache::Util::parsedate($date_str);
$num = Apache::Util::size_string($num);
$str = Apache::Util::unescape_uri($uri);
$str = Apache::Util::unescape_uri_info($uri);
```

Приложение Л

ОПЕРАТОРЫ ЯЗЫКА PHP

В этом *приложении...*

Л.1. Функции сервера Apache

apache_lookup_uri. Эта функция выполняет частичный запрос к заданному URI и возвращает информацию о нем.

```
class apache_lookup_uri(string filename)
```

apache_note. Функция приема и установки заметок о запросе Apache.

```
string apache_note(string note_name[, string note_value])
```

getallheaders. Выбрать все заголовки HTTP-запроса.

```
array getallheaders(void)
```

virtual. Выполнить подзапрос Apache.

```
int virtual(string filename)
```

Л.2. Функции работы с числами произвольной точности

bcadd. Сложить два числа произвольной точности.

```
string bcadd(string left operand,
        string right operand [, int scale])
```

bccomp. Сравнить два числа произвольной точности.

```
int bccomp(string left_operand, string right_operand [, int scale])
```

bcdiv. Разделить два числа произвольной точности.

```
string bcdiv(string left_operand, string right_operand [, int scale])
```

bcmod. Взять модуль числа произвольной точности.

```
string bcmod(string left operand, string modulus)
```

bcmul. Умножить два числа произвольной точности.

```
string bcmul(string left operand, string right operand [, int scale])
```

bcpow. Возвести число произвольной точности в степень, представленную числом произвольной точности.

```
string bcpow(string x, string y [, int scale])
```

bcscale. Установить стандартный параметр масштабирования для всех математических функций.

```
string bcscale (int scale)
```

bcsqrt. Взять квадратный корень числа произвольной точности.

```
string bcsgrt(string operand, int scale)
```

bcsub. Вычесть одно число произвольной точности из другого.

```
string bcsub(string left_operand, string right_operand [, int scale])
```

Л.3. Функции массивов

array. Создать массив.

```
array array(.. .)
```

array_count_values. Подсчитать значения массива.

```
array array_count_values(array input)
```

array_flip. Сбросить весь массив.

```
array array_flip(array trans)
```

array_keys. Вернуть все ключевые величины массива.

```
array array_keys(array input [, mixed search_value])
```

array_merge. Слить один или более массивов.

```
array array_merge(array array1, array array2 [, array ...])
```

array jad. Заполнить массив определенным значением на определенную длину.

```
array array_pad(array input, int pad_size, mixed pad_value)
```

arrayjpop. Вытолкнуть элемент из конца массива.

```
mixed array_pop(array array)
```

array_push. Вытолкнуть один или более элементов из конца массива.

```
int array_push(array array, mixed var [, mixed ...])
```

array_reverse. Возвратить массив, элементы которого расположены в обратном порядке по отношению к исходному массиву.

```
array array_reverse(array array)
```

array_shift. Вытолкнуть элемент из начала массива.

```
mixed array_shift(array array)
```

array_slice. Выделить часть массива.

```
array array_slice(array array, int offset [, int length])
```

array_splice. Удалить часть массива и заменить ее.

```
array array_splice(array input, int offset [, int length [, array replacement]])
```

array_unshift. Продвинуть один или более элементов в начало массива.

```
int array_unshift(array array, [mixed var, mixed ...])
```

array_values. Возвратить значения массива.

```
array array_values(array input)
```

array_walk. Применить пользовательскую функцию к любому элементу массива.

```
int array_walk(array arr, string func, mixed userdata)
```

arsort. Отсортировать массив в обратном порядке и создать индекс.

```
void arsort(array array)
```

asort. Отсортировать массив и создать индекс.

```
void asort(array array)
```

compact. Создать массив, содержащий переменные и их значение.

```
array compact(mixed varname | [mixed...])
```

count. Подсчет элементов переменной.

```
int count(mixed var)
```

current. Возвратить текущий элемент в массиве.

```
mixed current(array array)
```

each. Возвратить из массива пару ключ-значение.

```
array each(array array)
```

end. Установить внутренний указатель массива на последний элемент массива.

```
end(array array)
```

extract. Импортировать переменные из массива в символическую таблицу.

```
void extract(array var_array, [int extract_type], [string prefix])
```

in_array. Возвратить значение "истина", если значение присутствует в массиве.

```
bool in_array(mixed needle, array haystack)
```

key. Выбрать ключ из ассоциативного массива.

```
mixed key(array array)
```

krsort. Отсортировать массив по ключу в обратном порядке.

```
int krsort(array array)
```

ksort. Отсортировать массив по ключу.

```
int ksort(array array)
```

list. Присвоить переменные, как если бы они присутствовали в массиве.

```
void list(...)
```

next. Переместить внутренний указатель массива вперед.

```
mixed next(array array)
```

pos. Извлечь текущий элемент из массива.

```
mixed pos(array array)
```

prev. "Перемотать" внутренний указатель массива.

```
mixed prev(array array)
```

range. Создать массив, содержащий диапазон целых чисел.

```
array range(int low, int high)
```

reset. Установить внутренний указатель массива на его первый элемент.

```
mixed reset(array array)
```

rsort. Отсортировать массив в обратном порядке.

```
void rsort(array array)
```

shuffle. Перемешать значения массива.

```
void shuffle(array array)
```

sizeof. Получить количество элементов в массиве.

```
int sizeof(array array)
```

sort. Отсортировать массив.

```
void sort(array array)
```

uasort. Отсортировать массив с помощью пользовательской функции сравнения и провести индексирование.

```
void uasort(array array, function cmp_function)
```

uksort. Отсортировать массив по ключам, полученным с помощью пользовательской функции сравнения.

```
void uksort(array array, function cmp_function)
```

usort. Отсортировать массив с помощью значений, полученных с помощью пользовательской функции сравнения.

```
void usort(array array, function cmp_function)
```

Л.4. GZlib

Выполнение операций с файлами, сжатыми утилитой gzip.

gzclose. Закрыть указатель на gz-файл.

```
int gzclose(int zp)
```

gzeof. Проверка маркера конца файла gz-файла.

```
int gzeof(int zp)
```

gzfile. Считать весь gz-файл в массив.

```
array gzfile(string filename [, int use_include_path])
```

gzgetc. Получить символ с помощью указателя на gz-файл.

```
string gzgetc(int zp)
```

gzgets. Получить строку с помощью указателя на файл.

```
string gzgets(int zp, int length)
```

gzgetss. Получить строку из указателя на gz-файл и снять теги HTML.

```
string gzgetss(int zp, intlength [, string allowable_tags])
```

gzopen. Открыть gz-файл.

```
int gzopen(string filename, string mode [, int use_include_path])
```

gzpassthru. Вывод всех остающихся данных по указателю на gz-файл.

```
int gzpassthru(int zp)
```

gzputs. Запись по указателю на gz-файл.

```
int gzputs(int zp, string str [, int length])
```

gzread. Сохранение прочитанного gz-файла в двоичном формате.

```
string gzread(int zp, int length)
```

gzrewind. Перемотка положения указателя gz-файла.

```
int gzrewind(int zp)
```

gzseek. Поиск указателя на gz-файл.

```
int gzseek(int zp, int offset)
```

gztell. Установить указатель на gz-файл в положение чтение/запись.

```
int gztell(int zp)
```

gzwrite. Сохранить в двоичном формате gz-файл.

```
int gzwrite(int zp, string string, int [, int length])
```

readgzfile. Прочесть файл, заархивированный в формате gz.

```
int readgzfile(string filename, [int use_include_path])
```

gzcompress. Заархивировать строку в формате gz.

```
string gzcompress(string data [, int level])
```

gzuncompress. Разархивировать строку, сжатую в формате gz.

```
string gzcompress(string data [, int length])
```

Л.5. Работа с базой данных DBM

Операции с базами данных DBM.

dba_close. Закрыть базу данных.

```
void dba_close(int handle)
```

dba_delete. Удалить запись, определенную ключом.

```
string dba_delete(string key, int handle)
```

dba_exists. Проверка ключа на существование.

```
bool dba_ex ists(string key, int handle)
```

dba_fetch. Выборка данных, заданных ключом.

```
string dba_fetch(string key, int handle)
```

dba_firstkey. Выбрать первый ключ.

```
string dba_firstkey(int handle)
```

dba_insert. Вставить запись.

```
bool dba_insert(string key, string value, int handle)
```

dba_nextkey. Выборка следующего ключа.

```
string dba_nextkey(int handle)
```

dba_popen. Открыть базу данных.

```
int dba_popen(string path, string mode, string handler [,
```

dba_open. Открыть базу данных.

```
int dba_open(string path, string mode, string handler [, ...])
```

dba_optimize. Оптимизировать базу данных.

```
bool dba_optimize(int handle)
```

dba_replace. Заменить или вставить запись.

```
bool dba_replace(string key, string value, int handle)
```

dba_sync. Синхронизировать базу данных.

```
bool dba_sync(int handle)
```

Л.6. Календарные функции

checkdate. Проверить дату/время.

```
int checkdate(int month, int day, int year)
```

date. Форматировать местное дату/время.

```
string date (string format, tint timestamp])
```

getdate. Получить информацию о дате/времени.

```
array getdate(int timestamp)
```

gettimeofday. Текущее время.

```
array gettimeofday(void)
```

gmdate. Получить дату/время.

```
string gmdate(string format, int timestamp)
```

gmmktime. Временная метка ОС Unix для даты GMT.

```
int gmmktime (int hour, int minute, int second, int month, int day,
int year [, int is_dst]);
```

gmstrftime. Форматирование даты/времени в соответствии с местными установками.

```
string gmstrftime(string format, int timestamp)
```

localtimex. Местное время.

microtime. Текущая временная метка ОС Unix с указанием миллисекунд.

```
string microtime(void)
```

mktime. Временная метка Unix для даты.

```
int mktime(int hour, int minute, int second, int month, int day, int
year, [int is_dst])
```

strftime. Форматирование местного отображения дата/время в соответствии с местными установками.

