

The Essential Resource for Linux Administrators

Linux DNS Server Administration

Craig Hunt

Craig Hunt

Craig Hunt
Linux Library



Associate Publisher: Neil Edde
Contracts and Licensing Manager: Kristine O'Callaghan
Acquisitions & Developmental Editor: Maureen Adams
Editor: Sarah Lemaire
Production Editor: Molly Glover
Technical Editor: Will Deutsch
Book Designer: Bill Gibson
Graphic Illustrator: Tony Jonick
Electronic Publishing Specialist: Adrian Woolhouse
Proofreaders: Dave Nash, Laurie O'Connell, Nancy Riddiough, Suzanne Stein, Nathan Whiteside
Indexer: Matthew Spence
Cover Designer: Ingalls & Associates
Cover Illustrator: Ingalls & Associates

Copyright © 2000 SYBEX Inc., 1151 Marina Village Parkway, Alameda, CA 94501. World rights reserved. No part of this publication may be stored in a retrieval system, transmitted, or reproduced in any way, including but not limited to photocopy, photograph, magnetic or other record, without the prior agreement and written permission of the publisher.

Library of Congress Card Number: 00-105386

ISBN: 0-7821-2736-3

SYBEX and the SYBEX logo are trademarks of SYBEX Inc. in the USA and other countries.

Screen reproductions produced with Gnu Image Manipulation Program. GIMP is a freely available public domain package included as part of Linux.

Netscape Communications, the Netscape Communications logo, Netscape, and Netscape Navigator are trademarks of Netscape Communications Corporation. Netscape Communications Corporation has not authorized, sponsored, endorsed, or approved this publication and is not responsible for its content. Netscape and the Netscape Communications Corporate Logos are trademarks and trade names of Netscape Communications Corporation. All other product names and/or logos are trademarks of their respective owners.

Internet screen shot(s) using Microsoft Internet Explorer version 5 reprinted by permission from Microsoft Corporation.

TRADEMARKS: SYBEX has attempted throughout this book to distinguish proprietary trademarks from descriptive terms by following the capitalization style used by the manufacturer.

The author and publisher have made their best efforts to prepare this book, and the content is based upon final release software whenever possible. Portions of the manuscript may be based upon pre-release versions supplied by software manufacturer(s). The author and the publisher make no representation or warranties of any kind with regard to the completeness or accuracy of the contents herein and accept no liability of any kind including but not limited to performance, merchantability, fitness for any particular purpose, or any losses or damages of any kind caused or alleged to be caused directly or indirectly from this book.

Manufactured in the United States of America

10 9 8 7 6 5 4 3 2 1

1

The DNS Architecture

A great deal of ink is wasted explaining why you need to translate computer names to numeric addresses. System administrators already know the reason: Users like names and networks like numbers. And as a system administrator, it is your job to keep both the users and the network happy and productive.

In this chapter, you'll learn how names are translated into numbers on networks that use the Internet Protocol (IP). This chapter explains terms used later in this book and lays the groundwork for understanding why certain tasks are required to create a running domain name server.

You will begin by looking at the file that was originally used for this job. You will see how the limitations of that file lead to the distributed hierarchical database system structure of today's Domain Name System (DNS). The structure of domain names and what they show about the domain hierarchy are discussed along with your server's place in the hierarchy and the role of the root servers. For those readers who are completely new to DNS, how you obtain your own domain name is explained. All of this begins with a look into the past.

The */etc/hosts* File

In the beginning, the host table was the only tool used to map host names to Internet addresses. All Linux systems still have a host table, which is stored in the */etc/hosts* file.

In fact, all IP network servers have a host table. If you have experience with Unix, you recognize the name `/etc/hosts` because that is where Unix systems store the host table. If your background is Windows NT, you know the host table as the `%SystemRoot%\System32\Drivers\etc\hosts` file. The structure and content of the `hosts` file is the same on all of these systems. Listing 1.1 contains an example from a Linux system that illustrates the structure of the `hosts` file:

Listing 1.1 Structure of the `/etc/hosts` File

```
$ cat /etc/hosts
#
# Table of IP addresses and host names
#
127.0.0.1          localhost
172.16.5.5        crow
172.16.5.1        wren ns1
172.16.5.4        hawk gw5
172.16.5.20       kestrel kestral
172.16.5.2        robin redbreast bob
172.16.5.6        eagle www
172.16.5.7        bluebird blue news
```

Lines that begin with a sharp sign (#) are comments. All other lines define data for the host table and all of these lines have the same format: A data line begins with an IP address that is followed by a list of names that map to that address. The first name in the list is the primary name assigned to the address. All of the other names are aliases or nicknames.

