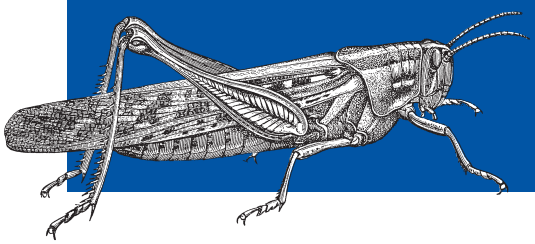
*Руководство для системных администраторов*

5-е издание
включает BIND 9.3

# DNS и BIND



Символ®
O'REILLY®

*Крикет Ли и Пол Альбитц*

# DNS and BIND

Fifth Edition

*Cricket Liu and Paul Albitz*

O'REILLY®

# DNS и BIND

Пятое издание

*Крикет Ли и Пол Альбитц*

*Санкт-Петербург — Москва*
*2008*

Крикет Ли, Пол Альбитц

# DNS и BIND, 5-е издание

Перевод М. Зислиса


| | |
|---|---|
| Главный редактор | *А. Галунов* |
| Зав. редакцией | *Н. Макарова* |
| Редактор | *В. Овчинников* |
| Корректор | *О. Макарова* |
| Верстка | *О. Макарова* |

*Ли К., Альбитц П.*
DNS и BIND, 5-е издание. – Пер. с англ. – СПб.: Символ-Плюс, 2008. – 712 с., ил.
ISBN-10: 5-93286-105-3
ISBN-13: 978-5-93286-105-9



Книга «DNS и BIND» стала библией для системных администраторов. Она уникальна по полноте изложения материала, что в сочетании с прекрасным авторским стилем делает ее незаменимой и актуальной для каждого, кто хочет наладить эффективную работу DNS. В пятом издании обсуждаются BIND 9.3.2 (последняя версия в ветви BIND 9) и BIND 8.4.7. BIND 9.3.2 включает усовершенствования безопасности и поддержки IPv6, а также ряд новых возможностей, таких как ENUM, SPF и использование имен доменов, содержащих буквы национальных алфавитов.

Рассмотрены следующие темы: функциональность и принципы работы DNS; структура пространства доменных имен; установка и настройка серверов имен; применение MX-записей для маршрутизации почты; настройка узлов на работу с DNS; разделение доменов на поддомены; обеспечение безопасности DNS-сервера; расширения системы безопасности DNS (DNSSEC) и подписи транзакций (TSIG); распределение нагрузки между DNS-серверами; динамические обновления, асинхронные уведомления об изменениях зоны, пошаговая передача зон; разрешение проблем (nslookup и dig, чтение отладочной диагностики); программирование при помощи функций библиотеки DNS-клиента.



**ISBN–10: 5–93286–105–3**
**ISBN–13: 978–5–93286–105–9**
**ISBN 0–596–10057–4 (англ)**





Издательство «Символ-Плюс». 199034, Санкт-Петербург, 16 линия, 7,
тел. (812) 324-5353, www.symbol.ru. Лицензия ЛП N 000054 от 25.12.98.
Налоговая льгота – общероссийский классификатор продукции
ОК 005-93, том 2; 953000 – книги и брошюры.

Подписано в печать 28.01.2008. Формат 70x100 1/16 . Печать офсетная.
Объем 44,5 печ. л. Тираж 2000 экз. Заказ N
Отпечатано с готовых диапозитивов в ГУП «Типография «Наука»
199034, Санкт-Петербург, 9 линия, 12.

# Оглавление

# Предисловие

Возможно, вам не так уж много известно о системе доменных имен (Domain Name System), но, работая в Интернете, вы неизбежно ее используете. Всякий раз, отправляя сообщения электронной почты или исследуя просторы World Wide Web, вы полагаетесь на DNS – систему доменных имен.

Дело в том, что люди предпочитают запоминать *имена* компьютеров, а компьютерам больше нравится обращаться друг к другу по числовым адресам. В Интернете этот адрес имеет разрядность 32, то есть может быть числом от нуля до четырех с хвостиком миллиардов.[1] Компьютеры с легкостью запоминают такие вещи, потому что обладают большими объемами памяти, идеально подходящей для хранения чисел, но для людей эта задача не в пример сложнее. Попробуйте случайным образом выбрать из телефонной книги десять номеров и запомнить их. Непросто? Теперь вернитесь к началу телефонной книги и сопоставьте каждому номеру случайный код района. Примерно настолько же сложно будет запомнить 10 произвольных интернет-адресов.

Отчасти именно по этой причине необходима система доменных имен. DNS занимается двунаправленным отображением имен узлов, подходящих для запоминания людьми, и интернет-адресов, с которыми работают компьютеры. По сути дела, DNS в сети Интернет является не только средством работы с адресами, но и стандартным механизмом для предоставления и получения разнообразной информации об узлах сети. DNS нужен практически для каждой программы, обеспечивающей сетевое взаимодействие, в том числе программам для работы с электронной почтой, терминальным клиентам (например, ssh), средствам передачи файлов, таким как ftp, и, разумеется, веб-броузерам, таким как Microsoft Internet Explorer.

Другой важной особенностью DNS является способность системы распространять информацию об узле по всей сети Интернет. Хранение доступной информации об узле на единственном компьютере полезно лишь для тех, кто пользуется этим компьютером. Система доменных имен обеспечивает получение информации из любой точки сети.

Более того, DNS позволяет распределять управление информацией об узлах между многочисленными серверами и организациями. Нет необ-

---

[1] А в системе IP-адресации версии 6 адреса имеют колоссальную длину – 128 бит, что позволяет охватить десятичные числа от 0 до 39-значных.

ходимости передавать данные на какой-то центральный сервер или регулярно синхронизировать свою базу данных с «основной». Достаточно убедиться, что ваш раздел, называемый *зоной*, соответствует действительности на ваших *DNS-серверах*. А они, в свою очередь, сделают информацию о зоне доступной всем остальным DNS-серверам сети.

Поскольку база данных DNS является распределенной, в системе должна быть предусмотрена возможность поиска нужной информации путем опроса множества возможных источников ее получения. Система доменных имен наделяет DNS-серверы способностью находить нужные источники информации и получать сведения по любой зоне.

Разумеется, система DNS не лишена недостатков. К примеру, в целях избыточности базы данных система позволяет хранить зональную информацию на более чем одном сервере, но при этом возникает опасность десинхронизации копий зональной информации.

Но *самая большая* проблема, связанная с DNS, несмотря на широкое распространение в сети Интернет, – это реальное отсутствие хорошей документации по работе с системой. Большинство администраторов сети Интернет вынуждены обходиться лишь той документацией, которую считают достаточной поставщики используемых программ, а также тем, что им удается выудить из соответствующих списков интернет-рассылок и конференций Usenet.

Такой дефицит документации означает, что понимание предельно важной интернет-службы, одной из монументальных основ сегодняшней сети Интернет, либо передается от администратора к администратору как ревностно хранимая семейная тайна, либо постоянно изучается повторно отдельными программистами и разработчиками. Новые администраторы зон повторяют ошибки, уже бесчисленное число раз сделанные другими.

Цель этой книги – изменить сложившуюся ситуацию. Мы осознаем, что не у каждого читателя есть время и желание становиться специалистом по DNS. У большинства из вас есть достаточно других занятий помимо управления зонами и DNS-серверами: системное администрирование, разработка сетевых инфраструктур или разработка программного обеспечения. Заниматься исключительно DNS может только сотрудник безумно большой организации. Мы постарались предоставить информацию, достаточную для решения основных рабочих задач, будь то управление небольшой зоной или целой международной системой, работа с единственным сервером имен или наблюдение за сотней серверов. Извлеките из книги нужный вам минимум и возвращайтесь к ней по мере необходимости.

DNS – это сложная тема, настолько сложная, что взяться за нее пришлось не одному, а двум авторам; но мы постарались представить систему настолько прозрачно и доступно, насколько это возможно. В первых двух главах содержится теоретический обзор и достаточный для

применения объем практической информации, а в последующих главах использование системы доменных имен рассмотрено более подробно. С самого начала мы предлагаем читателям нечто вроде дорожной карты, чтобы каждый мог выбрать собственный путь изучения книги, соответствующий рабочим задачам или интересам.

Когда речь пойдет о программах, обеспечивающих работу DNS, мы практически целиком сконцентрируемся на инструменте под названием BIND, Berkeley Internet Name Domain, который является наиболее популярной (и наиболее нами изученной) реализацией спецификаций DNS. Мы старались представить в этой книге выжимку из нашего опыта управления и поддержки зон с помощью BIND. (Так получилось, что некоторое время одна из наших зон являлась самой большой зоной сети Интернет; правда, это было очень давно). Где это было возможно, мы включали реальные программы, используемые нами в администрировании; многие из них переписаны на языке Perl с целью достижения большей скорости работы и повышения эффективности.

Надеемся, эта книга поможет вам познакомиться с системой DNS и инструментом BIND, если вы еще новичок, лучше понять их работу, если вы с ними уже знакомы, и приобрести ценное понимание и опыт, даже если вы уже знаете DNS и BIND как свои пять пальцев.

## Версии

Четвертое издание этой книги затрагивает новые версии BIND – 9.3.2 и 8.4.7, а также более старые версии BIND 8 и 9. Несмотря на то, что на момент написания этой книги версии 9.3.2 и 8.4.7 являются наиболее свежими, они пока не получили широкого распространения в составе UNIX-систем – отчасти потому, что обе версии были выпущены недавно, а многие поставщики настороженно относятся к использованию новых программ. Мы время от времени упоминаем и другие версии BIND, поскольку многие поставщики продолжают распространять программы, содержащие код, основанный на более старых версиях, в составе своих UNIX-разработок. Если определенная возможность доступна только в версии 8.4.7 или 9.3.2 либо существуют различия в поведении версий, мы постараемся четко определить, что именно работает и для какой версии BIND.

В наших примерах мы очень часто прибегаем к служебной программе DNS – *nslookup*. Мы пользуемся *nslookup* из комплекта поставки BIND версии 9.3.2. Более старые версии *nslookup* обеспечивают большую часть функциональности (но не всю) *nslookup* версии 9.3.2. В большинстве примеров мы использовали команды, доступные почти во всех версиях *nslookup*; случаи, когда это было невозможно, отмечены отдельно.

## Что нового в пятом издании?

Текст книги был обновлен, чтобы соответствовать наиболее поздним версиям BIND; добавлен следующий новый материал:

- Описание технологии SPF (Sender Policy Framework) – в главе 5.
- Более подробное рассмотрение динамических обновлений и механизма NOTIFY, включая и подписываемые динамические обновления (signed dynamic updates), а также описание нового для BIND 9 механизма *update-policy* – в главе 10.
- Поэтапная передача зоны – также в главе 10.
- Зоны ретрансляции, поддерживающие передачу по условию (conditional forwarding), – в главе 10.
- Прямое и обратное отображение адресов в контексте технологии IPv6 с использованием записей новых типов AAAA и ip6.arpa – в конце главы 10.
- Новый механизм подтверждения подлинности транзакций – транзакционные подписи (transaction signatures, известные также как TSIG) – описан в главе 11.
- Более подробное рассмотрение вопросов обеспечения безопасности DNS-серверов – в главе 11.
- Более подробное рассмотрение работы с брандмауэрами в сети Интернет – в главе 11.
- Описаны обновленные расширения DNS, связанные с безопасностью (DNS Security Extensions или DNSSECbis), представляющие собой механизм цифровой подписи зональных данных, – все в той же 11 главе.
- Новая глава 16 посвящена развертыванию полноценной архитектуры DNS в масштабах организации.
- В главе 17 описывается ENUM, технология для отображения телефонных номеров в формате стандарта E.164 в URI-адреса.
- Стандарт кодирования символов Unicode в именах доменов (IDN, Internationalized Domain Names) описан в главе 17.
- Обновлен раздел, посвященный совместной работе Active Directory и BIND, – в главе 17.

## Структура

Порядок следования глав настоящей книги приблизительно соответствует возможному развитию зоны и росту знаний ее администратора. В главах 1 и 2 обсуждается теория системы доменных имен. В главах с 3 по 6 рассматриваются вопросы, связанные с принятием решений по созданию собственных зон, а также действия администратора в случае необходимости создать зону. Следующая часть книги, главы с 7 по 11,

посвящена сопровождению зон, настройке узлов для использования DNS-серверов, планированию развития зон, созданию доменов различных уровней и безопасности серверов. Наконец, главы с 12 по 16 посвящены разрешению сложностей, возникающих при работе с различными инструментами, общим проблемам и забытому искусству программирования с применением библиотек DNS-клиента. Глава 16 сводит знания в единый архитектурный ансамбль. Перечислим темы по главам:

Глава 1 «Основы»

Описывает исторический фон создания системы, посвящена проблемам, приведшим к созданию DNS, а также собственно обзору теории системы доменных имен.

Глава 2 «Как работает DNS»

Посвящена более подробному рассмотрению теоретических основ DNS, в частности организации пространства имен в системе DNS, доменов, зон и DNS-серверов. Там же рассматриваются такие важные понятия, как разрешение адресов и кэширование.

Глава 3 «С чего начать?»

Рассматриваются получение пакета BIND в случае его отсутствия, применение пакета, когда он уже у вас в руках, определение и выбор доменного имени, а также установление связи с организацией, которая обладает полномочиями делегировать выбранную зону.

Глава 4 «Установка BIND»

Подробное рассмотрение того, как установить два первых DNS-сервера на основе BIND, включая создание базы данных серверов, запуск и диагностику их работы.