```
string strftime(string format, int timestamp)
```

time. Возвращает временную метку ОС Unix.

```
int   time(void)
```

strtotime. Преобразование почти любого текстового описания даты на английском языке во временную отметку ОС Unix.

```
int strtotime(string time [, int now])
```

Л.7. Функции взаимодействия с базой данных dBase

Эти функции обеспечивают взаимодействие с базой данных dBase.

dbase_create. Создание базы данных dBase.

```
int dbase_create(string filename, array fields)
```

dbase_open. Открыть запись из базы данных dBase.

```
int dbase_open(string filename, int flags)
```

dbase_close. Закрыть запись из базы данных dBase.

```
bool dbase_close(int dbase_identifier)
```

dbase pack. Упаковать базу данных dBase.

```
bool dbase_pack(int dbase_identifier)
```

dbase_add_record. Добавить запись в базу данных dBase.

```
bool dbase_add_record(int dbase_identifier, array record)
```

dbase_replace_record. Заменить запись в базе данных dBase.

```
bool dbase_replace_record(int dbase_identifier, array record, int dbase_record_number)
```

dbase_delete_record. Удалить запись из базы данных dBase.

```
bool dbase_delete_record(int dbase_identifier, int record)
```

dbase_get_record. Прочитать запись из базы данных dBase.

```
array dbase_get_record(int dbase_identifier, int record)
```

dbase_get_record_with_names. Прочитать запись из базы данных dBase как ассоциативный массив.

```
array dbase_get_record_with_names(int dbase_identifier, int recodr)
```

dbase_numfields. Определение количества полей в базе данных dBase.

```
int dbase_numfields (int dbase_identifier)
```

dbase_numrecords. Определение количества записей в базе данных dBase.

```
int dbase_numrecords(int dbase_identifier)
```

Л.8. Функции взаимодействия с базой данных DBM

dbmopen. Открыть базу данных DBM.

```
int dbmopen(string filename, string flags)
```

dbmclose. Закрыть базу данных DBM.

```
bool dbmclose(int dbm_identifier)
```

dbmexists. Проверить по ключу существование значений в базе данных DBM.

```
bool dbmexists(int dbm_identifier, string key)
```

dbmfetch. Выбрать значение из базы данных DBM по ключу.

```
string dbmfetch(int dbm_identifier, string key)
```

dbminsert. Вставить значение в базу данных DBM по ключу.
```
int dbminsert(int dbm_identifier, string key, string value)
```
dbmrcplace. Заменить значение в базе данных DBM по ключу.
```
bool dbmreplace(int dbm_identifier, string key, string value)
```
dbmdelete. Удалить значение из базы данных DBM по ключу.
```
bool dbmdelete(int dbm_identifier, string key)
```
dbmfirstkey. Выбор первого ключа из базы данных DBM.
```
string dbmfirstkey(int dbm_identifier)
```
dbmnextkey. Выбор следующего ключа из базы данных DBM.
```
string dbmnextkey(int dbm_identifier, string key)
```
dblist. Описание используемой DBM-совместимой библиотеки.
```
string dblist(void)
```

Л.9. Функции работы с каталогами

chdir. Изменить каталог.
```
int chdir(string directory)
```
dir. Класс каталога.
```
new dir(string directory)
```
closedir. Закрыть дескриптор каталога.
```
void closedir(int dir_handle)
```
opendir. Открыть дескриптор каталога.
```
int opendir(string path)
```
readdir. Прочесть запись из дескриптора каталога.
```
string readdir(int dir_handle)
```
rewinddir. "Перемотать" дескриптор каталога.
```
void rewinddir(int dir_handle)
```

Л.10. Функция динамической загрузки

dl. Загрузить PHP-расширение вовремя работы.
```
int dl(string library)
```

Л.11. Функции шифрования

mcrypt_get_cipher_name. Получить имя заданного шифра.
```
string mcrypt_get_cipher_name(int cipher)
```
mcrypt_get_block_size. Получить размер блока заданного шифра.
```
int mcrypt_get_block_size(int cipher)
```
mcrypt_get_key_size. Получить размер ключа заданного шифра.
```
int mcrypt_get_key_size(int cipher)
```
mcrypt_create_iv. Создать вектор инициализации (IV) из произвольного источника.
```
string mcrypt_create_iv(int size, int source)
```
mcrypt_cbc. Зашифровать или расшифровать данные в режиме CBC.

string mcrypt_cbc(int cipher, string key, string data, int mode [, string iv])

mcrypt_cfl>. Зашифровать или расшифровать данные в режиме CPB.

int mcrypt_cfb(int cipher, string key, string data, int mode, string iv)

mcrypt_ecb. Зашифровать или расшифровать данные в режиме ECB.

int mcrypt_ecb(int cipher, string key, string data, int mode)

mcrypt_ofb. Зашифровать или расшифровать данные в режиме OFB.

int mcrypt_ofb(int cipher, string key, string data, int mode, string iv)

Л.12. FilePro

filepro. Прочесть и проверить файл размещения.

bool filepro(string directory)

fflepro_fieldname. Получить имя поля.

string filepro_fieldname(int fiel'd_number)

filepro_fieldtype. Получить тип поля.

string filepro_fieldtype(int field_number)

filepro_fieldwidth. Получить ширину поля.

int filepro_fieldwidth(int field_number)

filepro_retrieve. Поиск данных из базыданных FilePro.

string filepro_retrieve(int row_number, int field_number)

filepro_fieldcount. Определение количества полей в базе данных FilePro.

int filepro_fieldcount(void)

filepro_rowcount. Определение количества строк в базе данных FilePro.

int filepro_rowcount(void)

Л.13. File System Functions

basename. Возвратить компонент имени файла из пути.

string basename(string path)

chgrp. Изменить группу файла.

int chgrp(string filename, mixed group)

chmod. Изменить права доступа к файлу.

int chmod(string filename, int mode)

chown. Изменить владельца файла.

int chown(string filename, mixed user)

clearstatcache. Очистить кэш статистики.

void clearstatcache(void)

copy. Скопировать файл.

int copy(string source, string dest)

delete. Ввод пустой записи.

void delete(string file)

dirname. Возвратить компонент имени каталога из дескриптора пути.

string dirname(string path)

diskfreespace. Возвратить свободное пространство, имеющееся в каталоге.

```
float diskfreespace(string directory)
```

fclose. Закрыть открытый указатель на файл.

```
int fclose(int fp)
```

feof. Проверить метку конца файла.

```
int feof(int fp)
```

fgetc. Выделить символ из указателя файла.

```
string fgetc(int fp)
```

fgetcsv. Получить строку из указателя на файл и проанализировать на наличие полей CSV.

```
array fgetcsvfint fp, int length, [string, delimiter])
```

fgets. Получить строку по указателю.

```
string fgets(int fp, int length)
```

fgetss. Получить строку по указателю на файл и удалить из нее теги HTML.

```
string fgetss(int fp, int length, [string allowable_tags])
```

file. Прочитать в массив весь файл.

```
array file (string filename [, int use_include_path])
```

file_exists. Проверить существование файла.

```
int file_exists(string filename)
```

fileatime. Получить время последнего доступа к файлу.

```
int fileatime(string filename)
```

filectime. Получить время изменения режима доступа к файлу.

```
int filectime(string filename)
```

filegroup. Получить группу, к которой принадлежит файл.

```
int filegroup(string filename)
```

fileinode. Получить значение inode для файла.

```
int fileinode(string filename)
```

filemtime. Получить время последней модификации файла.

```
int filemtime(string filename)
```

fileowner. Получить имя владельца файла.

```
int fileowner(string filename)
```

ffileperms. Получить права доступа к файлу.

```
int fileperms(string filename)
```

filesize. Получить размер файла.

```
int filesize(string filename)
```

filetype. Получить тип файла.

```
string filetype(string filename)
```

flock. Блокировка файла, рекомендуемая при переносе файла.

```
bool flock(int fp, int operation)
```

fopen. Открыть файл или URL.

```
int fopen(string filename, string mode [, int use_include_path])
```

fpassthru. Вывести все оставшиеся данные по указателю на файл.

```
int fpassthru(int fp)
```

fputs. Записать по указателю на файл.
```
int fputs(int fp, string str [, int length])
```
fread. Чтение файла в двоичном режиме.
```
string fread(int fp, int length)
```
fseek. Поиск по указателю на файл.
```
int fseek(int fp, int offset)
```
ftell. Определить положение указателя "чтение/запись".
```
int ftell(int fp)
```
fwrite. Запись в файл в двоичном режиме.
```
int fwrite(int fp, int buffer)
```
is_dir. Определить, есть ли файл filename в каталоге.
```
bool is_dir(string filename)
```
is_executable. Определить, является ли файл filename исполняемым.
```
bool is_executable(string filename)
```
is_file. Определить, является ли файл filename регулярным файлом.
```
bool is_file(string filename)
```
is_link. Определить, является ли файл filename **символической связью.**
```
bool is_link(string filename)
```
is_readable. Определить, открыт ли файл filename для чтения.
```
bool is_reada ble(string filename)
```
is_writeable. Определить, открыт ли файл filename для записи.
```
bool is_writeable(string filename)
```
link. Создать устойчивую связь.
```
int link(string target, string link)
```
linkinfo. Получить информацию о связи.
```
int linkinfo(string path)
```
mkdir. Создать каталог.
```
int mkdir(string pathname, int mode)
```
pclose. Закрыть указатель на файл процесса.
```
int pclose (int fp)
```
popen. Открыть указатель на файл процесса.
```
int popen(string command, string mode)
```
readfile. Вывести файл.
```
int readfile(string filename [, int use_include_path])
```
readlink. Возвратить результат символической связи.
```
string readlink(string path)
```
rename. Переименовать файл.
```
int rename(string oldname, string newname)
```
rewind. Изменить положение указателя на файл.
```
int rewind(int fp)
```
rmdir. Удалить каталог.
```
int rmdir(string dirname)
```

stat. Получить информацию о файле.