Host name aliases are used to provide shorter names, historic names, generic names, and alternate spellings for a given host name. All of these things are done as a convenience for users. The convenience of a shorter name is obvious and needs no explanation. *Historic names* are used to ease a name transition. When the name of a host is changed, the old historic name may still be used by some users or may still be embedded in some old scripts. *Generic names* are the names that users expect to find for given services. For example, `www` for the Web server, `mail` for the mail server, and `news` for the news server are names the user expects to find. Alternate spellings can help when host names are difficult to spell.

TIP If you must create alternate spelling aliases or you frequently change host names, then you probably chose the wrong host name in the first place. Host names should be easy to spell and should be independent of things that can easily change such as location, user, or task. See RFC 1178, “Naming Your Computer,” for advice on choosing a host name.

Using the sample `hosts` file in Listing 1.1, a user could specify `bluebird`, `blue`, or `news`, and the system would return the IP address 172.16.5.7. From the system’s point of view, all of these names are the same because they all point to the same IP address. Given this, you might wonder why the first name in the list is called the *primary name*. The primary name is the name used when the system does a *reverse lookup* to convert a numeric address back into a name. Numeric addresses are converted to names to create more readable displays. For example, the `netstat` command obtains IP addresses when it determines the network status but, by default, it displays host names for those addresses, as follows:

```
$ netstat --inet
Active Internet connections (w/o servers)
Proto R-Q S-Q Local Address Foreign Address State
tcp    1  0 robin:1967 eagle:80 CLOSE_WAIT
tcp    1  0 robin:1966 eagle:80 CLOSE_WAIT
tcp    1  0 robin:1964 eagle:80 CLOSE_WAIT
tcp    1  0 robin:1963 eagle:80 CLOSE_WAIT
tcp    0 126 robin:23 hawk:1449 ESTABLISHED
```

As this example shows, the `netstat` command uses the primary name for each host in its display. The `netstat --inet` command displays the TCP/IP network connections. Each line lists the hosts and the ports involved in a connection. If the local computer only has access to a host table for address-to-name resolution, the name that shows up for each host is the primary name associated with that host’s address in the host table.

Analyzing the Host Table

A line-by-line analysis of the host table in Listing 1.1 explains the various types of host table entries, including those entries you probably have in your own host table. These first two lines represent two lines found in every host table:

```
127.0.0.1 localhost
172.16.5.5 crow
```

The first line defines the loopback address 127.0.0.1 and assigns it to the host name `localhost`. The *loopback address* is a software construct that allows the systems to send data through the TCP/IP stack to itself without actually sending the data out on the network. The loopback facility simplifies testing, permits the system to use the same code used for network communications when sending data between two local processes, and reduces the amount of traffic on the network.

The second line defines the name and address of the local computer. In this example, the local computer is named `crow` and is assigned the address 172.16.5.5. Every computer has its own address and name in its host table.

The next two lines in the sample host table represent entries for the local DNS server and the local default router:

```
172.16.5.1    wren ns1
172.16.5.4    hawk gw5
```

These entries are useful when the system is booting. `wren`, which is the name server for this sample network, has the nickname `ns1`, which stands for “name server 1.” `hawk`, which is the default gateway for this network, has the nickname `gw5`, which stands for “gateway 5.” These entries illustrate the use of nicknames and represent the types of entries that you might actually have in your own host table.

The next two lines represent the type of entries that might be used to catch spelling errors and to handle historic names:

```
172.16.5.20  kestrel kestral
172.16.5.2   robin redbreast bob
```

In the first line, the system administrator has added a nickname to accept the typo `kestral` as a valid name because a user has trouble typing `kestrel` correctly. In the second line, a nickname is included for the historic name `bob` that was assigned to address 172.16.5.2 before the local network standardized on bird names. These lines were added by the system administrator to handle specific problems experienced by the system’s users. Rarely would anything like this be required in your host table.