Глава 5 «DNS и электронная почта»

Рассказывает о записи DNS типа MX, которая позволяет администраторам задавать альтернативные узлы, которым передается на обработку почта для определенных адресов. В этой главе описаны стратегии маршрутизации почты для различных типов сетей и узлов, включая сети с интернет-брандмауэрами и узлы, не имеющие прямого подключения к сети Интернет. В этой главе также повествуется о технологии Sender Policy Framework, позволяющей использовать DNS для авторизации отправления почты с определенных почтовых адресов.

Глава 6 «Конфигурирование узлов»

Рассказывает о том, как настраивать клиентскую часть (*resolver*) BIND, а также об особенностях реализаций клиента, применяемых на платформах Windows.

Глава 7 «Работа с BIND»

Посвящена регулярным действиям администратора, выполнение которых необходимо для поддержания устойчивой работы зон, находящихся под его началом, в частности проверке состояния DNS-сервера и вопросам, касающимся авторитетных серверов зоны.

Глава 8 «Развитие домена»

Рассказывает о планировании роста и эволюции зон, включая вопросы о том, как вырасти большим, а также о планировании переездов и перебоев в работе.

Глава 9 «Материнство»

О радостях, связанных с обретением потомства. Мы расскажем, когда имеет смысл заводить детей (создавать поддомены), как их называть, как их заводить (!) и как присматривать за ними.

Глава 10 «Дополнительные возможности»

Рассказывает о параметрах настройки сервера имен, которые используются не очень часто, но могут помочь в настройке производительности DNS-сервера и упростить процесс администрирования.

Глава 11 «Безопасность»

Посвящена обеспечению безопасности и тем настройкам DNS-сервера, которые относятся к работе с интернет-брандмауэрами, а также двум новым технологиям DNS, связанным с безопасностью: DNS Security Extensions и подписям транзакций (Transaction Signatures).

Глава 12 «nslookup и dig»

Подробно рассказывает о самых популярных инструментах DNS-отладки и содержит описания способов извлечения неявной информации из удаленных DNS-серверов.

Глава 13 «Чтение отладочного вывода BIND»

Это Розеттский камень[1] отладочной информации BIND. Глава поможет разобраться в таинственной отладочной информации, создаваемой пакетом BIND, а это, в свою очередь, поможет лучше понять, как работает DNS-сервер.

Глава 14 «Разрешение проблем DNS и BIND»

Содержит описания и способы разрешения многих распространенных проблем, связанных с использованием DNS и BIND, а также

[1] Розеттский камень – черная базальтовая плита с трехъязычной надписью, обнаруженная в 1799 г. при сооружении форта Сен-Жюльен на берегу Розеттского рукава Нила. Расшифровка иероглифического текста в 1822 г. стала началом изучения египетской иероглифической письменности. – *Примеч. ред.*

рассказывает о более редких случаях, связанных с ошибками, диагностика которых может вызывать затруднения.

Глава 15 «Программирование с использованием библиотечных функций»

Рассказывает о том, как использовать функции библиотеки клиента BIND для опроса DNS-серверов и получения информации в программе на языке C или Perl. Приводится исходный текст полезной (как мы надеемся) программы, которая проверяет работоспособность DNS-серверов и их авторитетность.

Глава 16 «Архитектура»

Описывает полноценную инфраструктуру DNS, включающую внешние DNS-серверы, ретрансляторы, а также внутренние DNS-серверы.

Глава 17 «Обо всем понемногу»

Посвящена незатронутым темам. Она содержит описание использования масок (wildcards) в DNS, принципов работы с узлами и сетями, не имеющими постоянного подключения к сети Интернет, кодировки сетевых имен, дополнительных типов записей ENUM и IDN, а также работы с Active Directory.

Приложение A «Формат сообщений DNS и RR-записи»

Содержит предельно подробный справочник по форматам, используемым в запросах и ответах DNS, а также полный перечень определенных в настоящее время типов RR-записей (resource records).

Приложение B «Таблица совместимости BIND»

Перечисление наиболее важных особенностей самых распространенных версий BIND.

Приложение C «Сборка и установка BIND на Linux-системах»

Содержит пошаговые инструкции по сборке BIND версии 9.3.2 в Linux.

Приложение D «Домены высшего уровня»

Перечисление существующих в настоящее время доменов высшего уровня сети Интернет.

Приложение E «Настройка DNS-сервера и клиента BIND»

Содержит справочник по синтаксису и семантике каждого из существующих параметров настройки серверов и библиотек клиента.

## Для кого эта книга

Прежде всего эта книга предназначена для системных и сетевых администраторов, которым приходится управлять зонами и одним или несколькими DNS-серверами, но она содержит материал, который будет

интересен проектировщикам сетей, почтовым администраторам и многим другим людям. Не все главы одинаково интересны для столь разношерстной аудитории, и, конечно же, читателю нет смысла копаться во всех семнадцати главах, чтобы найти интересующий его материал. Мы надеемся, что следующая карта поможет выстроить правильный путь по главам книги.

*Системным администраторам, впервые столкнувшимся с вопросами сопровождения зон*

Следует прочесть главы 1 и 2, чтобы получить теоретическую подготовку по DNS, главу 3 – в целях получения информации о первых шагах и выборе подходящего доменного имени, главы 4 и 5 – чтобы узнать, как происходит настройка зоны «с нуля». Глава 6 объясняет, как настроить узлы для работы с новыми DNS-серверами. Затем следует обратиться к главе 7, в которой объясняется, как «подкачать» объем, добавляя серверы и данные в зону. Главы с 12 по 14 содержат описание инструментов и методов, помогающих в устранении проблем.

*Опытным администраторам*

Может быть полезно прочитать главу 6, чтобы узнать, как настраивать DNS-клиенты на различных узлах, и главу 7, чтобы получить информацию о том, как грамотно сопровождать зоны. В главе 8 содержатся инструкции, связанные с планированием роста и развития зоны, которые должны быть особенно полезны людям, занятым в администрировании больших зон. Глава 9 рассказывает о том, как стать родителем, то есть о создании поддоменов, и является учебником *этикета*, обязательным к прочтению теми, кто планирует совершить этот трудный шаг. В главе 10 рассмотрены многие новые возможности BIND версий 9.3.2 и 8.4.7. Глава 11 посвящена обеспечению безопасности DNS-серверов, и для опытных администраторов может представлять особенный интерес. Главы с 12 по 14 содержат описание инструментов и действий, которые помогут устранить возникшие проблемы; эти главы могут оказаться занимательным чтением даже для очень опытных администраторов. Глава 16 поможет администраторам осмыслить общее положение дел.

*Системным администраторам сетей, не имеющих постоянного подключения к сети Интернет*

Рекомендуется прочесть главу 5, чтобы изучить процесс настройки маршрутизации почты в таких сетях, и главы 11 и 17, которые содержат описание создания независимой инфраструктуры DNS.

*Программистам*

В целях освоения теории DNS предлагается прочесть главы 1 и 2, а затем главу 15, в которой содержится подробное рассмотрение программирования при помощи библиотечных функций BIND.

*Сетевым администраторам, которые напрямую не вовлечены в процесс сопровождения зон*

Рекомендуется прочесть главы 1 и 2 в целях освоения теории DNS, главу 12, чтобы научиться использовать *nslookup* и *dig*, а затем главу 14, чтобы узнать о способах разрешения возникающих сложностей.

*Почтовым администраторам*

Следует прочесть главы 1 и 2 в целях освоения теории DNS, главу 5, чтобы узнать, как сосуществуют DNS и электронная почта, и главу 12, в которой описаны инструменты *nslookup* и *dig*; эта глава научит извлекать информацию о маршрутизации почты из пространства доменных имен.

*Заинтересованные пользователи*

Могут прочесть главы 1 и 2 в целях освоения теории DNS, а затем любые главы по желанию!

Мы предполагаем, что читатель знаком с основами администрирования UNIX-систем, сетевым взаимодействием TCP/IP, а также программированием на уровне простых сценариев командного интерпретатора или языка Perl. При этом никаких других специальных знаний не требуется. При появлении новых терминов и понятий они насколько возможно подробно объясняются в тексте книги. По возможности мы использовали аналогии с системами UNIX (и реальным миром), чтобы облегчить читателю восприятие новых для него концепций.

## Примеры программ

Исходные тексты программ-примеров, приводимых в книге[1], доступны для загрузки по протоколу FTP по следующим адресам:

- *ftp://ftp.uu.net/published/oreilly/nutshell/dnsbind/dns.tar.Z*
- *ftp://ftp.oreilly.com/published/oreilly/nutshell/dnsbind/*

В обоих случаях извлечь файлы из архива можно командой:

```
% zcat dns.tar.Z | tar xf -
```

На системах System V необходимо использовать следующую *tar*-команду:

```
% zcat dns.tar.Z | tar xof -
```

Если команда *zcat* недоступна в системе, следует использовать отдельные команды *uncompress* и *tar*.

Если не удается получить тексты примеров напрямую по сети Интернет, но существует возможность посылать и получать сообщения элек-

[1] Примеры также доступны по адресу *http://examples.oreilly.com/dns5*.

тронной почты, можно воспользоваться службой *ftpmail*. Чтобы получить справку по использованию службы *ftpmail*, необходимо отправить сообщение на адрес *ftpmail@online.oreilly.com*. Следует оставить пустым поле темы сообщения; тело письма должно содержать единственное слово – «help».

## Как с нами связаться

Комментарии и вопросы, связанные с этой книгой, можно направлять непосредственно издателю:

O'Reilly Media, Inc.
1005 Gravenstein Highway North
Sebastopol, CA 95472
800 998-9938 (в США или Канаде)
707 829-0515 (международный/местный)
707 829-0104 (факс)

Издательством O'Reilly создана веб-страница, посвященная этой книге, на которой доступна информация о найденных ошибках и будут появляться разнообразные дополнительные сведения. Страница доступна по адресу:

*http://www.oreilly.com/catalog/dns5*

Если у вас есть технический вопрос или комментарий, связанный с этой книгой, задайте его, отправив сообщение по адресу:

*bookquestions@oreilly.com*

На веб-сайте издательства O'Reilly доступна дополнительная информация о книгах, конференциях, программном обеспечении, источниках информации и сети O'Reilly (O'Reilly Network):

*http://www.oreilly.com*

## Типографские соглашения

Использованы следующие соглашения по шрифту и формату для команд, инструментов и системных вызовов UNIX:

- Выдержки из сценариев или конфигурационных файлов оформлены моноширинным шрифтом:

```
if test -x /usr/sbin/named -a -f /etc/named.con
then
    /usr/sbin/named
fi
```

- Примеры диалоговых сеансов, отображающие ввод в командной строке и соответствующую реакцию системы, оформлены моноши-

ринным шрифтом, причем ввод пользователя отмечен **жирным** выделением:

```
% cat /var/run/named.pid
78
```

- Если команда должна вводиться суперпользователем (администратором системы, или пользователем root), она предваряется символом диеза (#):

```
# /usr/sbin/named
```

- Заменяемые элементы кода оформлены *моноширинным курсивом*.
- Имена доменов, файлов, функций, команд, названия страниц руководства UNIX, функции Windows, URL-адреса, фрагменты кода оформлены *курсивом*, если они расположены в основном тексте.

Это подсказка, предложение или совет общего характера.

Это предупреждение или предостережение.

## Использование кода примеров

Эта книга должна помогать вам в работе. Как правило, код из этой книги вы можете использовать в своих программах и документации. Наше разрешение не требуется, за исключением случаев, когда вы собираетесь воспроизвести значительный объем кода. К примеру, написание программы, использующей несколько фрагментов кода из этой книги, разрешения не требует. Продажа или распространение компакт-диска с примерами книг O'Reilly *требует* разрешения. Разрешение не требуется, если вы отвечаете на вопросы, приводя цитаты и примеры кода из этой книги. Разрешение *требуется*, если вы включаете большой объем кода примеров из этой книги в документацию к своему продукту.

Мы не настаиваем, чтобы вы ссылались на первоисточник, но будем признательны, если вы не забудете это сделать. Ссылка обычно включает название, имя автора, издательство и номер ISBN. Например: «DNS and BIND, Fifth Edition, by Cricket Liu and Paul Albitz. Copyright 2006 O'Reilly Media, Inc., 0-596-10057-4».

Если вам кажется, что вы используете примеры более вольно, чем предполагается приведенными выше примерами, или выходите за рамки свободного использования (fair use), свяжитесь с нами по адресу *permissions@oreilly.com*.

## Safari® Enabled


Если на обложке книги есть пиктограмма «Safari® Enabled», это означает, что книга доступна в сети Интернет посредством технологии O'Reilly Network Safari Bookshelf (Safari, книжная полка сети O'Reilly.)

Safari предлагает решение, превосходящее электронные книги. Это виртуальная библиотека, которая позволяет выполнять поиск в тысячах лучших технических книг, копировать примеры кода, загружать главы книг на свой компьютер и быстро находить ответы, когда требуется самая точная и свежая информация. Нашу технологию можно бесплатно опробовать по адресу *http://safari.oreilly.com*.

## Цитаты

Цитаты из Льюиса Кэррола в каждой из глав приводятся по версии 2.9 издания Millenium Fulcrum электронного текста «Алисы в Стране чудес» из библиотеки проекта Гутенберга (Project Gutenberg) и по изданию 1.7 текста «Алиса в Зазеркалье». Цитаты в главах 1, 2, 5, 6, 8 и 14 из «Алисы в Стране чудес», а цитаты в главах 3, 4, 7, 9–13, 15–17 – из «Алисы в Зазеркалье».[1]

## Благодарности

Авторы выражают благодарность Кену Стоуну (Ken Stone), Джерри Мак-Коллому (Jerry McCollom), Питеру Джеффу (Peter Jeffe), Хэлу Стерну (Hal Stern), Кристоферу Дарему (Christopher Durham), Биллу Уизнеру (Bill Wisner), Дэйву Керри (Dave Curry), Джеффу Окамото (Jeff Okamoto), Брэду Ноулзу (Brad Knowles), Роберту Эльцу (K. Robert Elz), а также Полу Викси (Paul Vixie) за их бесценный вклад в написание этой книги. Мы также хотели бы поблагодарить наших рецензентов Эрика Пирса (Eric Pearce), Джека Репенинга (Jack Repenning), Эндрю Черенсона (Andrew Cherenson), Дэна Тринкла (Dan Trinkle), Билла Лефевра (Bill LeFebvre) и Джона Секреста (John Sechrest) за их критику и предложения. Без помощи этих людей эта книга была бы совсем не такой (и была бы гораздо короче!).