```
array stat(string filename)
```

lstat. Получить информацию о символической связи.

```
array lstat(string filename)
```

symlink. Создать символическую связь.

```
int symlink(string target, string link)
```

tempnam. Создать уникальное имя файла.

```
string tempnam(string dir, string prefix)
```

touch. Установить время модификациифайла.

```
int touch(string filename, int time)
```

umask. Изменить текущую установку umask.

```
int umask(int mask)
```

unlink. Удалить файл.

```
int unlink (string filename)
```

Л.14. Функции для работы с данными в FDF-формате

Обработка форм в FDF-формате.

fdf_open. Открыть документ в FDF-формате.

```
int fdf_open(string filename)
```

fdf_close. Закрыть документ в FDF-формате.

```
void fdf_close(int fdf_document)
```

fdf_create. Создать новый документ в FDF-формате.

fdf_save. Сохранить документ в FDF-формате.

```
int fdf_save(string filename)
```

fdf_get_value. Получить значение поля.

```
string fdf_get_value(int fdf_document, string fieldname)
```

fdf_set_value. Установить значение поля.

```
void fdf_set_value(int fdf_document, string fieldname, string value, int isName)
```

fdf_next_field_name. Получить имя следующего файла.

```
string fdf_next_field_name(int fdf_document, string fieldname)
```

fdf_set_ap. Установить вид поля.

```
void fdf_set_ap(int fdf_document, string field_name, int face, string filename, int page_number)
```

fdf_set_status. Установить значение ключа /STATUS.

```
void fdf_set_status(int fdf_document, string status)
```

fdf_get_status. Получить значение ключа /STATUS.

```
string fdf_get_status(int fdf_document)
```

fdf_set_file. Установить значение ключа /STATUS.

```
void fdf_set_file(int fdf_document, string filename)
```

fdf_get_flle. Получить значение клавиши /F.

```
string fdf_get_file(int fdf_document)
```

Л.15. FTP-функции

Функции, работающие с протоколом передачи файлов (File Transfer Protocol).

ftp_connect. Установить FTP-соединение.

```
int ftp_connect(string host [, int port])
```

ftp_login. Зарегистрировать FTP-соединение.

```
int ftp_login(int ftp_stream, string username, string password)
```

ftp_pwd. Возвратить имя текущего каталога.

```
int ftp_pwd(int ftp_stream)
```

ftp_cdup. Перейти в верхний каталог.

```
int ftp_cdup(int ftp_stream)
```

ftp_chdir. Перейти в другой каталог на FTP-сервере.

```
int ftp_chdir(int ftp_stream, string directory)
```

ftp_mkdir. Создать каталог.

```
string ftp_mkdir(int ftp_stream, string directory)
```

ftp_rmdir. Удалить каталог.

```
int ftp_rmdir(int ftp_stream, string directory)
```

ftp_nlist. Возвратить список файла определенного каталога.

```
int ftp_nlist(int ftp_stream, string directory)
```

ftp_rawlist. Возвратить подробный список файла определенного каталога.

```
int ftp_rawlist(int ftp_stream, string directory)
```

ftp_systype. Возвратить идентификатор типа удаленного FTP-сервера.

```
int ftp_systype(int ftp_stream)
```

ftp_pasv. Включение/выключение пассивного режима.

```
int ftp_pasv(int ftp_stream, int pasv)
```

ftp_get. Загрузить файл с FTP-сервера.

```
int ftp_get(int ftp_stream, string local_file, string remote_file,
            int mode)
```

ftp_fget. Загрузить файл с FTP-сервера в открытый файл.

```
int ftp_fget(int ftp_stream, int fp, string remote_file, int mode)
```

ftp_put. Загрузить файл на FTP-сервер.

```
int ftp_put(int ftp_stream, string remote_file, string local_file,
            int mode)
```

ftp_fput. Загрузить из открытого файла на FTP-сервер.

```
int ftp_fput(int ftp_stream, string remote_file, int fp, int mode)
```

ftp_size. Возвратить размер заданного файла.

```
int ftp_size(int ftp_stream, string remote_file)
```

ftp_mdtm. Возвратить время последней модификации заданного файла.

```
int ftp_mdtm(int ftp_stream, string remote_file)
```

ftp_rename. Переименовать файл на FTP-сервере.

```
int ftp_rename(int ftp_stream, string from, string to)
```

ftp_delete. Удалить файл на FTP-сервере.

```
int ftp_delete(int ftp_stream, string path)
```

ftp_site. Передать на сервер команду SITE.

```
int ftp_site(int ftp_stream, string cmd)
```

ftp_quit. Закрыть FTP-соединение.

```
int ftp_quit(int ftp_stream)
```

Л.16. Хэш-функции

Функции, предназначенные для работы с протоколом mhash.

mhash_get_hash_name. Получить имя заданного хэша.

```
string mhash_get_hash_name(inthash)
```

mhash_get_block_size. Размер блока заданного хэша.

```
int mhash_get_block_size(int hash)
```

mhash_count. Получить максимальный идентификатор хэша.

```
int mhash_count(void)
```

mhash. Вычислить хэш.

```
string mhash(int hash, string data)
```

Л.17. Функции HTTP

Обработка вывода, пересылаемого удаленному клиенту.

header. Послать заголовок HTTP.

```
int header(string string)
```

setcookie. Послать файл cookie.

```
int setcookie(string name, string value, int expire, string path,
              string domain, int secure)
```

Л.18. СУБД Informix

Функции взаимодействия с базами данных Informix.

ifx_connect. Установить соединение с СУБД Informix.

```
int ifx_connect([string database [, string userid [, string
                password]]])
```

ifx_pconnect. Установить устойчивое соединение с СУБД Informix.

```
int ifx_pconnect([string database [, string userid [, string
                 password]]])
```

ifx_close. Завершить связь с СУБД Informix.

```
int ifx_close([int link_identifier])
```

ifx_query. Передать запрос СУБД Informix.

```
int ifx_query(string query [, int link_identifier [, intcursor_type
              [, mixed blobidarray]]])
```

ifx_prepare. Подготовить оператор SQL для выполнения.

```
int ifx_prepare(string query, int conn_id [, int cursor_def, mixed
                blobidarray])
```

ifx_do. Выполнить предварительно приготовленный оператор SQL.

```
int ifx_do(int result_id)
```

ifx_error. Возвратить код ошибки последнего запроса к СУБД Informix.

```
string ifx_error(void)
```

ifx_errormsg. Возвратить сообщение об ошибке последнего запроса к СУБД Informix.

```
string ifx_errormsg([int errorcode])
```

ifx_affected_rows. Возвратить количество строк, задействованных в запросе.

```
int ifx_affected_rows(int result_id)
```

ifx_getsqlca. Получить содержимое sqlca.sqlerrd[0..5]после запроса.

```
array ifx_getsqlca (int result_id)
```

ifx_fetch_row. Получить строку как пронумерованный массив.

```
array ifx_fetch_row(int result_id [, mixed position])
```

ifx_htmltbl_result. Отформатировать результирующие строки запроса в HTML-таблицу.

```
int ifx_htmltbl_result(int result_id [, string html_table_options])
```

ifx_fieldtypes. Перечень полей СУБД Informix.

```
array ifx_fieldtypes(int result_id)
```

ifx_fieldproperties. Перечень свойств полей СУБД Informix.

```
array ifx_fieldproperties(int result_id)
```

irx_num_fields. Возврат количества столбцов в запросе.

```
int ifx_num_fields(int result_id)
```

ifx_num_rows. Подсчет строк, выбранных запросом.

```
int ifx_num_rows(int result_id)
```

ifx_free_result. Освободить ресурсы для запроса.

```
int ifx_free_result(int result_id)
```

ifx_create_char. Создать объект типа char.

```
int ifx_create_char(string param)
```

ifx_free_char. Удалить объект типа char.

```
int ifx_free_char(int bid)
```

ifx_update_char. Модифицировать содержимое объекта типа char.

```
int ifx_update_char(int bid, string content)
```

ifx_get_char. Возвратить содержимое объекта типа char.

```
int ifx_get_char(int bid)
```

ifx_create_blob. Создать объект типа blob.

```
int ifx_create_blob(int type, int mode, string param)
```

ifx_copy_blob. Дублировать данный объект типа blob.

```
int ifx_copy_blob(int bid)
```

ifx_free_blob. Удалить данный объект типа blob.

```
int ifx_free_blob(int bid)
```

ifx_get_blob. Возвратить содержимое объекта типа blob.

```
int ifx_get_blob(int bid)
```

ifx_update_blob. Модифицировать содержимое объекта типа blob.

```
ifx_update_blob(int bid, string content)
```

ifx_blobinffle_mode. Получить стандартный режим blob для всех запросов select.

```
void ifx_blobinfile_mode(int mode)
```

ifx_textasvarchar. Установить текстовый режим по умолчанию.

```
void ifx_textasvarchar(int mode)
```

ifx_byteasvarchar. Установить байтовый режим по умолчанию.

```
void ifx_byteasvarchar(int mode)
```

ifx_nullformat. Установить значение, возвращаемое по умолчанию.

```
void ifx_nullformat(int mode)
```

ifxus_create_slob. Создать объект типа slob и открыть его.

```
int ifxus_create_slob(int mode)
```

ifx_free_slob. Удалить объект типа slob.

ifxus_close_slob. Удалить объект типа slob.

```
int ifxus_close_slob(int bid)
```

ifxus_open_slob. Открыть объект типа slob.

```
int ifxus_open_slob(long bid, int mode)
```

ifxus_tell_slob. Возвратить текущий файл или позицию после операции seek.

```
int ifxus_tell_slob(long bid)
```

ifxus_seek_slob. Установить текущий файл или позицию после операции seek.

ifxus_read_slob. Прочесть nbytes объекта типа slob.

```
int ifxus_read_slob(long bid, long nbytes)
```

ifxus_write_slob. Записать строку в объект типа slob.