The last two lines illustrate the use of generic names, such as `www` and `news`:

```
172.16.5.6   eagle www
172.16.5.7   bluebird blue news
```

It is unlikely that you will need to put values like this in the local host table because, if you actually have a Web server or a network news server, you almost certainly have DNS running. When DNS is running, all computers can get these values from the DNS server and they don’t need to get them from the local host table.

The sample `/etc/hosts` file contains eight lines and yet it is probably twice as long as the `hosts` file you will actually have on your computer. On most computers, the host table has only limited utility and the system relies on a name server for most name-to-address translation. Even small, isolated networks generally rely on a service such as Network Information Service (NIS) to create a centralized host table because it is easier to maintain a single server than it is to maintain a separate copy of the host table on every system. Therefore, even when the host table is the primary means of resolving names, only one server has a large host table and most other computers have very small host tables. (See Appendix D, “Configuring Network Information Service,” for a description of deploying an NIS server under Linux.)

Host Table-to-DNS Scripts

There are scripts that convert a host table to a DNS database. The problem with these scripts is that they assume your system has a large host table worth converting. Personally, I haven't had a system with a large host table since 1987! These scripts are easy to use, but creating a large host table on a Linux system that runs DNS just doesn't make sense. Chapter 6, “Creating a Master Server,” explains how easy it is to put host information directly into the DNS database. You can safely ignore anything you read about host table-to-DNS scripts.

Uses for the Host Table

Though limited, the role of the host table is important. The host table is used to resolve critical addresses, like the address of the default gateway, at times when DNS is not available, such as during the initial boot. The default gateway is a particularly good example of this. It is very possible that the name server is on the far side of your default gateway. Until your system adds the default gateway to its configuration, it cannot communicate with the remote name server. In this case, placing the address of the default gateway in your local host table allows you to configure the gateway without querying a name server that you might not be able to reach.

Beyond these limited roles, the host table has no real use on most systems. All systems that have access to the Internet rely on DNS for name-to-address resolution. Of course, it is possible that your system is on a small isolated network that does not connect to the Internet and that never needs to communicate with remote systems. In that case, the host table might be adequate for all of your needs, particularly if combined with the central maintenance features provided by NIS. However, that is an extremely rare case. For

In Listing 6.4, every host name has a unique address and every address has a unique host name. This does not have to be the case, as illustrated in Listing 6.5.

Listing 6.5 Address Records for Multi-Homed Hosts

```

ow1           A       172.16.5.15
ow1           A       172.16.7.32
ow15          A       172.16.5.15
ow17          A       172.16.7.32

```

In Listing 6.5, `ow1` is a multi-homed host. The name `ow1` is assigned both of the host's addresses, so a query for `ow1` returns both `172.16.5.15` and `172.16.7.32`. The remote system's resolver can sort out which address it wants to use. (As you learned in Chapter 4, "Configuring the Resolver," the resolver `sortlist` option can be used to control which address is used.)

The address of each interface of the multi-homed host is also assigned a unique name—`ow15` and `ow17` in Listing 6.5. These unique names permit those interfaces to be addressed directly for testing or other purposes. Even simple name-to-address mapping is more flexible in DNS than it is in the host table.

Defining Nicknames

Host name aliases, or nicknames, provide shorter names, "historic" names, generic names, and alternate spellings. (All of these nickname functions are described in Chapter 1, "The DNS Architecture.") The CNAME record is used to define a host alias. The name field of the CNAME record contains the host alias and the data field contains the official (canonical) name of the host. That's why CNAME is short for canonical name record. The last five records in the sample zone file are CNAME records, as shown in Listing 6.6.

Listing 6.6 Sample CNAME Records

```

redbreast    CNAME  robin.foobirds.org.
bob          CNAME  robin.foobirds.org.
kestr1       CNAME  kestrel,foobirds.org.
www          CNAME  wren.foobirds.org.
news         CNAME  parrot.foobirds.org.

```

Moving down the list of CNAME records provides the following information: `redbreast` is a host name alias for `robin`. `bob` is also a nickname for `robin`. In this case, `bob` is a historic name from a time before we standardized on bird names. It still occasionally pops up out of an old mailing list or newsgroup so we keep the alias around. The `kestr1` alias is used as an alternate spelling of `kestrel` because we have a user who just

The *name* field contains the name of the domain or host to which the mail is addressed. The *server* field contains the name of the server to which the mail is delivered. *preference* is a number used to select the most preferred server when a domain or host has multiple MX records, with low numbers preferred over high numbers. See Chapter 6 for more information about using the MX record, which is defined in RFC 1035.