За второе издание этой книги авторы выражают благодарность безупречной команде рецензентов: Дэйву Бэрру (Dave Barr), Найджелу Кэмпбеллу (Nigel Campbell), Биллу Лефевру, Майку Миллигану (Mike Milligan) и Дэну Тринклу.

[1] В русском издании цитаты даны в переводе Нины Демуровой (М.:, ПРЕССА, 1992). – *Примеч. ред.*

За третье издание книги авторы признательны команде мечты технических рецензентов: Бобу Хэлли (Bob Halley), Барри Марголину (Barry Margolin) и Полу Викси.

Долг благодарности за четвертое издание причитается Кевину Данлэпу (Kevin Dunlap), Эдварду Льюису (Edward Lewis) и Брайану Веллингтону (Brian Wellington), первоклассной команде рецензентов.

За помощь в работе над пятым изданием авторы благодарят блестящую команду технических рецензентов: Джоао Дамаса (João Damas), Мэтта Ларсона (Matt Larson) и Пола Викси (Paul Vixie), а также Сильвию Хаген (Silvia Hagen) за помощь с IPv6 в последнюю минуту.

Крикет хотел бы отдельно поблагодарить своего бывшего руководителя Рика Норденстена (Rick Nordensten), образцового современного высокопроизводительного менеджера, под присмотром которого была написана первая версия этой книги; своих соседей, которые терпели его эпизодическую раздражительность в течение многих месяцев, и конечно же свою жену Пэйдж за постоянную поддержку и за то, что она мирилась с непрекращающимся, даже во время ее сна, стуком клавиш. Что касается второго издания, Крикет хотел бы добавить слова благодарности в адрес своих бывших руководителей Регины Кершнер (Regina Kershner) и Пола Клоуда (Paul Klouda) за их поддержку работы Крикета с сетью Интернет. За помощь в работе над третьим изданием Крикет считает своим долгом поблагодарить своего партнера Мэтта Ларсона (Matt Larson), который участвовал в разработке Acme Razor; за четвертое он благодарит своих преданных пушистиков Дакоту и Энни – за их поцелуи и участие, а также замечательного Уолтера Б. (Walter B), который время от времени заглядывал в кабинет и проверял, как у папы дела. Что касается пятого издания, он должен упомянуть пополнение, замечательного малыша Джи (Baby G.), и передает благодарности друзьям и коллегам в Infoblox за их тяжелую работу и великодушную поддержку, а также за их компанию.

Пол благодарит свою жену Катерину за ее терпение, за многочисленные разборы полетов и за доказательство того, что она в свободное время может гораздо быстрее сшить стеганое одеяло, чем ее муж напишет свою половину книги.

# 2

# Как работает DNS

*– Что толку в книжке, – подумала Алиса, –*
*если в ней нет ни картинок, ни разговоров?*

Система доменных имен – это, прежде всего, база данных, содержащая информацию об узлах сети. Да, вкупе с этим вы получаете целый набор всякой всячины: чудные имена с точками, серверы, которые подключаются к сети, загадочное «пространство имен». И все же следует помнить, что в конечном итоге услуга, предоставляемая DNS, сводится к получению информации об узлах сети.

Мы уже рассмотрели некоторые важные аспекты работы DNS, включая архитектуру «клиент-сервер» и структуру базы данных. Однако мы не особенно вдавались в детали и не объясняли работу основных механизмов DNS.

В этой главе мы объясним и проиллюстрируем процессы, на которых построена работа системы доменных имен. Будет представлена терминология, которая позволит прочесть и понять оставшуюся часть книги (а также вести интеллектуальные беседы с друзьями – администраторами DNS).

Но сначала все-таки взглянем чуть внимательнее на концепции, представленные в предшествующей главе. Попробуем углубиться в детали и придать им особый ракурс.

## Пространство доменных имен

Распределенная база данных системы доменных имен индексируется по именам узлов. Каждое доменное имя является просто путем в огромном перевернутом дереве, которое носит название *пространства доменных имен*. Иерархическая структура дерева, отображенная на рис. 2.1, похожа на структуру файловой системы UNIX. Единствен-

ный корень дерева расположен наверху.[1] В файловых системах UNIX эта точка называется корневым каталогом и представлена символом «слэш» (/). В DNS же это просто «корень» («root»). Как и файловая система, дерево DNS может иметь любое количество ответвлений в любой точке пересечения, или *узле*. Глубина дерева ограничена и может достигать 127 уровней (предел, до которого вы вряд ли когда-нибудь доберетесь).

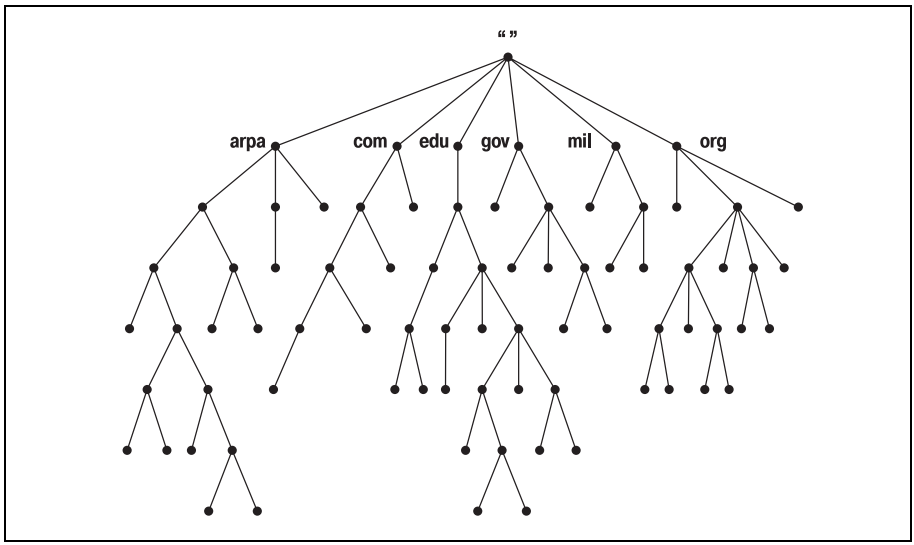



***Рис. 2.1.*** *Структура пространства имен DNS*

## Доменные имена

Каждому узлу дерева соответствует текстовая метка, длина которой не может превышать 63 символов, причем использование символа точки недопустимо. Пустая (нулевой длины) метка зарезервирована для корня. Полное *доменное имя* произвольного узла дерева – это последовательность меток в пути от этого узла до корня. Доменные имена всегда читаются от собственно узла к корню («вверх» по дереву), причем метки разделяются точкой.

Если метка корневого узла должна быть отображена в доменном имени, она записывается как символ точки, например так: «www.oreilly.com.». (На самом деле имя заканчивается точкой-разделителем и пустой меткой корневого узла.) Сама по себе метка корневого узла записывается исключительно из соображений удобства как самостоятельная точка (.). В результате некоторые программы интерпретируют имена доменов, заканчивающиеся точкой, как *абсолютные*. Абсолют-

[1] Понятно, что это дерево компьютерщика, а не ботаника.

ное доменное имя записывается относительно корня и однозначно определяет расположение узла в иерархии. Абсолютное доменное имя известно также под названием *полного доменного имени*, обозначаемого аббревиатурой *FQDN* (*fully qualified domain name*). Имена без завершающей точки иногда интерпретируются относительно некоторого доменного имени (не обязательно корневого) точно так же, как имена каталогов, не начинающиеся с символа «/» (слэш), часто интерпретируются относительно текущего каталога.

В DNS «братские» узлы, то есть узлы, имеющие общего родителя, должны иметь разные метки. Такое ограничение гарантирует, что доменное имя единственно возможным образом идентифицирует отдельный узел дерева. Это ограничение на практике не является ограничением, поскольку метки должны быть уникальными только для братских узлов одного уровня, но не для всех узлов дерева. То же ограничение существует в файловых системах UNIX: двум «единоутробным» каталогам или двум файлам в одном каталоге не могут быть присвоены одинаковые имена. Невозможно создать два узла *hobbes.pa.ca.us* в пространстве доменных имен, и невозможно создать два каталога */usr/bin* (рис. 2.2). Тем не менее можно создать пару узлов с именами *hobbes.pa.ca.us* и *hobbes.lg.ca.us* – точно так же, как можно создать пару каталогов с именами */bin* и */usr/bin*.

## Домены

*Домен* – это просто поддерево в пространстве доменных имен. Доменное имя домена идентично доменному имени узла на вершине домена. Так, к примеру, вершиной домена *purdue.edu* является узел с именем *purdue.edu* (рис. 2.3).

Аналогичным образом в файловой системе в корне каталога */usr* мы ожидаем увидеть узел с именем */usr* (рис. 2.4).

Каждое доменное имя в поддереве считается принадлежащим домену. Поскольку доменное имя может входить в несколько поддеревьев, оно также может входить в несколько доменов. К примеру, доменное имя *pa.ca.us* входит в домен *ca.us* и при этом является также частью домена *us* (рис 2.5).

Так что теоретически домен – это просто сегмент пространства доменных имен. Но если домен состоит только из доменных имен и других доменов, то где все узлы? Ведь домены-то – это группы узлов, верно?

Узлы сети, разумеется, присутствуют, и представлены они доменными именами. Следует помнить, что доменные имена являются просто указателями в базе данных DNS. «Узлы» – это доменные имена, которые указывают на информацию по каждому конкретному узлу. Домен содержит все узлы сети, доменные имена которых в него входят. Узлы сети связаны *логически*, зачастую по географическому или организа-

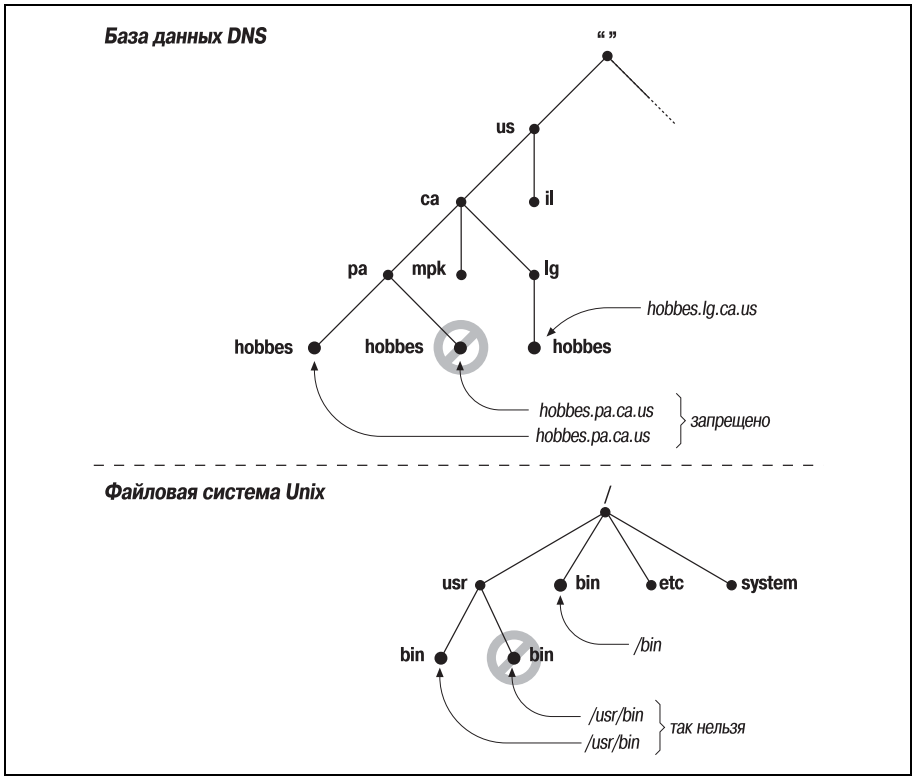



***Рис. 2.2.*** *Обеспечение уникальности доменных имен и путевых имен в файловых системах UNIX*

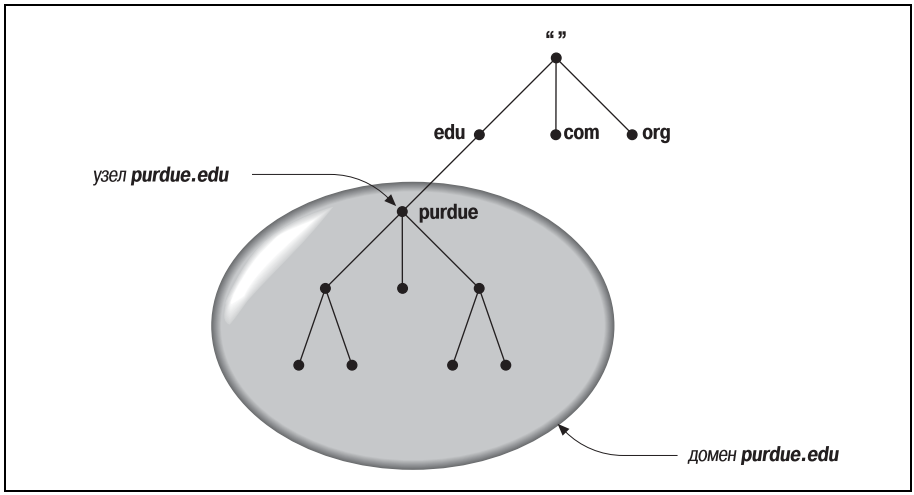



***Рис. 2.3.*** *Домен purdue.edu*

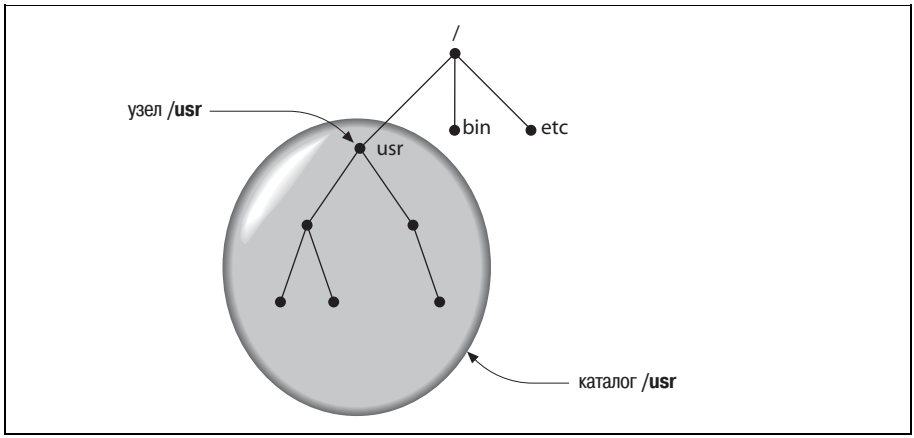



***Рис. 2.4.*** *Каталог /usr*

ционному признаку, и совсем необязательно – сетью, адресом или типом используемого оборудования. Десяток узлов, входящих в разные сети, возможно даже расположенных в разных странах, может принадлежать одному-единственному домену.