```
int ifxus_write_slob(long bid, string content)
```

Л.19. Почтовые функции

Рассылка электронной почты.

mail. Отправить электронную почту.

```
bool mail(string to, string subject, string message [, string
          additional headers])
```

Л.20. Математические функции

abs. Абсолютное значение.

```
mixed abs(mixed number)
```

acos. Арккосинус.

```
float acos(float arg)
```

asin. Арксинус.

```
float asin(float arg)
```

atan. Арктангенс.

```
float atan(float arg)
```

atan2. Арктангенс двух переменных.

```
float atan2(float y, float x)
```

base_convert. Преобразование чисел из одного произвольного основания в другое произвольное основание.

```
strin base_convert(string number, int frombase, int tobase)
```

bindec. Преобразование двоичных чисел в десятичные числа.

```
int bindec(string binary_string)
```

ceil. Округление до ближайшего большего целого.
```
int ceil(float number)
```
cos. Косинус.
```
float cos(float arg)
```
decbin. Преобразование десятичных чисел в двоичные числа.
```
string decbin(int number)
```
dechex. Преобразование десятичных чисел в шестнадцатеричные числа.
```
string dechex(int number)
```
decoct. Преобразование десятичных чисел в восьмеричные числа.
```
string decoct(int number)
```
deg2rad. Преобразовать градусы в радианы.
```
double deg2rad(double number)
```
exp. Возведение в степень.
```
float exp(float arg)
```
floor. Округление до ближайшего меньшего целого.
```
int floor(float number)
```
getrandmax. Максимальное произвольное число.
```
int getrandmax(void)
```
hexdec. Преобразование шестнадцатеричных чисел в десятичные числа.
```
int hexdec(string hex_string)
```
log. Натуральный логарифм.
```
float log (float arg)
```
log10. Логарифм с основанием 10.
```
float log10(float arg)
```
max. Найти максимальное значение.
```
mixed max(mixed arg1, mixed arg2, mixed argn)
```
min. Найти минимальное значение.
```
mixed min(mixed arg1, mixed arg2, mixed argn)
```
mt_rand. Генератор случайных чисел.
```
int mt_rand(int mt_rand([int min [, int max]])
```
mt_srand. Отсеять лучший генератор случайных чисел.
```
void mt_srand(int seed)
```
mt_getrandmax. Получить максимальное случайное число.
```
int mt_getrandmax(void)
```
number_format. Представить число в формате со сгруппированными тысячами.
```
string number_format(float number, int decimals, string dec_point,
                     string thousands_sep)
```
octdec. Преобразование восьмеричного представления числа в десятичное.
```
int octdec(string octal_string)
```
pi. Получить значение "пи".
```
double pi(void)
```
pow. Функция экспоненциального выражения.
```
float pow(float base, float exp)
```

rad2deg. Функция преобразования значения, заданного в радианах, в эквивалентное число, заданное в градусах.

```
double rad2deg(double number)
```

rand. Функция генерации случайного числа.

```
int rand([int min [, int max]])
```

round. Функция округления плавающего числа.

```
double round(double val)
```

sin. Функция синуса.

```
float sin(float arg)
```

sqrt. Функция квадратного корня.

```
float sqrt(float arg)
```

srand. Задать начальное значение для генератора случайных чисел.

```
void srand(int seed)
```

tan. Функция тангенса.

```
float tan(float arg)
```

Л.21. СУБД MS-SQL

Функции взаимодействия с СУБД Microsoft SQL.

mssql_close. Прервать соединение с сервером СУБД MS SQL.

```
int mssgl_close([int link_identifier])
```

mssql_connect. Установить соединение с сервером СУБД MS SQL.

```
int mssql_connect([string servername [, string username [, string
                  password]]])
```

mssql_data_seek. Перенести внутренний указатель на заданную строку.

```
int mssql_data_seek(int result_identifier, int row_number)
```

mssql_fetch_array. Выбрать строку в массив.

```
int mssql_fetch_array(int result)
```

mssql_fetch_field. Получить информацию о поле.

```
object mssql_fetch_field(int result)
```

mssql_fetch_object. Выбрать информацию о строке как объекте.

```
int mssql_fetch_object(int result)
```

mssql_fetch_row. Выбрать строку в нумерованный массив.

```
array mssql_fetch_row(int result)
```

mssql_field_length. Получить длину поля.

```
int mssql_field_length(int result [, int field_offset]])
```

mssql_field_name. Получить имя поля.

```
int mssql_field_name(int result [, int offset])
```

mssql_field_seek. Задать сдвиг поля.

```
int mssql_field_seek(int result, int field_offset)
```

mssql_field_type. Задать тип поля.

```
string mssql_field_type(int result [, int offset])
```

mssql_free_result. Освободить память.

```
int mssql_free_result(int result)
```

mssql_get_last_message. Возвратить последнее сообщение сервера.

```
stringmssql_get_last_message
```

mssql_min_error_severity. Установить минимальный уровень ошибок.

```
void mssql_min_error_severity
```

mssql_min_message_severity. Установить минимальный уровень серьезности регистрируемых сообщений.

```
void mssql_min_message_severity(int severity)
```

mssql_num_fields. Показать количество полей, полученных в результате запроса.

```
int mssql_num_fields(int result)
```

mssql_num_rows. Количество полученных строк.

```
int mssql_num_rows(stringresult)
```

mssql_pconnect. Установить устойчивое соединение с СУБД MS SQL.

```
int mssql_pconnect([string servername [, string username [, string
                    password]]])
```

mssql_query. Послать запрос на сервер MS SQL.

```
int mssql_query(string query [, int link_identifier])
```

mssql_result. Получить результаты выборки.

```
int mssql_result(int result, int i, mixed field)
```

mssql_select_db. Выборка базы данных СУБД MS SQL.

```
int mssql_select_db(string database_name [, int link_identifier])
```

Л.22. Разные функции

Функции, которым нельзя дать никакого другого определения.

connection_aborted. Возвратить значение "истина", если клиент отключился.

```
int connection_aborted(void)
```

connection_status. Возвратить бит состояния соединения.

```
int connection_status(void)
```

connection_timeout. Возвратить значение "истина" при превышении времени ожидания сценарием.

```
int connection_timeout(void)
```

define. Объявить константу.

```
int define(string name, mixed value [, int case_insensitive])
```

defined. Проверить указанную константу на существование.

```
int defined(string name)
```

die. Вывести сообщение и прервать выполнение текущего сценария.

```
void die(string message)
```

eval. Рассмотреть строку как PHP-код.

```
void eval(string code_str)
```

exit. Прервать выполнение текущего сценария.

```
void exit(void)
```

func_get_arg. Возвратить параметр из списка аргументов.

```
int func getarg(int arg num)
```

func_get_args. Возвратить массив, состоящий из списка аргументов функции.

```
int func_get_args(void)
```

func_num_args. Возвратить количество аргументов, переданных функции.

```
int func_num_args(void)
```

function_exists. Возвратить значение "истина", если заданная функция определена.

```
int function_exists(string function_name)
```

get_browser. Определить возможности броузера пользователя.

```
object get_browser([string user_agent])
```

ignore_user_abort. Установить, должно ли отключение клиента прекращать выполнение сценария.

```
int ignore_user_abort([int setting])
```

iptcparse. Разбить двоичный блок IPTC http://www.xe.net/iptc/block на отдельные теги.

```
array iptcparse(string iptcblock)
```

leak. Расход памяти.

```
int ignore_user_abort(int bytes)
```

pack. Упаковать данные в двоичную строку.

```
string pack(string format [, mixed args ...])
```

register_shutdown_function. Зарегистрировать функцию для выполнения при выключении.

```
int register_shutdown_function(string func)
```

serialize. Генерировать хранимое представление значения.

```
string serialize(mixed value)
```

sleep. Приостановить выполнение.

```
void sleep(int seconds)
```

uniqid. Создать уникальный идентификатор.

```
int uniqid(string prefix [, boolean leg])
```

unpack. Распаковать данные из двоичной строки.

```
array unpack(string format, string data)
```

unserialize. Создать значение PHP из сохраненного представления.

```
mixed unserialize(string str)
```

usleep. Задержать выполнение (в микросекундах).

```
void usleep(int micro_seconds)
```

Л.23. Функции взаимодействия с СУБД mSQL

Взаимодействие с СУБД mSQL.

msql. Передать запрос СУБД mSQL.

```
int msql(string database, string query, int link_identifier)
```

msql_affected_rows. Возвратить количество полученных строк.

```
int msql_affected_rows(int query_identifier)
```

msql_close. Завершить соединение с СУБД mSQL.

```
int msql_close(int link_identifier)
```

msql_connect. Установить соединение с СУБД mSQL.

```
int msql_connect(string hostname)
```

msql_create_db. Создать базу данных mSQL.
int msql_create_db(string database name (, int link_identifier])
msql_createdb. Создать базу данных mSQL.
int msql_createdb(string database name [, int link_identifier])
msql_data_seek. Перенести внутренний указатель настройки.
int msql_data_seek(int query_identifier, int row_number)
msql_dbname. Получить имя текущей базы данных.
string msql_dbname (int query_identifier, int i)
msql_drop_db. Удалить базу данных mSQL.
int msql_drop_db(string database_name, int link_identifier)
msql_error. Сообщение об ошибке последнего вызова mSQL.
string msql_error()
msql_fetch_array. Выбрать строку как массив.
int msql_fetch_array(int query_identifier [, int result_type])
msql_fetch_field. Получить информацию о поле.
object msql_fetch_field(int query_identifier, int field_offset)
msql_fetch_object. Выбрать строку как объект.
int msql_fetch_object(int query_identifier [, int result_type])
msql_fetch_row. Выбрать строку в пронумерованный массив.
array msql_fetch_row(int query identifier)
msql_fieldname. Получить имя поля.
string msql_fieldname(int query_identifier, int field)
msql_field_seek. Получить смещение поля.
int msql_field_seek (int query_identifier, int field_offset)
msql_fieldtable. Получить имя таблицы по имени поля.
int msql_fieldtable(int query_identifier, int field)
msql_fieldtype. Получить тип поля.
string msql_fieldtype(int query_identifier, int i)
msql_fieldflags. Получить флаги полей.
string msql_fieldflags(int query_identifier, int i)
msql_fieldlen. Получить длину поля.
int msql_fieldlen(int query_identifier, int i)
msql_free_result. Очистить память, занятую результатом последней операции.
int msql_free_result(int query_identifier)
msql_freeresult. Очистить память, занятую результатом последнейоперации.
msql_list_fields. Перечень полученных полей.
int msql_list_fields(string database, string tablename)
msq_listfields. Перечень полученных полей.
msql_list_dbs. Перечень баз данных mSQL, имеющихся на сервере.
int msql_list_dbs(void)
msql_listdbs. Перечень баз данных, имеющихся на сервере СУБД mSQL.
msql_list_tables. Перечень таблиц, имеющихся в базе данных mSQL.