Text Record (TXT) The Text record is used to define free-form information about the named object. Its format is simple

```
name ttl class TXT string
```

On Linux systems, the TXT record is generally used to provide information about a host to the technical support group. The following example illustrates this use:

```
buzzard IN TXT "Accounting Department server in room B152"
```

Because of its free-form nature, the TXT record has been used over time for special purposes, such as providing input to locally developed scripts that collect domain information. An earlier version of BIND that runs on some Unix systems even uses these records to define security information. Linux, of course, uses the latest version of BIND so it does not need to use TXT records for security. RFC 1035 defines the TXT record.

Responsible Person Record (RP) The Responsible Person record identifies the point of contact for a host or domain. The format of the RP record is

```
name ttl class RP mail_address text_pointer
```

The *mail_address* is the e-mail address of the responsible person. The @ usually included in an e-mail address is replaced with a dot. Thus, `craig@foobirds.org` becomes `craig.foobirds.org`. The *text_pointer* is the domain name of a TXT record that contains additional information about the responsible person. Here's an example of how an RP record is used with a TXT record:

```
ibis.foobirds.org. IN RP craig.foobirds.org. ibisRP
ibisRP.foobirds.org. IN TXT "Craig Hunt (301)555-1234 X237"
```

The RP record states that the person responsible for `ibis.foobirds.org` can be reached via e-mail at `craig@foobirds.org` and that additional information about the person can be obtained in the TXT records for `ibisRP.foobirds.org`. The TXT record provides the contact person's name and phone number.

Use RP records to make it easier for system administrators to contact each other when things go wrong. Unfortunately, most domains don't use RP records. Maybe the system administrators don't want people to know how to get in touch with them. But if you don't use RP records, remote administrators will contact the domain administrator when the

Linux

DNS Server Administration

Answers to all your DNS questions—written specifically for Linux administrators *Linux DNS Server Administration* is the most complete, most advanced guide to DNS for Linux you'll find anywhere. Written by a leading Linux expert, this book teaches you, step-by-step, all the standard and advanced techniques you need to know to configure and maintain a DNS server on a Linux box. Hundreds of clear, consistent examples illustrate these techniques in detail—so you stay on track and accomplish all your goals. Coverage includes:

- Understanding DNS architecture
- Understanding the protocols and messages used by DNS clients and servers
- Downloading, compiling, and installing BIND
- Configuring the resolver using the resolv.conf file and environment variables
- Setting the order of resolver queries
- Configuring the named.conf file for caching, slave, and master name servers
- Creating a domain database file and a reverse domain database file
- Creating delegated and non-delegated subdomains for your domain and reverse domain
- Securing a Linux DNS server with wrapper and ipchains
- Understanding and deploying DNSSEC protocols
- Testing DNS with host, dig, and nslookup
- Using the named.run, named_dump.db, and named.stats files to monitor your server
- Customizing BIND logging

About the Library

The **Craig Hunt Linux Library** is an eight-book set that provides in-depth, advanced coverage of the key topics for Linux administrators. Topics include Samba, System Administration, Sendmail, Apache, NFS and Automounter, and Linux Security. Each book in the library is either written by or meticulously reviewed by Craig Hunt to ensure the highest quality and most complete coverage for networking professionals working specifically in Linux environments.



Visit Sybex's Web site
(www.sybex.com)
for more information.



SYBEX®

COMPUTER BOOK SHELF CATEGORY

Networking

US \$39.99 CAN \$59.95 UK £29.99

ISBN 0-7821-2736-3

- Perform Advanced Configuration Tasks
- Clear, In-Depth Coverage of Every Aspect of the Domain Name System
- Master the Features of BIND 8 —and Look Ahead to the Forthcoming BIND 9
- Look for Other Titles in the *Craig Hunt Linux Library* from Sybex

About the Author

Craig Hunt is a noted TCP/IP and Linux expert who lectures regularly on the topics at the NetWorld+Interop, ComNet, and other networking trade shows. His other books include the best-selling *Linux Network Servers 24seven* from Sybex and the classic *TCP/IP Network Administration* from O'Reilly & Associates.