Предостережение: не стоит путать домены DNS с доменами в службе NIS, Network Information Service от Sun. Несмотря на то, что домен NIS – это тоже группа узлов, а доменные имена в обеих службах имеют сходную организацию, концептуальные различия достаточно велики. В NIS используется иерархическая организация имен, но иерархия на этом и заканчивается: узлы сети, входящие в один домен NIS, разделяют определенную информацию об узлах и пользователях, но не могут опрашивать пространство имен NIS с целью поиска информации в других доменах NIS. Домены NT, обеспечивающие управление доступом

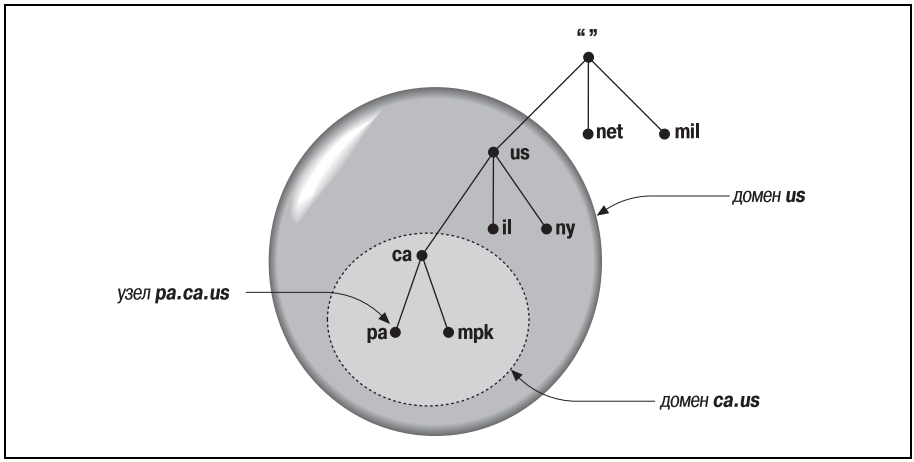



***Рис. 2.5.*** *Узел, входящий в несколько доменов*

и службами безопасности, также не имеют никакого отношения к доменам DNS. Однако домены Active Directory близко связаны с доменами DNS. Мы обсудим эту связь в главе 17 «Обо всем понемногу».

Доменные имена, соответствующие листьям дерева, как правило, относятся к отдельным узлам сети и могут указывать на сетевые адреса, информацию об оборудовании и о маршрутизации почты. Доменные имена внутри дерева могут идентифицировать отдельные узлы, а *также* могут указывать на информацию об этом домене. Имена доменов внутри дерева не привязаны жестко к тому или другому варианту. Они могут представлять как домен (имени которого соответствуют), так и отдельный узел в сети. Так, *hp.com* является именем домена компании Hewlett-Packard и доменным именем, указывающим на узлы, на которых расположен главный веб-сервер Hewlett-Packard.

Тип информации, получаемой при использовании доменного имени, зависит от контекста применения имени. Посылка почтовых сообщений кому-то в домен *hp.com* приводит к получению информации о маршрутизации почты, открытие сеанса *ssh*-связи с этим доменом приводит к поиску информации об узле (на рис. 2.6, например, это IP-адрес узла *hp.com*).

Домен может содержать несколько поддеревьев, которые носят название *поддоменов*.[1]

Самый простой способ выяснить, является ли домен поддоменом другого домена, – сравнить их доменные имена. Доменное имя поддомена заканчивается доменным именем родительского домена. К примеру, домен *la.tyrell.com* должен являться поддоменом домена *tyrell.com*, по-

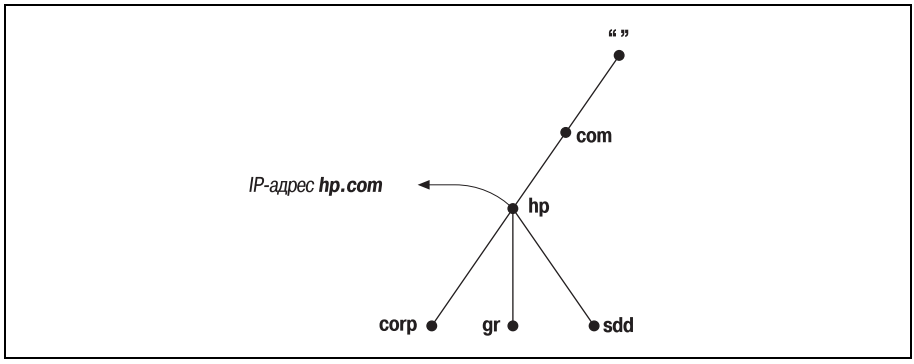



***Рис. 2.6.*** *Внутренний узел дерева, связанный как с информацией о конкретном узле сети, так и с иерархической*

[1] Термины *домен* и *поддомен* в документации по DNS часто взаимозаменяемы. В этой книге термин «поддомен» используется как относительный: домен является поддоменом другого домена, если корень поддомена принадлежит включающему домену.

скольку имя *la.tyrell.com* заканчивается именем *tyrell.com*. Этот домен является также поддоменом домена *com*, как и домен *tyrell.com*.

Помимо относительных степеней в идентификации доменов в качестве входящих в другие домены, используется также *уровневая* классификация доменов. В списках рассылки и конференциях Usenet можно встретить термины *домен высшего уровня* и *домен второго уровня*. Эти термины просто определяют положение домена в пространстве доменных имен:

- Домен высшего уровня в качестве непосредственного родителя имеет корневой домен.
- Домен первого уровня в качестве непосредственного родителя *также* имеет корневой домен (то есть он является доменом высшего уровня).
- Домен второго уровня в качестве непосредственного родителя имеет домен первого уровня и т. д.

## Записи ресурсов

Информация, связанная с доменными именами содержится в *записях ресурсов* (RRs, resource records).[1] Записи разделяются на классы, каждый из которых определяет тип сети или программного обеспечения. В настоящее время существуют классы для интернет-сетей (на основе семейства протоколов TCP/IP), сетей на основе протоколов Chaosnet, а также сетей, которые построены на основе программного обеспечения Hesiod. (Chaosnet – старая сеть, имеющая преимущественно историческое значение).

Популярность класса интернет-сетей значительно превосходит популярность остальных классов. (Мы не уверены, что где-то до сих пор используется класс Chaosnet, а использование класса Hesiod в основном ограничивается пределами Массачусетского технологического института – MIT). В этой книге мы сосредоточимся на классе интернет-сетей.

В пределах класса записи делятся на типы, которые соответствуют различным видам данных, хранимых в пространстве доменных имен. В различных классах определяются различные типы записей, хотя некоторые типы могут являться общими для нескольких классов. Так, практически в каждом классе определен тип *адрес*. Каждый тип записи в конкретном классе определяет формат, который должны соблюдать все RR-записи, принадлежащие этому классу и имеющие данный тип.

Не беспокойтесь, если информация кажется излишне схематичной: записи класса интернет-сетей будут более подробно рассмотрены поз-

[1] В дальнейшем употребляется выражение RR-запись, или просто RR. – *Примеч. ред.*

же. Наиболее часто используемые RR-записи описаны в главе 4, а более полный перечень приводится в приложении А.

# Пространство доменных имен сети Интернет

До сих пор мы говорили о теоретической структуре пространства доменных имен и о том, какого сорта данные могут в нем содержаться, и даже обозначили – в наших (порой выдуманных) примерах – типы имен, которые можно встретить в этом пространстве. Но это ничем не поможет в расшифровке доменных имен, которые ежедневно встречаются пользователям в сети Интернет.

Система доменных имен довольно либеральна в отношении меток, составляющих доменные имена, и не определяет *конкретные* значения для меток определенного уровня пространства имен. Если администратор управляет сегментом пространства имен, он и определяет семантику доменных имен, входящих в сегмент. Черт возьми, администратор может присвоить поддоменам в качестве имен буквы алфавита от A до Z, и никто его не остановит (хотя сильно будут советовать этого не делать).

При этом существующее пространство доменных имен сети Интернет обладает некоторой сложившейся структурой. Это в особенности касается доменов верхних уровней, доменные имена в которых подчиняются определенным традициям (которые не являются правилами, поскольку их можно нарушать, и так не раз бывало). Эти традиции вносят в доменные имена некоторую упорядоченность. Понимание традиций – это большое подспорье для попыток расшифровки доменных имен.

## Домены высшего уровня

Изначально домены высшего уровня делили пространство имен сети Интернет на семь доменов:

*com*

Коммерческие организации, такие как Hewlett-Packard (*hp.com*), Sun Microsystems (*sun.com*) или IBM (*ibm.com*).

*edu*

Образовательные организации, такие как Калифорнийский университет Беркли (*berkeley.edu*) и университет Пердью (*purdue.edu*).

*gov*

Правительственные организации, такие как NASA (*nasa.gov*) и Национальный научный фонд (*nsf.gov*).

*mil*

Военные организации, такие как армия (*army.mil*) и флот (*navy.mil*) США.

*net*

В прошлом организации, обеспечивающие работу сетевой инфраструктуры, такие как NSFNET (*nsf.net*) и UUNET (*uu.net*). В 1996 году домен *net*, как и *com*, стал доступен для всех коммерческих организаций.

*org*

В прошлом некоммерческие организации, такие как Фонд электронной границы (Electronic Frontier Foundation) (*eff.org*). Как и в случае с доменом *net*, ограничения были сняты в 1996 году.

*int*

Международные организации, такие как NATO (*nato.int*).

Существовал еще один домен высшего уровня, который назывался *arpa* и использовался в процессе перехода сети ARPAnet от таблиц узлов к системе доменных имен. Изначально все узлы ARPAnet имели имена, принадлежавшие домену *arpa*, так что их было несложно отличать. Позже они разбрелись по различным поддоменам организационных доменов высшего уровня. При этом домен *arpa* все еще используется, и чуть позже мы расскажем, как именно.

Можно заметить некоторую националистическую предрасположенность в наших примерах: прежде всего речь идет об организациях США. Это легче понять – и простить, – если вспомнить, что сеть Интернет началась с сети ARPAnet – исследовательского проекта, который финансировался правительством США. Никто и не предполагал, что создание ARPAnet увенчается подобным успехом и что этот успех в итоге приведет к созданию международной сети Интернет.

В наши дни эти домены называются *родовыми доменами высшего уровня* (generic top-level domains, или gTLDs). Слово «родовые» призвано отличать их от доменов высшего уровня, разделенных по географическому признаку.

## Географические домены высшего уровня

Чтобы справиться с потребностями быстро растущих сегментов сети Интернет, расположенных во многих странах мира, пришлось пойти на определенные компромиссы, связанные с пространством доменных имен Интернета. Было решено не придерживаться схемы организационного деления доменов высшего уровня, но разрешить использование географических обозначений. Новые домены высшего уровня были зарезервированы (но не во всех случаях созданы) для каждой страны. Эти доменные суффиксы соответствуют существующему международному стандарту ISO 3166.[1] Стандарт ISO 3166 определяет официальные двухбуквенные сокращения для каждой страны мира. Текущий список доменов высшего уровня приведен в приложении D.

### Новые домены высшего уровня

В конце 2000 года организация, ответственная за управление системой доменных имен сети Интернет – Internet Corporation for Assigned Names and Numbers (ICANN), – создала семь новых родовых доменов высшего уровня, чтобы приспособить иерархию к взрывному росту сети Интернет и удовлетворить потребность в доменном «пространстве». Некоторые из этих доменов были действительно родовыми, как *com*, *net* и *org*, тогда как другие больше походили на домены, вроде *gov* и *mil*, и были зарезервированы для использования конкретными (иногда удивительно немногочисленными) группами людей. ICANN называет эту последнюю разновидность доменов высшего уровня *спонсируемыми доменами высшего уровня* (*sTLDs*, *sponsored top-level domains*), а классический вариант – *неспонсируемыми родовыми доменами высшего уровня* (*unsponsored gTLDs*). Спонсируемый домен высшего уровня имеет устав, определяющий его назначение, а также организацию-спонсора, устанавливающую правила управления этим доменом и контролирующую работу с ним по поручению ICANN.