```
int msql_list_tables(string database)
```

msql_listtables. Перечень таблиц в базе данных mSQL.

msql_num_fields. Получить количество результирующих полей.

```
int msql_num_fields(int query_identifier)
```

msql_num_rows. Получить количество результирующих строк.

```
int msql_num_rows(int query_identifier)
```

msql_numfields. Получить количество результирующих полей.

```
int msql_numfields(int query_identifier)
```

msql_numrows. Получить количество результирующих строк.

```
int msql_numrows(void)
```

msql_pconnect. Установить устойчивое соединение с mSQL.

```
int msql_pconnect(string hostname)
```

msql_query. Послать запрос mSQL.

```
int msql_query(string query, int link_identifier)
```

msql_regcase. Создать регулярное выражение для выборки данных без распознавания регистра.

msql_result. Получить результирующие данные.

```
int msql_result(int query_identifier, int i, mixed field)
```

msql_select_db. Выборка базы данных mSQL.

```
int msql_select_db(string database_name, int link_identifier)
```

msql_selectdb. Выборка базы данных mSQL.

msql_tablename. Получить имя таблицы по имени поля.

```
string msql_tablename(int query_identifier, int field)
```

Л.24. Функции, работающие с СУБД MySQL

Эти функции обеспечивают взаимодействие с СУБД MySQL.

mysql_affected_rows. Количество задействованных строк в предыдущей операции MySQL.

```
int mysql_affected_rows([int link_identifier])
```

mysql_change_user. Поменять зарегистрировавшегося пользователя на активное соединение.

```
int mysql_change_user(string user, string password [, string
                      database [, int link_identifier]])
```

mysql_close. Завершить соединение с базой данных.

```
int mysql_close([int link_identifier])
```

mysql_connect. Установить соединение с сервером MySQL.

```
int mysql_connect([string hostname [:port] [:/path/to/socket] [
                  string username [, string password]]])
```

mysql_create_db. Создать базу данных MySQL.

```
int mysql_create_db(string database name [, int link_identifier])
```

mysql_data_seek. Переместить внутренний указатель.

```
int mysql_data_seek(int result_identifier, int row_number)
```

mysql_db_query. Послать запрос MySQL.

```
int mysql_db_query(string database, string query [, int
                   link_identifier])
```

mysql_drop_db. Удалить базу данных.

```
int mysql_drop_db(string database_name [, int link_identifier])
```

mysql_errno. Возвратить номер сообщения об ошибке предыдущей операции.

```
int mysql_errno([int link_identifier])
```

mysql_error. Возвратить сообщение об ошибке о предыдущей операции.

```
string mysql_error([int link_identifier])
```

mysql_fetch_array. Возвратить результирующую строку как ассоциативный массив.

```
array mysql^fetch_array(int result [, int result_type])
```

mysql_fetch_field. Принять информацию о столбце и возвратить его как объект.

```
object mysql_fetch_field(int result [, int field_offset])
```

mysql_fetch_lengths. Длина каждого полученного результата.

```
array mysql_fetch_lengths(int result)
```

mysql_fetch_object. Выборка результирующей строки как объекта.

```
object mysql_fetch_object(int result [, int result_typ])
```

mysql_fetch_row. Выборка результирующей строки в пронумерованный массив.

```
array mysql_fetch_row(int result)
```

mysql_field_name. Имя заданного результирующего поля.

```
string mysql_field_name(int result, int field_index)
```

mysql_field_seek. Установить указатель по смещению поля.

```
int mysql_field_seek(int result, int field_offset)
```

mysql_field_table. Получить имя заданного поля.

```
string mysql_field_table(int result, int field_offset)
```

mysql_field_type. Получить тип указанного результирующего поля.

```
string mysql_field_type(int result, int field_offset)
```

mysql_field_flags. Флаг, связанный с указанным полем в полученном результате.

```
string mysql_field_flags(int result, int field_offset)
```

mysql_field_len. Возвратить длину заданного поля.

```
int mysql_field_len(int result, int field_offset)
```

mysql_free_result. Освободить память, занятую результатом предыдущего запроса.

```
int mysql free result(int result)
```

mysql_insert_id. Идентификатор предыдущей операции INSERT,

```
int mysql_insert_id([int link_identifier])
```

mysql_list_fields. Перечень результирующих полей.

```
int mysql_list_fields(string database_name, string table_name [, int
                      link_identifier])
```

mysql_list_dbs. Перечень баз данных, имеющихся на сервере MySQL.

```
int mysql_list_dbs([int link_identifier])
```

mysql_list_tables. Получить перечень таблиц для данной базы данных MySQL.

```
int mysql_list_tables(string database [, int link_identifier])
```

mysql_num_fields. Получить число результирующих строк.

```
int mysql_num_fields (int result)
```

mysql_num_rows. Получить количество результирующих строк.

```
int mysql_num_rows(int result)
```

mysql_pconnect. Установить устойчивое соединение с сервером MySQL.

```
int mysql_pconnect([string hostname [:port] [:/path/to/socket] [,
                   string username [, string password]]])
```

mysql_query. Послать SQL-запрос базе данных MySQL.

```
int mysql_query(string query [, int link_identifier])
```

mysql_result. Получить результирующие данные.

```
int mysql_result(int result, int row [, mixed field])
```

mysql_select_db. Выборка базы данных MySQL.

```
int mysql_select_db(string database_name [, int link_identifier])
```

mysql_tablename. Получить по имени поля имя таблицы.

```
string mysql tablename(int result, int i)
```

Л.25. Сетевые функции

Взаимодействие с системой на сетевом уровне.

checkdnsrr. Проверить записи базы DNS относительно заданного имени узла или IP-адреса.

```
int checkdnsrr(string host [, string type])
```

closelog. Закрыть соединение с системным регистрационным журналом.

```
int closelog(void)
```

debugger_off. Отключить внутренний отладчик PHP.

```
int debugger_off(void)
```

debugger_on. Включить внутренний отладчик PHP.

```
int debugger_on(string address)
```

fsockopen. Открыть Internet-соединение или Unix-соединение с сокетом домена.

```
int fsockopen(string hostname, int port [, int errno [, string
              errstr [, double timeout]]])
```

gethostbyaddr. Получить полное имя узла, соответствующее заданному IP-адресу.

```
string gethostbyaddr(string ip_address)
```

gethostbyname. Получить IP-адрес, соответствующий заданному имени узла.

```
string gethostbyname(string hostname)
```

gethostbynamel. Получить список IP-адресов, соответствующих заданному имени узла.

```
array gethostbynamel(string hostname)
```

getmxrr. Получить MX-записи, соответствующие заданному имени узла.

```
int getmxrr(string hostname, array mxhosts, [array weight])
```

getprotobyname. Получить номер протокола, связанный с именем протокола.

```
int getprotobyname(string name)
```

getprotobynumber. Получить имя протокола, связанное с номером протокола.

```
string getprotobynumber(int number)
```

getservbyname. Получить номер порта, связанный с сервисом Internet и протоколом.

```
int getservbyname(string service, string protocol)
```

getservbyport. Получить сервис Internet, связанный с номером порта и протоколом.

```
string getservbyport(int port, string protocol)
```

openlog. Установить связь с системным журналом.

```
int openlog (string ident, int option, int facility)
```

pfsockopen. Создать устойчивое Internet-соединение или Unix соединение с сокетом домена.

```
int pfsockopen(string hostname, int port [, int errno [, stri:
                errstr [, int timeout]]])
```

set_socket_blocking. Установить режим блокирования/разблокирования сокета.

```
int set_socket_blocking(int socket descriptor, int mode)
```

syslog. Создать системное регистрационное сообщение.

```
int syslog(int priority, string message)
```

Л.26. Функции NIS

Взаимодействие с сетевым информационным сервером (NIS).

yp_get_default_domain. Возвратить стандартный домен NIS для компьютера.

```
int yp_get_default_domain(void)
```

yp_order. Возвратить порядковый номер карты размещения.

```
int yp_order(string domain, string map)
```

yp_master. Возвратить имя компьютера, содержащее главный сервер NIS, для карты размещения.

```
string yp_master(string domain, string map)
```

yp_match. Возвратить строку, соответствующую шаблону.

```
string yp_match(string domain, string map, string key)
```

yp_first. Возвратить первую пару ключевых значений для указанной карты размещения.

```
string[] yp_first(string domain, string map)
```

yp_next. Возвратить следующую пару ключевых значений для указанной карты размещения.

```
string[] yp_next(string domain, string map, string key)
```

Л.27. ODBC-функции

Функции взаимодействия с протоколом Open DataBase Connectivity.

odbc_autocommit. Включение/выключение режима autocommit.

```
int odbc_autocommit(int connection_id [, int OnOff])
```

odbc_binmode. Обработка данных столбца двоичного типа данных.

```
int odbc_binmode(int result_id, int mode)
```

odbc_close. Закрыть ODBC соединение.

```
void odbc_close (int connection_id)
```

odbc_close_all. Закрыть все ODBC соединения.