Вот новые родовые домены высшего уровня:

*aero*
: Спонсируемый; предназначается для авиационной индустрии.

*biz*
: Родовой.

*coop*
: Спонсируемый; для кооперативных обществ.

*info*
: Родовой.

*museum*
: Спонсируемый; для музеев.

*name*
: Родовой; для частных лиц.

*pro*
: Родовой; для специалистов.

Не так давно, в начале 2005 года, ICANN утвердила дополнительные спонсируемые домены высшего уровня: *jobs*, который предназначается для кадровой индустрии, и *travel* – для индустрии туризма. В настоящее время рассматривается еще ряд спонсируемых доменов, в частности *cat* – для культурно-языковой группы каталонцев, *mobi* – для

[1] За исключением Великобритании. В соответствии со стандартом ISO 3166 и традициями Интернета доменный суффикс высшего уровня для Великобритании должен быть *gb*. На деле же большинство организаций Великобритании и Северной Ирландии (то есть Соединенного Королевства) используют суффикс *uk*. А еще они ездят по неправильной стороне дороги.

мобильных устройств, а также *post* – для почтового сообщества в киберпространстве. К настоящему моменту делегирование от корня получил только домен *mobi*. Страница организации ICANN в Интернете доступна по адресу *http://www.icann.org*.

## По нисходящей

В пределах упомянутых доменов верхних уровней существующие традиции и степень их соблюдения начинают варьироваться. Некоторые из доменов высшего уровня, упомянутых в стандарте ISO 3166, в общем и целом следуют исходной организационной схеме США. К примеру, в домен высшего уровня Австралии, *au*, входят такие поддомены, как *edu.au* и *com.au*. Некоторые из прочих доменов ISO 3166 следуют примеру домена *uk* и порождают поддомены организационного деления, например *co.uk*, для корпоративного использования, а скажем *ac.uk* – для нужд академического сообщества. Тем не менее в большинстве случаев географические домены высшего уровня имеют организационное деление.

Однако это не относилось изначально к домену высшего уровня *us*. В домен *us* входило 50 поддоменов, которые соответствовали (попробуйте угадать!) пятидесяти штатам.[1] Имя каждого из этих поддоменов соответствует стандартному двухбуквенному сокращению названия штата, именно эти сокращения стандартизированы почтовой службой США. В пределах каждого поддомена деление было в основном такое же, географическое: большинство поддоменов соответствовало отдельным городам. В пределах поддомена города все прочие поддомены обычно соответствовали отдельным узлам.

Но, как и многие другие правила иерархии доменных имен, эта структура ушла в прошлое в 2002 году, когда управлением доменом *us* занялась новая компания, Neustar. Теперь домен *us* наравне с доменами *com* и *net* открыт для всех.

## Чтение доменных имен

Теперь, познакомившись со свойствами имен доменов высшего уровня и структурой пространства доменных имен, читатели, вероятно, смогут гораздо быстрее определять кроющийся в именах доменов смысл. Попрактикуемся на следующих примерах:

*lithium.cchem.berkeley.edu*

Здесь у читателей фора, поскольку мы уже упоминали, что *berkeley.edu* – это домен университета Беркли. (Даже если бы мы не рассказали заранее, можно было бы догадаться, что имя, вероятнее всего, принадлежит одному из университетов США, поскольку принад-

[1] В действительности поддоменов в домене *us* чуть больше: один для Вашингтона (округ Колумбия), один для острова Гуам и т. д.

лежит домену высшего уровня *edu*.) *cchem* – поддомен *berkeley.edu*, принадлежащий факультету химии. И наконец, *lithium* (литий) – имя конкретного узла в домене, помимо которого в поддомене есть, вероятно, еще около ста узлов, если под каждый химический элемент выделен отдельный узел.

*winnie.corp.hp.com*

Этот пример посложнее, но ненамного. Домен *hp.com*, по всей видимости, принадлежит компании Hewlett-Packard (кстати, и об этом мы уже говорили). Поддомен *corp,* вне всякого сомнения, подразумевает корпоративную штаб-квартиру. А *winnie* – вероятно, просто неудачно выбранное имя узла.

*fernwood.mpk.ca.us*

В этом примере придется использовать знания о структуре домена *us*. *ca.us*, домена штата Калифорния, но *mpk* может означать что угодно. В данном случае очень трудно догадаться, что это доменное имя Менло-Парка, если не знать географию района Сан-Франциско. (Нет, это не тот самый Менло-Парк, в котором жил Эдисон, – тот находится в Нью-Джерси.)

*daphne.ch.apollo.hp.com*

Этот пример мы приводим для того, чтобы у читателей не появилось ложного ощущения, будто все доменные имена состоят из четырех меток. *apollo.hp.com* – бывший поддомен компании Apollo Computer, принадлежащий домену *hp.com*. (Когда компания HP приобрела компанию Apollo, то не забыла купить и интернет-домен Apollo, *apollo.com*, который был переименован в *apollo.hp.com*.) *ch.apollo.hp.com* – отделение Apollo в Челмсфорде (штат Массачусетс). *daphne* – просто узел в Челмсфорде.

## Делегирование

Надеемся, читатели еще не забыли, что одной из основных целей разработки системы доменных имен была децентрализация администрирования? Эта цель достигается путем *делегирования*. Делегирование доменов во многом схоже с распределением рабочих задач. Начальник может разделить крупный проект на небольшие задачи и делегировать ответственность за каждую из них разным подчиненным.

Аналогичным образом организация, сопровождающая домен, может разделить его на несколько поддоменов, каждый из которых может быть *делегирован* какой-либо другой организации. Организация, получившая домен таким путем, несет ответственность за работу с данными этого поддомена. Она может свободно изменять эти данные, разделять поддомен на несколько более мелких поддоменов, а те, в свою очередь, снова делегировать. Родительский домен сохраняет только указатели на источники данных для поддоменов в целях перенаправ-

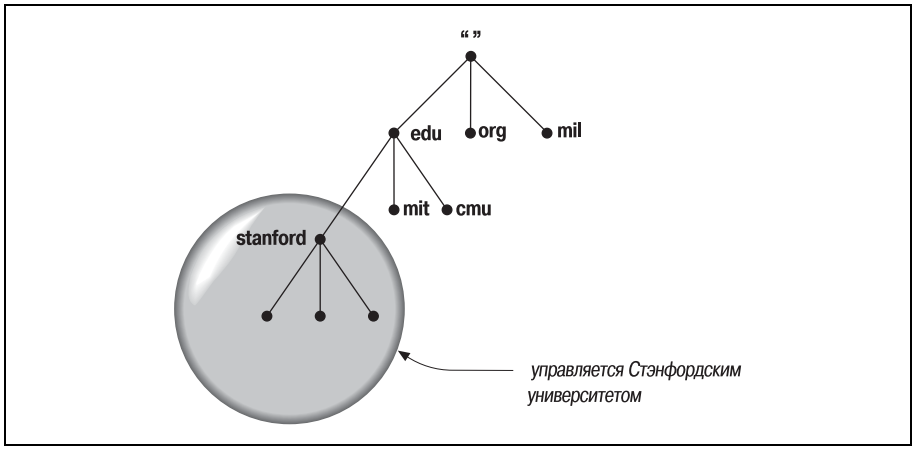



***Рис. 2.7.*** *stanford.edu делегирован Стэнфордскому университету*

ления запросов по соответствующим адресам. К примеру, домен *stanford.edu* делегирован тем ребятам в Стэнфорде, которые управляют университетскими сетями (рис. 2.7.)

Не все организации делегируют домены целиком, как не каждый начальник делегирует всю свою работу. Домен может иметь несколько делегированных поддоменов, но также включать узлы, которые не принадлежат этим поддоменам. К примеру, корпорация Acme (она снабжает одного известного койота различными приспособлениями) имеет отделение в городе Рокавей, а ее штаб-квартира расположена в Каламазу, поэтому могут существовать поддомены *rockaway.acme.com* и *kalamazoo.acme.com*. Однако несколько узлов, соответствующих отделам сбыта Acme, разбросаны по всей стране и скорее принадлежат собственно домену *acme.com*, нежели какому-либо из поддоменов.[1]

Позже мы расскажем, как создавать и делегировать поддомены. Сейчас же важно понять, что термин *делегирование* относится к передаче ответственности за поддомен сторонней организации.

## DNS–серверы и зоны

Программы, владеющие информацией о пространстве доменных имен, называются *DNS-серверами*. DNS-серверы обычно обладают полной информацией по определенным сегментам (или *зонам*) пространства доменных имен, которая загружается из файла либо может быть полу-

[1] «ACME Co.» – корпорация из мультипликационного сериала «Bugs Bunny & Roadrunner»; в действительности не существует. Домен *acme.com* на самом деле принадлежит организации, разрабатывающей программное обеспечение для UNIX-систем. – *Примеч. ред.*

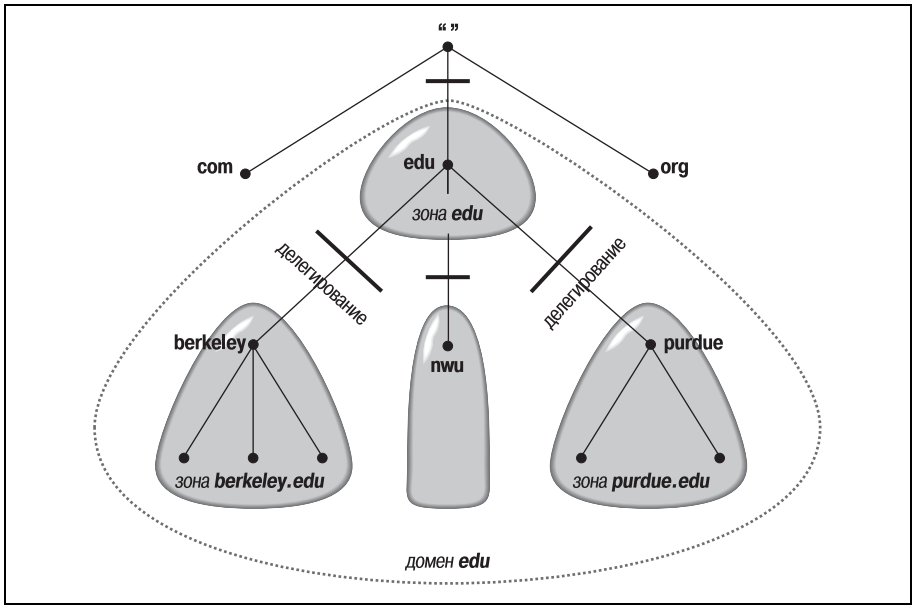



***Рис. 2.8.*** *Разбивка домена edu на зоны*

чена от других серверов имен. В таких случаях говорят, что сервер имен является *авторитетным* для конкретной зоны. DNS-серверы могут быть авторитетными также и для нескольких зон.

Разница между зоной и доменом важна, но не очень заметна. Все домены высшего уровня и домены уровней от второго и ниже, такие как *berkeley.edu* и *hp.com*, разбиваются на более мелкие, легко управляемые единицы путем делегирования. Эти единицы называются зонами. Домен *edu* (рис. 2.8) разделен на много зон, включая зону *berkeley.edu*, зону *purdue.edu* и зону *nwu.edu*. На вершине домена существует также зона *edu*. Естественно, что ребята, управляющие *edu*, разбивают домен *edu* на более мелкие единицы: в противном случае им пришлось бы самим сопровождать поддомен *berkeley.edu*. Гораздо более разумно делегировать *berkeley.edu* Беркли. Что же остается тем, кто управляет *edu*? Зона *edu*, которая содержит преимущественно информацию о делегировании для поддоменов, входящих в *edu*.

Поддомен *berkeley.edu*, в свою очередь, разбивается на несколько зон путем делегирования (рис. 2.9). Делегированные поддомены носят имена *cc*, *cs*, *ce*, *me* и т. д. Каждый из этих поддоменов делегируется ряду серверов имен, некоторые из которых являются компетентными и для *berkeley.edu*. Тем не менее зоны живут самостоятельно и могут иметь совершенно отдельный набор авторитетных DNS-серверов.