```
void odbc_close_all (void)
```

odbc_commit. Выполнить ODBC транзакцию.

```
int odbc_commit(int connection_id)
```

odbc_connect. Подключиться к источнику данных.

```
int odbc_connect(string dsn, string user, string password [, in
                 cursor_type])
```

odbc_cursor. Получить имя курсора.

```
string odbc_cursor(int result_id)
```

odbc_do. Синоним **odbc_exec().**

```
string odbc_do(int conn_id, string query)
```

odbc_exec. Приготовить и выполнить оператор SQL.

```
int odbc_exec(int connection_id, string query_string)
```

odbc_execute. Выполнить готовый оператор SQL.

```
int odbc_execute(int result_id [, array parameters_array])
```

odbc_fetch_into. Выбрать одну результирующую строку в массив.

```
int odbc_fetch_into(int result_id [, int rownumber, array
                    result_array])
```

odbc_fetch_row. Выборка строки.

```
int odbc_fetch_row(int result_id [, int row_number])
```

odbc_field_name. Получить имя столбца.

```
string odbc_field_name(int result_id, int field_number)
```

odbc_field_type. Тип данных поля.

```
string odbc_field_type(int result_id, int field_number)
```

odbc_field_len. Получить длину (точность) поля.

```
int odbc_field_len(int result_id, int field_number)
```

odbc_free_result. Освободить ресурсы, связанные с результатом.

```
int odbc_free_result(int result_id)
```

odbc_longreadlen. Обработка столбцов типа LONG.

```
int odbc_longreadlen(int result_id, int length)
```

odbc_num_flelds. Число столбцов в результате.

```
int odbc_num_fields(int result_id)
```

odbc_pconnect. Установить устойчивое соединение с базой данных.

```
int odbc_pconnect(string dsn, stringuser, string password [, int
                  cursor_type])
```

odbc_prepare. Подготовка оператора к выполнению.

```
int odbc_prepare(int connection_id, string query_string)
```

odbc_num_rows. Число строк в результате.

```
int odbc_num_rows (int result_id)
```

odbc_result. Получить результирующие данные.

```
string odbc_result(int result_id, mixed field)
```

odbc_result_all. Распечатать результат в формате HTML-таблицы.

```
int odbc_result_all(int result_id [, string format])
```

odbc_rollback. Откатить транзакцию.

```
int odbc_rollback(int connection_id)
```

odbc_setoption. Настройка установок ODBC. Возвращает значение "ложь" в случае возникновения ошибки.

```
int odbc_setoption(int id, int function, intoption, int param)
```

odbc_tables. Получить список имен таблиц, хранящихся в специальном источнике данных. Возвращает результирующий идентификатор, содержащий эту информацию.

```
int odbc_tables(int connection_id [, string qualifier [, string
    owner [, string name [, string types]]]])
```

odbc_tableprivileges. Список таблиц и связанных с ними привилегий.

```
int odbc_tableprivileges(int connection_id [, string qualifier [,
    string owner [, string name]]])
```

odbc_columns. Список имен столбцов в заданных таблицах. Возвращает результирующий идентификатор, содержащий эту информацию.

```
int odbc_columns(int connection_id [, string qualifier [, string owner
    [, string table_name [, string column_name]]]])
```

odbc_columnprivileges. Возвращает результирующий идентификатор, который может быть использован для выборки перечня столбцов с соответствующими привилегиями.

```
int odbc_columnprivileges(int connection_id [, string qualifier [,
    string owner [, string table_name [, string
    column_name]]]])
```

odbc_gettypeinfo. Возвращает результирующий идентификатор, содержащий информацию о типах данных, которые поддерживаются источником данных.

```
int odbc_gettypeinfo(int connection_id [, int data_type])
```

odbc_primarykeys. Возвращает результирующий идентификатор, который может быть использован для выборки имен столбцов, удовлетворяющих первичному ключу таблицы.

```
int odbc_primarykeys(int connection_id, string qualifier, string
    owner, string table)
```

odbc_foreignkeys. Возвращает внешние ключи в заданной таблице или список внешних ключей в других таблицах, которые связаны с первичными ключами в заданной таблице.

```
int odbc_foreignkeys(int connection_id, string pk_qualifier, string
    pk_owner, string pk_table, string fk_qualifier,
    string fk_owner, string fk_table)
```

odbc_procedures. Получить список процедур, сохраненных в заданном источнике данных. Возвращает результирующий идентификатор, содержащий информацию.

```
int odbc_procedures(int connection_id [, string qualifier [, string
    owner [, string name]]])
```

odbc_procedurecolumns. Выборка информации о параметрах процедур.

```
int odbc_procedurecolumns(int connection id [, string qualifier [,
    string owner [, string proc [, string
    column]]]])
```

odbc_specialcolumns. Возвращает оптимальный набор столбцов, которые уникально определяют строку в таблице или столбцы, автоматически модифицирующиеся при модификации любого значения в строке.

```
int odbc_specialcolumns(int connection_id, int type, string
    qualifier, string owner, string table, int
    scope, int nullable)
```

odbc_statistics. Статистические данные о таблице.

```
int odbc_statistics(int connection_id, string qualifier, string owner,
    string table_name, int unique, int accuracy)
```

Л.28. СУБД Oracle

Взаимодействие с СУБД Oracle.

Ora_Bind. Связать переменную языка PHP с параметром Oracle.

```
int Ora_Bind(int cursor, string PHP_variable_name, string
        SQL_parameter_name, int length [, int type])
```

Ora_Close. Закрыть курсор СУБД Oracle.

```
int Ora_Close(int cursor)
```

Ora_CoIumnName. Получить имя результирующего столбца Oracle.

```
string Ora_ColumnName(int cursor, int column)
```

Ora_ColumnType. Получить тип результирующего столбца Oracle.

```
string Ora_ColumnType(int cursor, int column)
```

Ora_Commit. Выполнить транзакцию СУБД Oracle.

```
int Ora_Commit(int conn)
```

Ora_CommitOff. Отменить автоматическое выполнение транзакций.

```
int Ora_Cortvmitoff (int conn)
```

Ora_CommitOn. Включить автоматическое выполнение транзакций.

```
int Ora_Commiton(int conn)
```

Ora_Error. Получить сообщение СУБД Oracle.

```
string Ora_Error(int cursor_or_connection)
```

Ora_ErrorCode. Получить код ошибки.

```
int Ora_ErrorCode(int cursor_or_connection)
```

Ora_Exec. Выполнить проанализированный оператор в курсоре СУБД Oracle.

```
int Ora_Exec(int cursor)
```

Ora_Fetch. Выборка строки данных из курсора.

```
int Ora_Fetch(int cursor)
```

Ora_GetColumn. Получить данные из выбранной строки.

```
mixed Ora_Getcolumn(int cursor, mixed column)
```

Ora_Logoff. Закрыть соединение Oracle.

```
int Ora_Logoff(int connection)
```

Ora_Logon. Открыть соединение Oracle.

```
int Ora_Logon(string user, string password)
```

Ora_Open. Открыть курсор Oracle.

```
int Ora_0pen(int connection)
```

Ora_Parse. Проанализировать оператор SQL.

```
int Ora_Parse(int cursor_ind, string sql_statement, int defer)
```

Ora_RoIlback. Откатить транзакцию.

```
int Ora_Rollback(int connection)
```

Л.29. СУБД Oracle 8

Взаимодействие с СУБД Oracle 8.

OCIDefineByName. Использовать переменную PHP для объявления во время операции SELECT.

```
int OCIDefineByName(int stmt, string Column-Name, mixed Svariable [,
                    int type])
```

OCIBindByName. Привязать переменную PHP к маркеру Oracle.

```
int OCIBindByName(int stmt, string ph_name, mixed svariable,
                  intlength [, int type])
```

OCILogon. Установить соединение с базой данных Oracle.

```
int OCILogon(string username, string password [, string db])
```

OCIPLogon. Установить устойчивое соединение с базой данных Oracle и зарегистрироваться при новом соединении. Возвращает новый сеанс.

```
int OCIPLogon(string username, string password [, string db])
```

OCINLogon. Установить соединение с базой данных Oracle и зарегистрироваться при новом соединении. Возвращает новый сеанс.

```
int OCINLogon(string username, string password [, string db])
```

OCILogOff. Отключиться от сервера Oracle.

```
int OCILogOff(int connection)
```

OCIExecute. Выполнить оператор.

```
int OCIExecute(int statement [, int mode])
```

OCICommit. Выполнить просроченные транзакции.

```
int OCICommit(int connection)
```

OCIRollback. Откатить просроченные транзакции.

```
int OCIRollback(int connection)
```

OCINewDescriptor. Инициализировать новый пустой дескриптор LOB/FILE (по умолчанию LOB).

```
string OCINewDescriptor(int connection [, int type])
```

OCIRowCount. Выбрать количество задействованных строк.

```
int OCIRowCount(int statement)
```

OCINumCols. Возвратить количество полученных столбцов в операторе.

```
int OCINumCols(int stmt)
```

OCIResuIt. Возвратить значение столбца для выбранной строки.

```
mixed OCIResuIt(int statement, mixed column)
```

OCIFetch. Выбрать следующую строку в буфер.

```
int OCIFetch(int statement)
```

OCIFetchInto. Выбрать следующую строку в массив result.

```
int OCIFetchInto(int stmt, array &result [, int mode])
```

OCIFetchStatement. Выбрать все полученные данные в массив.

```
int OCIFetchStatement(int stmt, array Svariable)
```

OCIColumnIsNULL. Проверить содержимое столбца на наличие значения NULL.

```
int OCIColumnlsNULL(int stmt, mixed column)
```

OCIColumnSize. Возвратить размер результирующего столбца.

```
int OCIColumnSize(int stmt, mixed column)
```

OCIServerVersion. Возвратить строку, содержащую информацию о версии сервера.

```
string OCIServerVersion(int conn)
```

OCIStatementType. Возвратить тип оператора OCI.

```
string OCIStatementType(int stmt)
```

OCINewCursor. Возвратить новый курсор (дескриптор оператора).

```
int OCINewCursor(int conn)
```

OCIFreeStatement. Освободить все ресурсы, связанные с оператором.

```
int OCIFreeStatement(int stmt)
```

OCIFreeCursor. Освободить все ресурсы, связанные с курсором.

```
int OCIFreeCursor(int stmt)
```

OCIColumnName. Возвращает имя столбца.

```
string OCIColumnName(int stmt, int col)
mixed OCIColumnName(int stmt, int col)
```

OCIColumnType. Возвращает тип данных столбца.

```
mixed OCIColumnType(int stmt, int col)
```

OCIParse. Анализирует запрос и возвращает оператор.

```
int OCIParse(int conn, strint query)
```

OCIError. Возвращает последнюю ошибку stmt | conn I global. Возвращает значение "ложь"' при отсутствии ошибки.

```
int OCIError([int stmtI conn|global])
```

OCIInternalDebug. Включение/выключение вывода информации при внутренней отладке. По умолчанию эта возможность отключена.

```
void OCHnternalDebug (int onoff)
```

Л.30. Регулярные выражения языка Perl

Обработка регулярных выражений языка Perl.

preg_match. Выполнить поиск регулярного выражения.

```
int preg_match(string pattern, string subject [, array matches])
```

preg_match_all. Выполнить глобальный поиск регулярного выражения.

```
int preg_match_all(string pattern, string subject, array matches [,
                   int order])
```

preg_replace. Выполнить поиск и замену регулярного выражения.

```
mixed preg_replace(mixed pattern, mixed replacement, mixed subject)
```

preg_split. Разбить строку на регулярные выражения.

```
array preg_split(string pattern, string subject [, int limit [, int
                 flags]])
```

preg_quote. Взять в кавычки символы регулярного выражения.

```
string preg_quote(string str)
```

preg_grep. Возвратить элементы массива, соответствующие шаблону.

```
array preg_grep(string pattern, array input)
```

Л.31. Функции POSIX

Взаимодействие с подмножеством функций POSIX.

posix_kill. Послать сигнал процессу.

```
bool posix_kill(int pid, int sig)
```

posix_getpid. Возвратить идентификатор текущего процесса.

```
int posix_getpid(void)
```

posix_getppid. Возвратить идентификатор процесса-родителя.

```
int posix_getppid(void)
```

posix_getuid. Возвратить реальный идентификатор пользователя текущего процесса.

```
int  posix_getuid(void)
```

posix_geteuid. Возвратить эффективный групповой идентификатор пользователя текущего процесса.

```
int posix_geteuid(void)
```

posix_getgid. Возвратить реальный групповой идентификатор текущего процесса.

```
int posix_getgid(void)
```

posix_getegid. Возвратить эффективный идентификатор группы текущего процесса.

```
int posix_getegid(void)
```

posix_setuid. Установить эффективный идентификатор пользователя текущего процесса.

```
bool posix_setuid(int uid)
```

posix_setgid. Установить эффективный идентификатор группы текущего процесса.

```
bool posix_setgid(int gid)
```

posix_getgroups. Возвратить набор группы для текущего процесса.

```
array posix_getgroups(void)
```

posix_getlogin. Возвратить регистрационное имя.

```
string posix_getlogin(void)
```

posix_getpgrp. Возвратить идентификатор группы, к которой принадлежит данный процесс.

```
int posix_getpgrp(void)
```

posix_setsid. Сделать текущий процесс лидером сеанса.

```
int posix_setsid(void)
```

posix_setpgid. Установить идентификатор группы, к которой принадлежит процесс.

```
int  posix_setpgid(int pid,  int pgid)
```

posix_getpgid. Идентификатор группы, которой принадлежит процесс.

```
int posix_getpgid(int pid)
```

posix_getsid. Текущий идентификатор sid процесса.

```
int posix getsid(int pid)
```

posix_uname. Получить системное имя.

```
array posix_uname(void)
```

posix_times. Получить времена процесса.

```
array posix_times(void)
```

posix_ctermid. Получить путь управляющего терминала.

```
stringposix_ctermid(void)
```

posix_ttyname. Определить имя терминального устройства.

```
string posix_ttyname(int fd)
```

posix_isatty. Определить, является ли дескриптор файла интерактивным терминалом.

```
bool posix_isatty(int fd)
```

posix_getcwd. Путь текущего каталога.

```
string posix_getcwd(void)
```

```
bool posix_getcwd(string pathname, int mode)
```

posix_getgrnam. Возвратить информацию о группе по имени группы.

```
array posix_getgrnam(string name)
```

posix_getgrgid. Возвратить информацию о группе по идентификатору группы.

```
array posix_getgrgid(int gid)
```

posix_getpwnam. Возвратить информацию о пользователе по имени пользователя.

```
array posix_getpwnam(string username)
```

posix_getpwuid. Возвратить информацию о пользователе по идентификатору пользователя.

```
array posix_getpwuid(int uid)
```

posix_getrlimit. Возвратить информацию о предельных значениях ресурсов системы.

```
array posix_getrlimit(void)
```

Л.32. Функции выполнения программ

Функции, позволяющие выполнять другие программы.

escapeshellcmd. Метасимволы выхода из оболочки.

```
string escapeshellcmd(string command)
```

exec. Выполнить внешнюю программу.

```
string exec(string command [, string array [, int return_var]])
```

passthru. Выполнить внешнюю программу и вывести неформатированный вывод.

```
void passthru(string command [, int return_var])
```

system. Выполнить внешнюю программу и отобразить вывод.

```
string system(string command [, int return_var])
```

Л.33. Recode

Использовать функции перекодировки GNU.

recode_string. Перекодировать строку в соответствии с запросом перекодировки.

```
string recode_string(stringrequest, string string)
```

recede. Перекодировать строку в соответствии с запросом перекодировки.

```
string recode_string(string request, string string)
```

recode_file. Перекодировать из одного файла в другой файл в соответствии с запросом перекодировки.

```
bool recode_file(int input, int output)
```

Л.34. Функции, работающие с сеансами

Сохранить данные во время последующих сеансов доступа.

session_start. Инициализировать данные сеанса.

```
bool session_start(void)
```

session_destroy. Удалить все данные, зарегистрированные во время сеанса.

```
bool session_destroy(void)
```

session_name. Получить и/или установить имя текущего сеанса.

```
string session_name([string name])
```

session_module_name. Получить и/или установить модуль текущего сеанса.

```
string session_module_name([string module])
```

session_save_path. Получить и/или установить сохраненный путь текущего сеанса.

```
string session_save_path(string path)
```

session_id. Получить или установить идентификатор текущего сеанса.

```
string session_id([string id])
```

session_register. Зарегистрировать одну или более переменных во время сеанса.

```
bool session_register(mixed name [, mixed ...])
```

session_unregister. Отменить регистрацию переменных во время сеанса.

```
bool session_unregister(string name)
```

session_unset. Освободить все переменные сеанса.

```
void session_unset(void)
```

session_is_registered. Проверить, зарегистрирована ли переменная во время сеанса.

```
bool session_is_registered(string name)
```

session_get_cookie_params. Получить параметры cookie-файлов.

```
arraysession_get_cookie_params(void)
```

session_set_cookie_params. Установить параметры cookie-файлов.

```
void session_set_cookie_params(int lifetime [, string path [, string
                               domain]])
```

session_decode. Декодировать данные текущего сеанса из строки.

```
bool session_decode(string data)
```

session_encode. Закодировать данные текущего сеанса в виде строки.

```
bool session_encode(void)
```

Л.35. Функции протокола SNMP

Функции протокола SNMP.

snmpget. Выбрать объект SNMP.

```
string snmpget(string hostname, string community, string object_id
               [, int timeout [, int retries]])
```

snmpset. Установить объект SNMP.

```
bool snmpset(string hostname, string community, string object_id,
             string type, mixed value [, int timeout [, int retries]])
```

snmpwalk. Выбрать все объекты SNMP.

```
array snmpwalk(string hostname, string community, string object_id
               [, int timeout [, int retries]])
```

snmpwalkoid. Запрос информации об элементе сети.

```
array snmpwalkoid(string hostname, string community, string
                  object_id [, int timeout [, int retries]])
```

snmp_get_quick_print. Выбрать текущее значение переменной quick_print библиотеки UCD.

```
boolean snmp_get_quick_print(void)
```

snmp_set_quick_print. Установить текущее значение переменной quick_print библиотеки UCD.

```
void snmp_set_quick_print(boolean quick_print)
```

Л.36. Строковые функции

Обработка строковых значений.

addcslashes. Ограничить строку в стиле комментариев языка C (две косых черты).

```
string addcslashes(string str, string charlist)
```

addslashes. Ограничить строку косой чертой.

```
string addslashes (string str)
```

bin2hex. Преобразовать двоичные данные в шестнадцатеричное представление.

```
string bin2hex(string str)
```

chop. Удалить пробел в конце строки.

```
string chop (string str)
```

chr. Возвратить символ.

```
string chrfint ascii)
```

chunk_split. Разбить строку на подстроки.

```
string chunk_split(string string [, int chunklen [, string end]])
```

convert_cyr_string. Преобразовать из одного набора кириллических символов в другой.

```
string convert_cyr_string(string str, string from, string to)
```

count_chars. Возвратить информациюо символах, используемых в строке.

```
mixed count_chars (string string [, mode])
```

crypt. Зашифровать строку.

```
string crypt(string str [, string salt])
```

echo. Вывод одной или более строк.

```
echo(string argl, string ....)
```

explode. Разбить строку на отдельные строки.

```
array explode(string separator, string string)
```

flush. Сбросить буфер вывода.

```
void flush(void)
```

get_html_translation_table. Возвратить таблицу трансляции, которая используется функциями `htmlspecialchars()` **и** `htmlentities()`.

```
string get_html_translation_table(int table)
```

get_meta_tags. Извлечь все атрибуты и метатеги содержимого из файла и возвратить массив.