Зона включает все доменные имена, которые включает домен с тем же именем, за исключением доменных имен, принадлежащих к делегированным поддоменам. Так, домен высшего уровня *ca* (Канада) вклю-

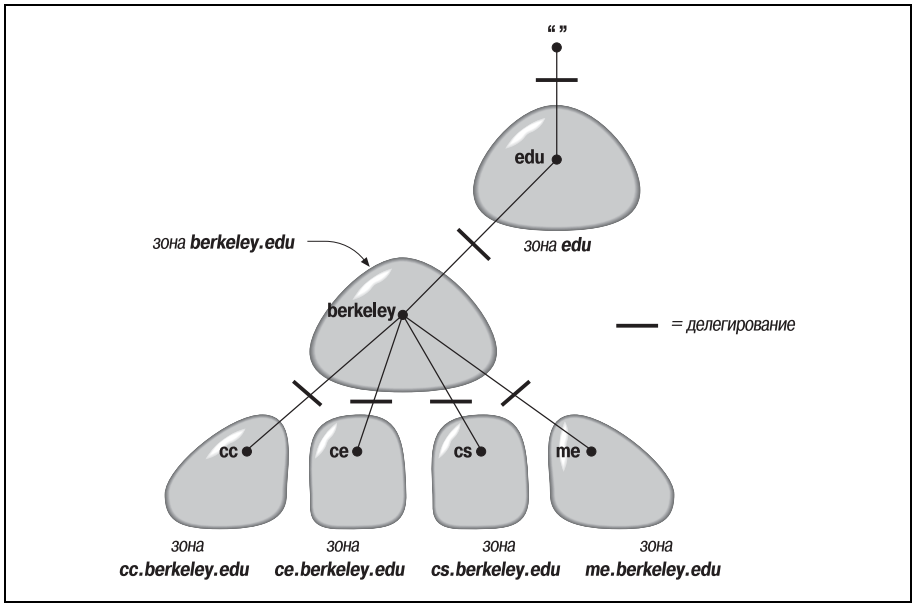



***Рис. 2.9.*** *Разбивка домена berkeley.edu на зоны*

чает поддомены *ab.ca*, *on.ca* и *qc.ca*, относящиеся к провинциям Альберта, Онтарио и Квебек. Ответственность за сопровождение данных в поддоменах *ab.ca*, *on.ca* и *qc.ca* может быть делегирована серверам имен в каждой из этих провинций. Домен *ca* содержит всю информацию для *ca*, а также всю информацию в *ab.ca*, *on.ca* и *qc.ca*. Однако зона *ca* содержит данные только для *ca* (рис. 2.10), и эти данные, вероят-

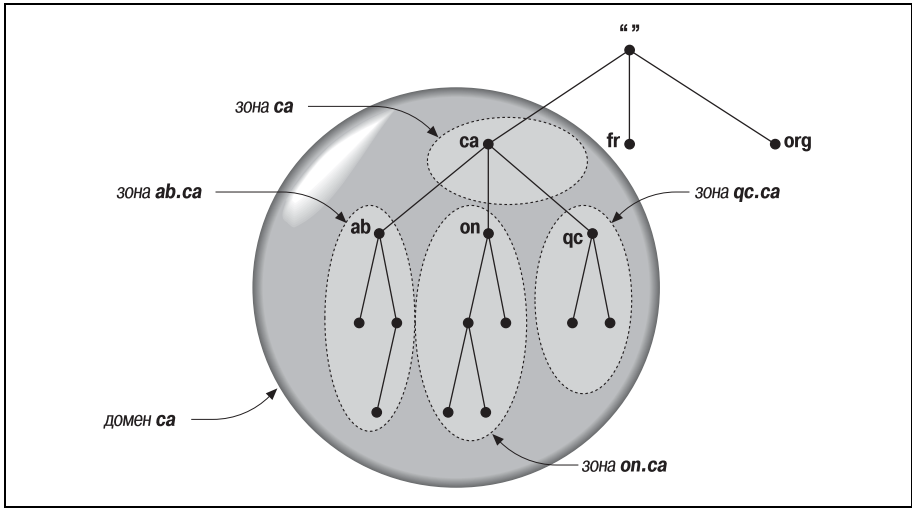



***Рис. 2.10.*** *Домен ca...*

но, в основном являются указателями на делегированные поддомены. *ab.ca*, *on.ca* и *qc.ca* являются отдельными от *ca* зонами.

Зона также содержит имена доменов и данные всех поддоменов, которые не были делегированы. Так, поддомены *bc.ca* и *sk.ca* (Британская Колумбия и Саскачеван) домена *ca* могут существовать, но могут не быть делегированы. (Возможно, власти провинций Британская Колумбия и Саскачеван еще не готовы управлять собственными зонами, а люди, ответственные за зону высшего уровня *ca*, желают сохранить согласованность пространства имен и немедленно создать поддомены для всех провинций Канады.) В таком случае зона *ca* будет иметь неровную нижнюю границу, включая *bc.ca* и *sk.ca*, но не другие поддомены *ca* (рис. 2.11).

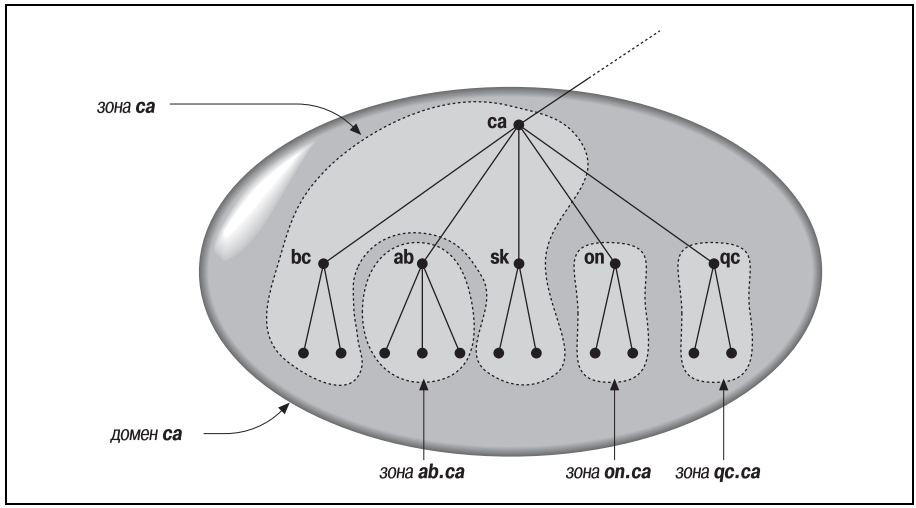



***Рис. 2.11. ...и зона ca***

Теперь становится понятно, почему объектом, загружаемым DNS-сервером, является зона, а не домен: домен может содержать больше информации, чем требуется для работы сервера.[1] Домен может содержать данные, делегированные другим серверам. Поскольку зоны ограничиваются делегированием, они никогда не включают делегированные данные.

Обычно в начале существования у домена еще нет никаких поддоменов. В таком случае, поскольку делегирования не происходит, домен и зона содержат одинаковые данные.

[1] Представьте, что получится, если корневой сервер имен загрузит корневой домен вместо зоны: он загрузит информацию обо всем пространстве имен!

## Делегирование поддоменов

Даже в случае отсутствия насущной необходимости делегировать части домена полезно более широко понимать процедуру делегирования поддомена. Делегирование теоретически включает передачу ответственности за какую-то часть домена другой организации. На деле же происходит назначение различных DNS-серверов в качестве авторитетных в делегируемых поддоменах (мы не случайно употребляем здесь множественное число – именно серверов).

Хранимая зоной информация после делегирования включает уже не информацию по делегированному поддомену, а информацию о серверах имен, являющихся для этого поддомена авторитетными. В таком случае, если у DNS-сервера родительского домена запрашиваются данные для поддомена, в ответ предоставляется список DNS-серверов, которые обладают соответствующей информацией.

## Разновидности DNS–серверов

Спецификация DNS определяет два типа DNS-серверов: первичный мастер-сервер (primary master) и дополнительный, или вторичный мастер-сервер (secondary master). *Первичный мастер-сервер* производит загрузку данных для зоны из файла на машине-сервере. *Вторичный мастер-сервер* получает данные зоны от другого DNS-сервера, который является авторитетным для этой зоны и называется его *мастером* (*master server*). Довольно часто мастер-сервер является первичным мастером для зоны, но не обязательно: вторичный мастер-сервер может получать зональные данные и от другого вторичного. Когда запускается вторичный сервер, он устанавливает связь с мастером и при необходимости получает зональные данные. Этот процесс называется *передачей*, или *трансфером зоны* (*zone transfer*). В наше время предпочтительным термином для вторичного мастер-сервера имен является *slave* (*подчиненный, ведомый*)[1], хотя многие люди (и некоторые программы, в частности, консоль Microsoft DNS) все еще пользуются прежним термином.

Как первичный, так и вторичный мастер-серверы зоны являются для этой зоны авторитетными. Несмотря на несколько уничижительное название, вторичные (slave) серверы не являются серверами второго сорта. Эти два типа серверов предусмотрены в DNS с целью облегчения администрирования. После создания зональных данных и установки первичного мастера можно не беспокоиться о копировании данных с узла на узел при создании дополнительных DNS-серверов зоны. Достаточно просто установить вторичные DNS-серверы, которые будут получать данные от первичного мастер-сервера для этой зоны. После это-

[1] В переводе книги в большинстве случаев употребляется термин «вторичный», т. к. это устоявшийся профессиональный язык. – *Примеч. перев.*

го передача и получение зональных данных будут происходить по необходимости автоматически.

Вторичные серверы имеют большое значение, поскольку наличие нескольких авторитетных DNS-серверов для каждой зоны – идея очень правильная. Понадобится больше одного сервера, чтобы обеспечить избыточность, распределить нагрузку, гарантировать, что каждому из узлов зоны легко доступен хотя бы один DNS-сервер. Использование вторичных серверов делает такую архитектуру выгодной и в плане управления.

Однако было бы несколько неточно называть *конкретный* DNS-сервер первичным или вторичным. Ранее мы говорили, что сервер может являться авторитетным для более чем одной зоны. Точно так же DNS-сервер может являться первичным для одной зоны и вторичным – для другой. Но в большинстве случаев сервер имен является либо первичным, либо вторичным для большинства загружаемых им зон. Поэтому, когда мы называем конкретный сервер имен первичным или вторичным, то имеем в виду, что он является таковым для *большинства* зон, которые входят в сферу его компетенции.

## Файлы данных зоны

Файлы, из которых первичные DNS-серверы производят чтение зональных данных, называются, и вполне логично, *файлами данных зоны*. Мы достаточно часто называем их *файлами данных*. Вторичные DNS-серверы также могут загружать зональные данные из файлов. Обычно вторичные серверы настраиваются таким образом, что при каждом получении зональных данных с основного сервера происходит сохранение резервной копии полученной информации в файлах данных. При последующем перезапуске или сбое происходит сначала чтение файлов с резервных копий в целях определения актуальности зональных данных. Это позволяет устранить необходимость в передаче зональных данных, если они не изменились, и обеспечивает вторичный DNS-сервер рабочим набором данных в случае недоступности первичного сервера.

Файлы данных содержат RR-записи, описывающие зону. RR-записи описывают все узлы сети в зоне и помечают делегирование поддоменов. В BIND также существуют специальные директивы для включения содержимого других файлов данных в файл данных зоны аналогично директиве *#include* языка программирования C.

# Клиенты DNS

Клиенты DNS (resolvers) позволяют осуществлять доступ к DNS-серверам. Программы, которым требуется информация из пространства доменных имен, используют DNS-клиент, решающий следующие задачи:

- Опрашивание DNS-серверов
- Интерпретация полученных ответов (RR-записей или сообщений об ошибках)
- Возврат информации в программу, которая ее запросила

В пакете BIND клиент – это просто набор библиотечных процедур, которые вызываются из таких программ, как *ssh* и *ftp*. Клиент – это даже не отдельный процесс. Клиент практически во всем полагается на DNS-серверы: его хватает ровно на то, чтобы создать запрос, отправить его и ждать ответа, затем повторно послать запрос, если ответ не получен; и это практически все, на что он способен. Большая часть работы, связанной с поиском ответа на вопрос, ложится на сервер имен. Спецификация DNS называет этот тип анализатора *примитивным* (*stub resolver*).

В других реализациях системы DNS существуют более совершенные клиенты, способные делать гораздо более сложные вещи, например следовать перенаправляющим ответам и находить DNS-серверы, являющиеся авторитетными для определенной зоны.

# Разрешение имен

DNS-серверы исключительно хорошо умеют получать данные из пространства доменных имен. Что неудивительно, учитывая ограниченную интеллектуальность большинства DNS-клиентов. Серверы могут не только поставлять информацию по зонам, для которых являются авторитетными, но и производить поиск в пространстве доменных имен, получая данные зон, в их компетенцию не входящих. Этот процесс называется *разрешением имен* или просто *разрешением*.

Поскольку пространство имен имеет структуру перевернутого дерева, серверу нужна лишь частичка информации, чтобы пробраться к любому узлу дерева: доменные имена и адреса корневых DNS-серверов (разве это больше, чем частичка?). Сервер может обратиться к корневому DNS-серверу с запросом по любому доменному имени пространства доменных имен, после чего получает руководящие указания по продолжению поиска.

## Корневые DNS-серверы

Корневые серверы обладают информацией об авторитетных DNS-серверах для каждой из зон высшего уровня. (На деле некоторые корневые DNS-серверы одновременно являются авторитетными для некоторых родовых зон высшего уровня.) Получив запрос по любому доменному имени, корневой сервер может вернуть, по меньшей мере, список имен и адресов DNS-серверов, авторитетных для зоны высшего уровня, в иерархии которой расположен домен. А DNS-серверы высшего уровня могут вернуть список авторитетных серверов для зоны второго уровня, в иерархии которой расположен домен. Каждый из опрашива-

емых серверов либо возвращает информацию о том, как подобраться «поближе» к искомому ответу, либо сразу конечный ответ.

Корневые серверы, очевидно, имеют очень большое значение для процесса разрешения имен. Поэтому в DNS существуют механизмы (скажем, кэширование, которое мы обсудим чуть позже), позволяющие разгрузить корневые серверы. Но в отсутствие другой информации разрешение должно начинаться с корневых DNS-серверов. И это делает корневые серверы ключевым элементом работы DNS. Недоступность всех корневых DNS-серверов сети Интернет в течение длительного времени привела бы к невозможности разрешения имен в сети. Чтобы исключить подобные ситуации, в сети Интернет существует (на момент написания этой книги) тринадцать корневых серверов, расположенных в различных частях сети[1]. Один из них находится в сети PSINet, коммерческой информационной магистрали Интернета, один в научной сети NASA, два в Европе, а один в Японии.

Корневые серверы находятся под постоянной нагрузкой, поскольку на них направлено огромное число запросов; даже с учетом того, что серверов тринадцать, трафик для каждого из них очень велик. Недавний неофициальный опрос администраторов корневых DNS-серверов показал, что некоторые из серверов обрабатывают десятки тысяч запросов в секунду.