```
array get_meta_tags(string filename [, int use_include_path])
```

htmlentities. Преобразовать все возможные символы в сущности HTML.

```
string htmlentities(string string)
```

htmlspecialchars. Преобразовать специальные файлы в сущности HTML.

```
string htmlspecialchars(string string)
```

implode. Объединить элементы массива в строку.

```
string implode(string glue, array pieces)
```

join. Объединить элементы массива со строкой.

```
string join(string glue, array pieces)
```

levenshtein. Вычислить расстояние Ливенштейна между двумя строками.

```
int levenshtein(string str1, string str2)
```

ltrim. Убрать пробел в начале строки.
```
string ltrim(string str)
```
md5. Вычислить хэш md5 для строки.
```
string md5(string str)
```
metaphone. Вычислить метафонный ключ строки.
```
string metaphone (string str)
```
nl2br. Преобразовать начало строк в разрывы строк в формате HTML.
```
string nl2br(string string)
```
ord. Возвратить ASCII-код символа.
```
int ord (string string)
```
parse_str. Разложить строки на переменные.
```
void parse_str(string str)
```
print. Вывести строку.
```
print(string arg)
```
printf. Вывести отформатированную строку.
```
int printf(string format [, mixed args . . . ] )
```
quoted_printable_decode. Преобразовать строки, взятые в кавычки, в 8-битовые строки.
```
string quoted_printable_decode(string str)
```
quotemeta. Взять метасимволы в кавычки.
```
string quotemeta(string str)
```
rawurldecode. Декодировать строки, закодированные URL.
```
string rawurldecode(string str)
```
rawurlencode. Кодировка URL в соответствии со стандартом RFC 1738.
```
string rawurlencode(string str)
```
setlocale. Установить локальную информацию.
```
string setlocale(string category, string locale)
```
similar_text. Вычислить подобие между двумя строками.
```
int similar_text(string first, string second [, double percent])
```
soundex. Определение звукового аналога строки.
```
string soundex(string str)
```
sprintf. Возвратить отформатированную строку.
```
string sprintf(string format [, mixed args...])
```
strcasecmp. Двоичное сравнение строк независимо от регистра.
```
int strcasecmp(string str1, string str2)
```
strchr. Найти первое появление символа в строке.
```
string strchr(string haystack, string needle)
```
strcmp. Двоичное сравнение строки.
```
int strcmp(string str1, string str2)
```
strcspn. Определить длину начального сегмента, не соответствующего маске.
```
int strcspn(string str1, string str2)
```
strip_tags. Убрать из строки HTML-теги и PHP-теги.
```
string strip_tags(string str [,string allowable_tags])
```

stripcslashes. Убрать выделение строки косой чертой.

```
string stripcslashes (string str)
```

stripslashes. Убрать выделение строки косой чертой.

```
string stripslashes(string str)
```

stristr. Найти первое появление подстроки в строке независимо от регистра символов.

```
string stristr(string haystack, string needle)
```

strlen. Получить длину строки.

```
int strlen(string str)
```

strpos. Найти первое появление подстроки в строке.

```
int strpos(string haystack, string needle [, int offset])
```

strrchr. Найти последнее появление символа в строке.

```
string strrchr(string haystack, string needle)
```

str_repeat. Повторить строку.

```
string str_repeat(string input, int multiplier)
```

strrev. Инвертировать порядок символов в строке.

```
string strrev(string string)
```

strrpos. Определить положение последнего символа char в строке.

```
int strrpos(string haystack, char needle)
```

strspn. Определить длину маски соответствия начального сегмента.

```
int strspn(string str1, string str2)
```

strstr. Обнаружить первое появление строки.

```
string strstr (string haystack, string needle)
```

strtok. Пометить строку.

```
string strtok(string arg1, string arg2)
```

strtolower. Преобразовать строку в нижний регистр.

```
string strtolower(string str)
```

strtoupper. Преобразовать строку в верхний регистр.

```
string strtoupper(string string)
```

str_replace. Заменить все появления подстроки needle в строке haystack строкой str.

```
string str_replace(string needle, string str, string haystack)
```

strtr. Преобразовать определенные символы.

```
string strtr(string str, string from, string to)
```

substr. Возвратить часть строки.

```
string substr(string string, int start [, int length])
```

substr_replace. Заменить текст в части строки.

```
string substr_replace(string string, string replacement, int start
                      [, int length])
```

trim. Удалить пробелы из начала и из конца строки.

```
string trim(string str)
```

ucfirst. Перевести первый символ в строке в верхний регистр.

```
string ucfirst(string str)
```

ucwords. Перевести первый символ каждого слова в строке в верхний регистр.

```
string ucwords(string str)
```

Л.37. Функции СУБД Sybase

Взаимодействие с СУБД Sybase.

sybase_affected_rows. Получить количество строк, обработанных последним запросом.

```
int sybase_affected_rows ([int link_identifier])
```

sybase_close. Отключить от СУБД Sybase.

```
int sybase_close(int link_identifier)
```

sybase_connect. Подключение к серверу СУБД Sybase.

```
int sybase_connect(string servername, string username, string
                   password)
```

sybase_data_seek. Переместить внутренний указатель.

```
int sybase_data_seek(int result_identifier, int row_number)
```

sybase_fetch_array. Выбрать строку как массив.

```
int sybase_fetch_array(int result)
```

sybase_fetch_field. Получить информацию о поле.

```
object sybase_fetch_field(int result, int field_offset)
```

sybase_fetch_object. Выбрать строку как объект.

```
int sybase_fetch_object(int result)
```

sybase_fetch_row. Получить строку как пронумерованный массив.

```
array sybase_fetch_row(int result)
```

sybase_field_seek. Установить смещение поля.

```
int sybase_field_seek(int result, int field_offset)
```

sybase_free_result. Освободить результирующую память.

```
int sybase_free_result(int result)
```

sybase_num_fields. Количество полученных полей.

```
int sybase_num_fields(int result)
```

sybase_num_rows. Количество полученных строк.

```
int sybase_num_rows(string result)
```

sybase_pconnect. Установить устойчивое соединение с сервером Sybase.

```
int sybase_pconnect(string servername, string username, string
                    password)
```

sybase_query. Передать запрос Sybase.

```
int sybase_query(string query, int link_identifier)
```

sybase_result. Получить результирующие данные.

```
int Sybase_result(int result, int i, mixed field)
```

sybase_select_db. Выбрать базу данных Sybase.

```
int sybase_select_db(string database_name, int link_identifier)
```

Л.38. Функции URL

Кодирование и декодирование URL-строк.

base64_decode. Декодировать данные, закодированные с применением кода mime base64.

```
string base64_decode(string encoded_data)
```

base64_encode. Закодировать данные с применением кода mime base64.

```
string base64_encode(string data)
```

parse_url. Проанализировать URL и возвратить его компоненты.

```
array parse_url(string url)
```

urldecode. Декодировать закодированные URL-строки.

```
string urldecode(string str)
```

urlencode. Закодировать URL-строки.

```
string urlencode(string str)
```

Л.39. Функции, управляющие переменными

Эти функции позволяют устанавливать и проверять характеристики переменных.

call_user_func. Вызвать пользовательскую функцию, заданную первым параметром.

```
mixed call_user_func(string function_name [, mixed parameter [,
                     mixed ...]])
```

doubleval. Получить значение переменной типа double.

```
double doubleval(mixed var)
```

empty. Проверка установки переменной.

```
int empty(mixed var)
```

gettype. Получить тип переменной.

```
string gettype(mixed var)
```

intval. Получить целое значение переменной.

```
int intval (mixed var [, int base])
```

is_array. Определить, является ли переменная массивом.

```
int is_array(mixed var)
```

is_double . Определить, имеет ли переменная тип double.

```
int is_double(mixed var)
```

is_float. Определить, имеет ли переменная тип float.

```
int is_float(mixed var)
```

is_int. Определить, имеет ли переменная тип integer.

```
int is_int(mixed var)
```

is_integer. Определить, имеет ли переменная тип integer.

```
int is_integer(mixed var)
```

is_long. Определить, имеет ли переменная тип long.

```
int is_long(mixed var)
```

is_object. Определить, является ли переменная объектом.

```
int is_object(mixed var)
```

is_real. Определить, имеет ли переменная тип real.

```
int is_real(mixed var)
```

is_string. Определить, является ли переменная строкой.

```
int is_string(mixed var)
```

isset. Определить, установлена ли переменная.

```
int isset(mixed var)
```

print_r. Напечатать читабельную информацию о переменной.

```
void print_r(mixed expression)
```

settype. Установить тип переменной.

```
int settype(string var, string type)
```

strval. Получить строковое значение переменной.

```
string strval(mixed var)
```

unset. Отменить установку переменной.

```
int unset(mixed var)
```

var_dump. Вывести информацию о переменной.

```
void var_dump(mixed expression)
```

Предметный указатель

Ж

З

И

К

Т

У

Ф

Ш

Я

Научно-популярное издание

Скотт Хокинс

Администрирование Web-сервера Apache и руководство по электронной коммерции

Литературный редактор *Н.В. Никифорова*
Верстка *О. В. Линник*
Художественные редакторы *В.Г. Павлютин, С.А. Чернокозинский*
Технический редактор *Г.Н. Горобец*
Корректоры *Л. В. Коровкина, О. В. Мишутина*

Издательский дом "Вильяме".
101509, Москва, ул. Лесная, д. 43, стр. 1.
Изд. лиц. ЛР № 090230 от 23.06.99
Госкомитета РФ по печати.

Подписано в печать 03.10.2001. Формат 70x100/16.
Гарнитура Times. Печать офсетная.
Усл. печ. л. 20,6. Уч.-изд. л. 16.
Тираж 5000 экз. Заказ № 1814.

Отпечатано с диапозитивов в ФГУП "Печатный двор"
Министерства РФ по делам печати,
телерадиовещания и средств массовых коммуникаций.
197110, Санкт-Петербург, Чкаловский пр., 15.