Несмотря на такую нагрузку, разрешение имен в сети Интернет работает вполне надежно. На рис. 2.12 показан процесс разрешения для адреса реального узла в реальном домене с учетом того, как производится обход дерева пространства доменных имен.

Локальный DNS-сервер запрашивает адрес *girigiri.gbrmpa.gov.au* у корневого сервера и получает ссылку на DNS-серверы домена *au*. Локальный сервер повторяет запрос, отправляя его одному из DNS-серверов *au*, и получает ссылку на серверы *gov.au*. DNS-сервер *gov.au* отсылает локальный DNS-сервер к серверам *gbrmpa.gov.au*. И наконец, локальный DNS-сервер запрашивает DNS-сервер *gbrmpa.gov.au* и получает искомый адрес.

## Рекурсия

Читатели, вероятно, заметили разницу в количестве работы, выполняемой разными DNS-серверами в последнем примере. Четыре сервера, получая запросы, просто возвращали наилучший из уже имевшихся у них ответов – как правило, ссылку на другой DNS-сервер. Эти серве-

[1] На деле эти 13 «логических» корневых DNS-серверов представлены гораздо большим числом физических машин. Для большинства корневых серверов применяется балансировка нагрузки с единым IP-адресом, совместное использование одного адреса группой распределенных серверов либо сочетание этих методов.

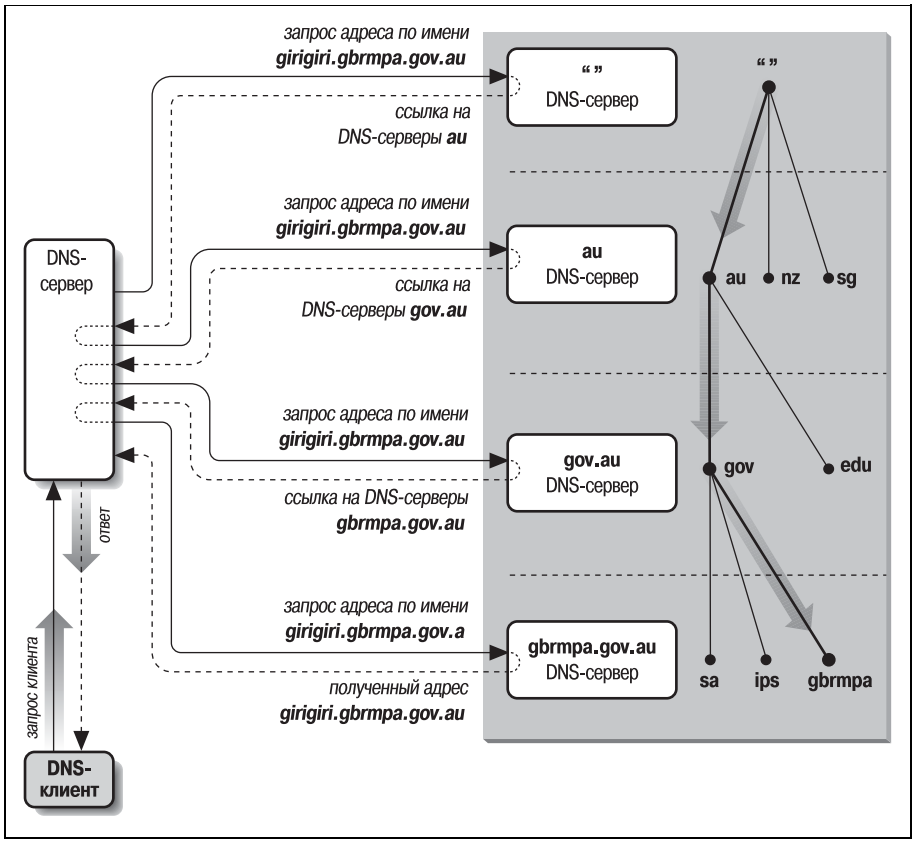



***Рис. 2.12.*** *Разрешение имени girigiri.gbrmpa.gov.au в сети Интернет*

ры не выполняли собственные запросы с целью поиска запрошенной информации. Но один DNS-сервер – тот, к которому обратился клиент, – следовал по предлагаемым ссылкам, пока не получил окончательный ответ.

Почему локальный DNS-сервер попросту не перенаправил клиент к другому серверу? Потому что примитивный клиент не способен следовать по таким ссылкам. Каким образом сервер понял, что отвечать клиенту ссылкой – пустая трата времени? Очень просто: клиент сделал *рекурсивный запрос*. Существуют запросы двух видов: *рекурсивные* и *итеративные* (или *нерекурсивные*). Рекурсивные запросы возлагают большую часть работы по разрешению имени на единственный DNS-сервер. *Рекурсия*, или *рекурсивное разрешение имен*, – это название последовательности действий DNS-сервера при получении им рекурсивного запроса. Сервер DNS повторяет какую-то базовую последовательность действий (посылает запрос удаленному серверу и следует по ссылкам), пока не получит ответ, то есть действует аналогично рекурсивному алгоритму программирования.

*Итерация*, или *итеративное разрешение имен*, – это название последовательности действий DNS-сервера при получении им итеративного запроса.

В случае рекурсии клиент посылает серверу рекурсивный запрос информации для определенного доменного имени. Сервер в таком случае обязан вернуть ответ на запрос (запрошенную информацию) либо ошибку, сообщающую, что данные указанного типа не существуют либо не существует доменное имя.[1] Сервер не может вернуть ссылку на другой DNS-сервер, если запрос рекурсивный.

Если DNS-сервер, получивший запрос, не авторитетен для запрашиваемой информации, ему придется опрашивать другие серверы в поисках ответа. В этом случае сервер может делать либо рекурсивные запросы, возлагая таким образом ответственность за нахождение ответа на опрашиваемые DNS-серверы (и снимая с себя ответственность), либо итеративные запросы, возможно получая ссылки на DNS-серверы, расположенные «ближе» к искомому доменному имени. Существующие реализации очень воспитаны и по умолчанию действуют по второму варианту, следуя по ссылкам, пока не будет найден ответ.[2]

DNS-сервер, получивший рекурсивный запрос, на который он сам не в состоянии ответить, отправляет запрос «ближайшим известным» серверам. Ближайшие известные серверы – это те, которые являются авторитетными для зоны, ближе всего расположенной к доменному имени, о котором идет речь. К примеру, если сервер получает рекурсивный запрос для адреса доменного имени *girigiri.gbrmpa.gov.au*, он, прежде всего, выяснит, известно ли, какие серверы являются авторитетными для *girigiri.gbrmpa.gov.au*, и, если это известно, отправит запрос одному из них. В противном случае будет произведен аналогичный поиск DNS-серверов для *gbrmpa.gov.au*, а затем для *gov.au* и *au*. По умолчанию поиск не будет продолжаться дальше корневой зоны, поскольку каждому DNS-серверу известны доменные имена и адреса корневых серверов имен.

Использование ближайших известных DNS-серверов позволяет во всех случаях максимально сократить процесс разрешения. Если DNS-сервер *berkeley.edu* получает рекурсивный запрос адреса для *waxwing.ce.berkeley.edu*, он не будет опрашивать корневые серверы, а использует информацию о делегировании и отправится за данными прямо к DNS-серверам *ce.berkeley.edu*. Точно так же сервер, который толь-

[1] DNS-серверы большинства версий пакета BIND могут быть настроены таким образом, чтобы игнорировать или отказывать в выполнении рекурсивных запросов; причины и способы поступать именно так описаны в главе 11.

[2] Исключением является DNS-сервер, настроенный таким образом, чтобы передавать все запросы, на которые он не располагает ответом, определенному DNS-серверу. Такой сервер называется ретранслятором (forwarder). Подробно об использовании ретрансляторов рассказано в главе 10.

ко что производил поиск адреса для доменного имени в *ce.berkeley.edu*, не будет начинать поиск с корня, если необходимо повторить процедуру для *ce.berkeley.edu* (или для *berkeley.edu*); причины этого мы опишем чуть позже в разделе «Кэширование».

Сервер DNS, получающий рекурсивный запрос от DNS-клиента, при поиске повторяет этот запрос в точности так, как он получен от клиента. В случае рекурсивного запроса адреса для *waxwing.ce.berkeley.edu* сервером никогда не будет отправлен явный запрос на информацию о DNS-серверах *ce.berkeley.edu* или *berkeley.edu*, несмотря на то, что эта информация также хранится в пространстве имен. Прямые запросы могут приводить к осложнениям: DNS-серверы *ce.berkeley.edu* могут не существовать (то есть *ce.berkeley.edu* может являться частью зоны *berkeley.edu*). Помимо этого вполне возможно, что сервер имен *edu* или *berkeley.edu* уже знает адрес *waxwing.ce.berkeley.edu*. Прямой запрос о DNS-серверах *berkeley.edu* или *ce.berkeley.edu* не приведет к получению этой информации.

## Итеративное взаимодействие

Итеративное разрешение не требует от DNS-сервера такой серьезной работы. При итеративном разрешении сервер имен возвращает наилучший ответ *из уже известных ему*. Выполнение дополнительных запросов в таком случае не требуется. Сервер имен, получивший запрос, сверяется с локальными данными (в том числе и данными кэша, о котором мы скоро поговорим) в поисках запрошенной информации. Если конечный ответ в этих данных не содержится, сервер находит в локальных данных имена и адреса DNS-серверов, наиболее близко расположенных к доменному имени, для которого сделан запрос, и возвращает их в качестве указания для продолжения процесса разрешения. Заметим, что ответ включает *все* серверы имен, содержащиеся в локальных данных, и выбор следующего DNS-сервера для опроса совершается отправителем итеративного запроса.

## Выбор авторитетного DNS-сервера

Некоторые из читателей (состоящие в обществе Менса[1]), возможно, задаются вопросом: каким образом сервер, получивший рекурсивный запрос, выбирает DNS-сервер из списка авторитетных? Мы говорили о том, что существует 13 корневых DNS-серверов в сети Интернет. Посылает ли наш DNS-сервер запрос первому из серверов в списке? Или выбирает позицию в списке случайно?

---

[1] Международное сообщество, объединяющее людей, которые по своему IQ попадают в первые 2% населения. Основали его в 1946 г. в Англии адвокат Роланд Беррилл (Roland Berrill) и доктор Ланс Вэр (Lance Ware), ученый и юрист. – *Примеч. ред.*

DNS-серверы BIND используют метрику, называемую *временем отклика* (*roundtrip time*, или RTT), для выбора среди авторитетных DNS-серверов одной зоны. Время отклика определяет задержку, с которой приходит ответ на запросы от удаленного сервера. Каждый раз при передаче запроса удаленному серверу DNS-сервер BIND запускает внутренний таймер. Таймер останавливается при получении ответа, и метрика фиксируется локальным сервером. Если серверу приходится выбирать один из нескольких авторитетных серверов, выбор падает на сервер с наименьшим временем отклика.

До того как сервер BIND впервые послал запрос некоему серверу и получил от него ответ, удаленному серверу присваивается случайное значение времени отклика, которое меньше, чем все прочие, полученные на основании замеров. Таким образом, DNS-сервер BIND гарантированно опросит все авторитетные серверы для определенной зоны случайным образом, прежде чем начнет выбирать предпочтительный на основании метрики.

В общем и целом, это простой, но элегантный алгоритм, позволяющий DNS-серверам BIND достаточно быстро «зацикливаться» на ближайших DNS-серверах, не прибегая к нестандартным, требовательным к ресурсам механизмам измерения производительности.

## Картина в целом

В итоге мы можем наблюдать процесс разрешения, который в целом выглядит примерно так, как показано на рис. 2.13.

DNS-клиент отправляет запрос локальному DNS-серверу, который посылает итеративные запросы ряду других серверов в поисках ответа. Каждый из опрашиваемых серверов возвращает ссылку на другой сервер, который является авторитетным для зоны, расположенной ближе к искомому доменному имени и глубже в дереве пространства имен. В итоге локальный сервер посылает запрос авторитетному серверу, который и возвращает ответ. На протяжении всего процесса поиска локальный DNS-сервер использует каждый из получаемых ответов, будь то ссылка или искомая информация, для обновления метрики RTT для реагирующих на запросы DNS-серверов, что впоследствии позволяет принимать осмысленные решения при выборе серверов, используемых в процессе разрешения доменных имен.

## Отображение адресов в имена

До сих пор в разговоре о процессе разрешения мы не затрагивали один важный элемент функциональности, а именно – отображение адресов в имена доменов. Отображение адрес-имя необходимо для получения вывода, который легко воспринимается людьми (скажем, при чтении log-файлов). Это отображение также применяется в авторизации. Например, UNIX-узлы преобразуют адреса в доменные имена с целью

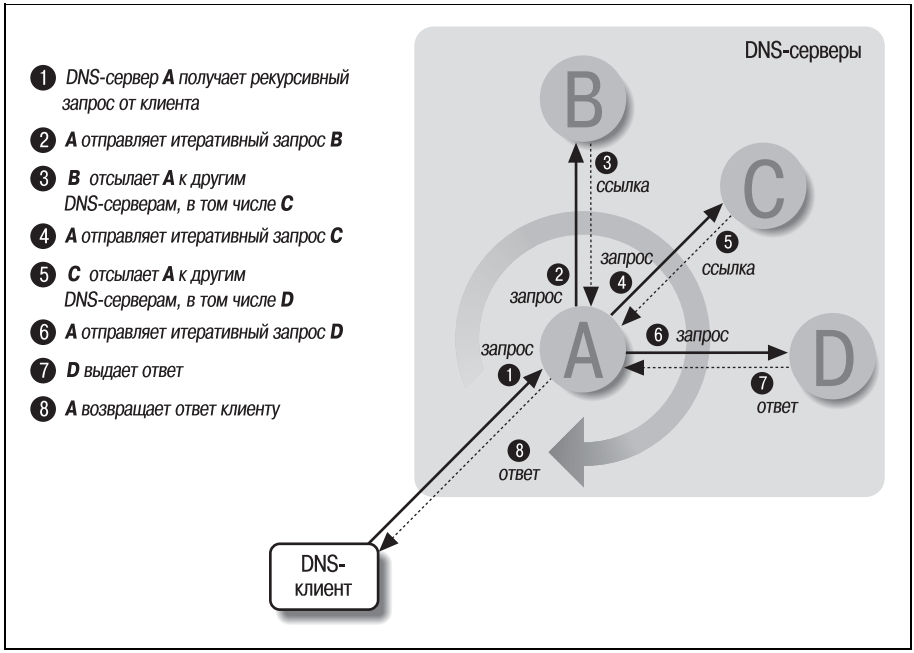



***Рис. 2.13.*** *Процесс разрешения*

сравнения их с записями в файлах *.rhosts* и *hosts.equiv*. При использовании таблиц узлов отображение адресов в доменные имена довольно тривиально. Требуется обычный последовательный поиск адреса в таблице узлов. Поиск возвращает указанное в таблице официальное имя узла. Но в DNS преобразование адреса в имя происходит несколько сложнее. Данные, в том числе и адреса, которые хранятся в пространстве доменных имен, индексируются по именам. Если есть доменное имя, поиск адреса – процедура относительно простая. Но поиск доменного имени, которому соответствует заданный адрес, казалось бы, потребует полного перебора данных для всех доменных имен дерева.

На деле же существует другое решение, более разумное и эффективное. Несложно производить поиск по доменным именам, поскольку они являются своеобразными индексами базы данных, и точно таким же образом можно создать сектор пространства доменных имен, в котором в качестве меток будут использоваться адреса. В пространстве доменных имен сети Интернет таким свойством обладает домен *in-addr.arpa*.

Узлам домена *in-addr.arpa* в качестве меток присваиваются числа в нотации IP-адреса (dotted octet representation – октеты, разделенные точками, – распространенный метод записи 32-битных IP-адресов в виде четырех чисел, принадлежащих интервалу от 0 до 255 и разделяемых точками). Так, домен *in-addr.arpa* может содержать до 256 поддоменов, каждый из которых будет соответствовать одному из возможных

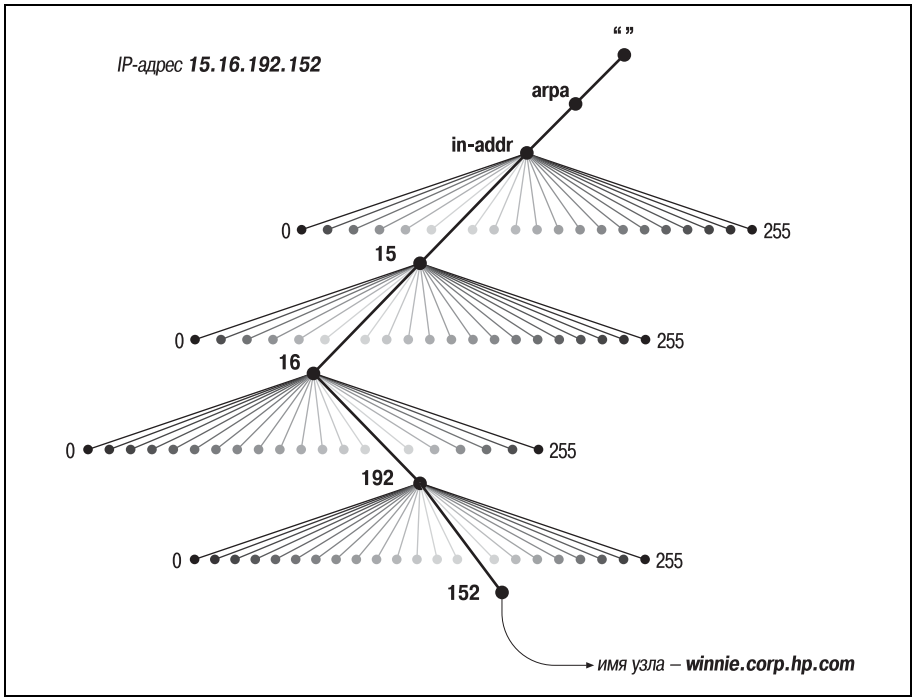



***Рис. 2.14.*** *Домен in-addr.arpa*

значений первого октета IP-адреса. Каждый из этих поддоменов может содержать до 256 собственных поддоменов, каждый из которых будет соответствовать одному из возможных значений второго октета. Наконец, на четвертом уровне существуют RR-записи, ассоциированные с последним октетом, которые содержат полное доменное имя узла по данному IP-адресу. Результатом подобных построений является невероятно объемный домен: *in-addr.arpa*, отображенный на рис. 2.14, достаточно вместительный, чтобы охватить все IP-адреса сети Интернет.

Заметим, что при чтении в доменном имени IP-адрес оказывается записанным наоборот, поскольку имя читается от листа дерева к корню. К примеру, если IP-адрес узла *winnie.corp.hp.com* – 15.16.192.152, соответствующий узел в домене *in-addr.arpa* – *152.192.16.15.in-addr.arpa*, который отображается в доменное имя *winnie.corp.hp.com*.

IP-адреса могли бы быть представлены в пространстве имен другим способом так, чтобы первый октет IP-адреса был дальше от корня домена *in-addr.arpa*. Тогда в доменном имени IP-адрес читался бы в «правильном» направлении. Однако IP-адреса, как и доменные имена, образуют иерархию. Сетевые номера выделяются почти так же, как и доменные имена, и администраторы вольны разделять «свое» адресное пространство на подсети и делегировать нумерацию в сети другим администраторам. Разница заключается в том, что конкретизация узла для IP-адреса

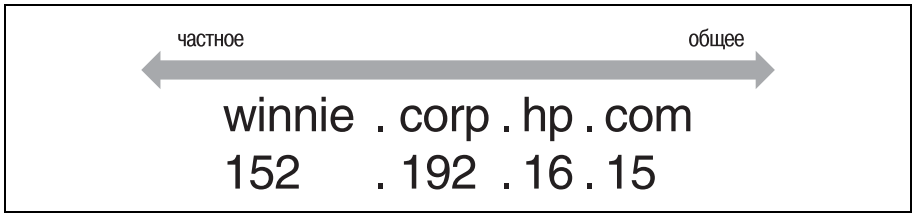



***Рис. 2.15.*** *Иерархия имен и адресов*

возрастает при чтении слева направо, тогда как для доменных имен – справа налево. Суть этого явления отражена на рис. 2.15.

То, что первые октеты IP-адреса расположены выше в дереве, позволяет администраторам делегировать ответственность за зоны *in-addr.arpa* в соответствии с топологией сети. Возьмем для примера зону *15.in-addr.arpa*, которая содержит данные обратного преобразования для всех узлов, адреса которых начинаются с цифры 15: эта зона может быть делегирована администраторам сети 15/8. Это стало бы невозможным, если бы октеты были расположены в обратном порядке. Если бы IP-адреса записывались наоборот (в другом порядке), зона *15.in-addr.arpa* включала бы каждый узел, IP-адрес которого *заканчивается* цифрой 15, и делегировать эту зону было бы немыслимо.

## Кэширование

По сравнению с простым поиском в таблице узлов процесс разрешения может показаться ужасно запутанным и нескладным. В действительности же этот процесс довольно быстр. Одна из возможностей, существенно его ускоряющих, – *кэширование*.

Чтобы обработать рекурсивный запрос, DNS-серверу приходится самостоятельно сделать довольно много запросов. Однако в процессе сервер получает большое количество информации о пространстве доменных имен. Каждый раз, получая список DNS-серверов в ссылке, он знакомится с серверами, авторитетными для какой-то зоны, и узнает адреса этих серверов. Когда завершается процесс разрешения и необходимые данные возвращаются клиенту, сделавшему исходный запрос, новые знания могут быть сохранены для последующего использования. В сервере BIND даже реализовано *отрицательное кэширование*: если авторитетный сервер возвращает информацию о том, что запрошенное имя домена или указанный тип данных не существует, локальный сервер временно кэширует и эту информацию.

DNS-серверы кэшируют получаемые данные, чтобы ускорить обработку последующих запросов. Когда в следующий раз DNS-клиент сделает запрос по доменному имени, о котором серверу уже что-то известно, процесс разрешения пройдет быстрее. Если сервер кэшировал ответ, положительный или отрицательный, ответ просто возвращается кли-

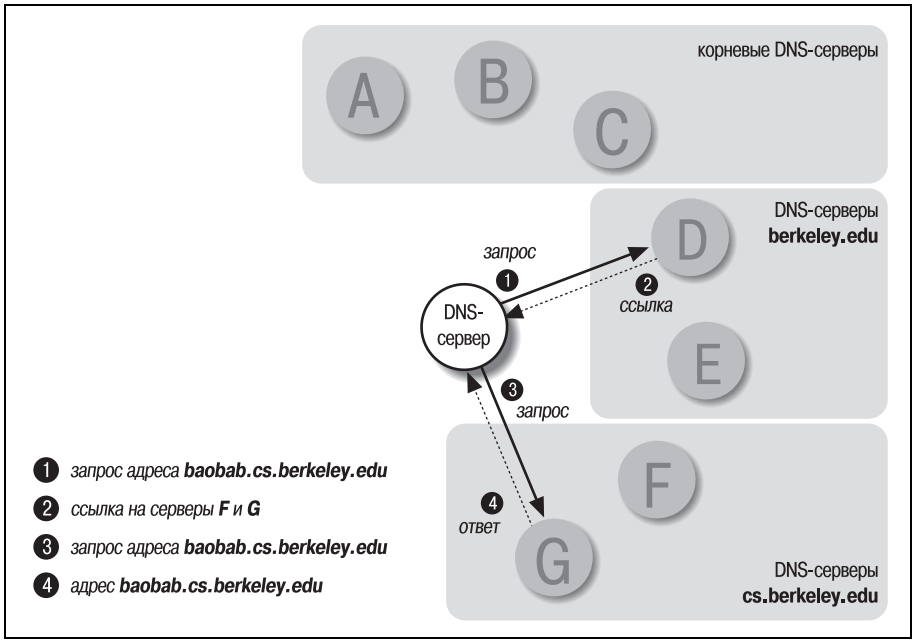



***Рис. 2.16.*** *Процесс разрешения для baobab.cs.berkeley.edu*

енту. Но даже если готового ответа у DNS-сервера нет, он, возможно, «помнит в лицо» те DNS-серверы, которые являются авторитетными для зоны этого конкретного доменного имени, и будет напрямую работать с ними.

Предположим, наш DNS-сервер уже выяснял адрес для *eecs.berkeley.edu*. В процессе были кэшированы имена и адреса DNS-серверов *eecs.berkeley.edu* и *berkeley.edu* (а также IP-адрес *eecs.berkeley.edu*). Если теперь клиент пошлет DNS-серверу запрос, касающийся адреса для имени *baobab.cs.berkeley.edu*, при обработке этого запроса наш сервер сможет пропустить отправку запросов корневым DNS-серверам. Опознав в *berkeley.edu* ближайшего предка *baobab.cs.berkeley.edu,* о котором уже что-то известно, наш сервер начнет с запроса к DNS-серверу *berkeley.edu* (рис. 2.16). С другой стороны, если бы наш DNS-сервер выяснил, что адреса для *eecs.berkeley.edu* не существует, в следующий раз на подобный запрос он вернул бы соответствующий ответ из кэша.

Помимо ускорения разрешения, кэширование устраняет необходимость вовлекать в процесс разрешения корневые DNS-серверы для случаев, когда ответ может быть получен локально. Это означает уменьшение зависимости от корневых серверов и снижение нагрузки на них.

## Время жизни

Разумеется, DNS-серверы не могут кэшировать данные навсегда. Иначе изменения на авторитетных серверах никогда не распространялись бы по сети. Удаленные серверы просто продолжали бы использовать кэшированную информацию. Поэтому администратор зоны, которая содержит данные, обычно определяет для этих данных *время жизни* (*time to live*, или TTL). Время жизни – это интервал времени, в течение которого произвольному DNS-серверу разрешается пользоваться кэшированными данными. По окончании этого временного интервала сервер обязан удалить кэшированную информацию и получить новую от авторитетных DNS-серверов. Это касается также кэшируемых отрицательных ответов, сервер имен обязан удалять их на случай, если на авторитетных DNS-серверах появилась новая информация.

Когда администратор выбирает время жизни для своих данных, то фактически пытается найти золотую середину между производительностью и согласованностью данных. Небольшой показатель TTL гарантирует, что данные о зоне будут согласованы по всей сети, поскольку удаленные серверы будут обязаны быстрее выбросить данные из кэша и обратиться на авторитетные серверы за новыми данными. С другой стороны, увеличивается нагрузка на DNS-серверы зоны и увеличивается время разрешения, когда речь идет об информации из этой зоны.

Напротив, большой показатель TTL сокращает среднее время, затрачиваемое на получение информации из зоны, поскольку данные о зоне кэшируются в течение длительного времени. Минус же в том, что информация на удаленных DNS-серверах в течение более длительного времени может не соответствовать действительности в случае изменения данных локальных DNS-серверов.

Но хватит уже теории – читателям, вероятно, не терпится с ней закончить. Однако, прежде чем можно будет перейти к реальным зонам и серверам имен, предстоит сделать домашнее задание, которое мы дадим в следующей главе.