

KAPITEL 8

Die Konfiguration des DNS

Herzlichen Glückwunsch! Sie haben TCP/IP im Kernel installiert, die Netzwerkschnittstelle konfiguriert und Routing eingerichtet. An dieser Stelle haben Sie alle Konfigurationsarbeiten erledigt, die zum Betrieb von TCP/IP auf einem Unix-System erforderlich sind. Die nun noch verbleibenden Arbeiten sind zwar nicht *notwendig*, um TCP/IP zu betreiben, machen das Netz aber freundlicher und nützlicher. In den beiden nächsten Kapiteln sehen wir uns an, wie man die grundlegenden TCP/IP-Netzwerkdienste konfiguriert. Der wohl wichtigste dieser Dienste ist der Namensdienst.

Es handelt sich, wie der Name es andeutet, um einen Dienst – und zwar um einen Dienst, der das Netzwerk benutzerfreundlicher machen soll. Computer sind mit IP-Adressen zufrieden, während wir Menschen Namen vorziehen. Wie wichtig der Namensdienst ist, können Sie schon allein daran sehen, welchen Umfang wir ihm in diesem Buch einräumen. Kapitel 3 erläutert, *warum* der Namensdienst notwendig ist. Dieses Kapitel beschreibt, *wie* er konfiguriert wird, und Anhang C behandelt die *Details* der Name-server-Konfigurationsbefehle. In diesem Kapitel versorgen wir Sie mit ausreichend Informationen, um Ihnen zu zeigen, wie Sie die BIND-Software auf Ihrem System zum Laufen bringen.¹ Sollten Sie jedoch genauer wissen wollen, warum oder wie bestimmte Dinge gemacht werden, können Sie in Kapitel 3 und Anhang C nachsehen.

BIND: Unix-Namensdienst

Unter Unix ist das DNS im Software-Paket *Berkeley Internet Name Domain* (BIND) implementiert. BIND ist ein Client/Server-System. Die Client-Seite von BIND wird *Resolver* genannt. Der Resolver generiert die Abfragen (Queries) nach den Domain-Namen und sendet diese an den Server. Die Software des DNS-Servers beantwortet die Abfragen des Resolvers. Die Server-Seite von BIND ist ein Dämon mit der Bezeichnung *named*.

¹ BIND 8 ist die Version der Domain Name Software, die mit den meisten Linux-Versionen und Solaris 8 geliefert wird. Eine neuere Version der DNS-Software – BIND 9 – steht ebenfalls zur Verfügung. BIND 8 und BIND 9 verwenden im Prinzip die gleiche Syntax in der Konfigurationsdatei. Die hier gezeigten Beispiele sollten sowohl mit BIND 8 als auch mit BIND 9 funktionieren.

Dieses Kapitel behandelt drei grundlegende BIND-Konfigurationsaufgaben:

- Die Konfiguration des BIND-Resolvers.
- Die Konfiguration des BIND-Nameservers (*named*).
- Den Aufbau der Datenbankdateien des Nameservers, der sogenannten *Zonendateien* (*zone files*).

Eine *Zone* ist ein Teil des Namensraums einer Domain, über den ein Nameserver die Autorität besitzt. Eine Zone kann keine Domain enthalten, die an einen anderen Server delegiert wurde. Wir verwenden den Begriff »Zone«, wenn wir von einer DNS-Datenbankdatei sprechen, während der Begriff »Domain« in allgemeinerem Kontext eingesetzt wird. In diesem Buch ist eine Domain ein Teil der Domain-Hierarchie, die durch den Domain-Namen gekennzeichnet ist. Eine Zone ist eine Sammlung von Domain-Informationen, die in einer Domain-Datenbankdatei enthalten sind. Die Datei, die diese Domain-Informationen enthält, wird als Zonendatei bezeichnet.

RFC 1033, der *Domain Administrators Operations Guide*, definiert den grundlegenden Satz von Standarddatensätzen (Records), die zum Aufbau der Zonendatei verwendet werden. Viele RFCs schlagen neue DNS-Records vor, die allerdings weitgehend nicht implementiert sind. In diesem Kapitel und in Anhang C konzentrieren wir uns auf die grundlegenden Resource Records, die Sie am häufigsten verwenden werden. Wir werden diese Records benutzen, um die in diesem Kapitel verwendeten Zonendateien aufzubauen. Aber wie oder ob Sie überhaupt Zonendateien für Ihr System aufbauen müssen, wird durch die Art der BIND-Konfiguration vorgegeben, für die Sie sich entscheiden.

BIND-Konfigurationen

BIND-Konfigurationen werden durch die Art des Dienstes beschrieben, den die Software anbieten soll. Die vier Stufen von Diensten, die in einer BIND-Konfiguration definiert werden können, sind *reine Resolver-Systeme*, *Caching-Only-Server* (reine Cache-Server), *Master-Server* (primäre Server) und *Slave-Server* (sekundäre Server).

Der Resolver ist der Code, der Nameserver nach Domain-Informationen abfragt. Auf Unix-Systemen ist der Resolver in Form einer Programmbibliothek implementiert, d. h., es handelt sich nicht um ein separates Client-Programm. Manche Systeme, die sogenannten reinen Resolver (*resolver-only*), arbeiten nur mit dem Resolver. Sie führen also keinen Nameserver aus. Solche reinen Resolver-Systeme sind sehr einfach zu konfigurieren: Sie müssen nur die Datei */etc/resolv.conf* einrichten.

Für die drei anderen BIND-Konfigurationen ist es notwendig, daß die Server-Software *named* auf dem lokalen System läuft. Diese Konfigurationen sind:

Master

Der Master-Nameserver ist die verbindliche Quelle (*authoritative source*) für alle Informationen über eine bestimmte Zone. Er lädt die Domain-Informationen aus einer lokalen Datei von der Festplatte, die vom Domain-Administrator aufgebaut

und gepflegt wird. Diese Datei (die Zonendatei) enthält die genauesten Informationen über den Teil der Domain-Hierarchie, über den dieser Nameserver die Autorität besitzt. Der Master-Server ist ein autoritativer Server, weil er jede Anfrage über seine Zone mit voller Autorität beantworten kann.

Die Konfiguration eines Master-Servers erfordert einen vollständigen Satz an Konfigurationsdateien: Zonendateien für die Forward-Mapping-Zone und die Reverse-Mapping-Zone, die conf-Datei, die root-hints-Datei und die Loopback-Datei. Keine andere Konfiguration erfordert den vollständigen Einsatz all dieser Dateien.

Slave

Ein Slave-Server überträgt den kompletten Satz an Zoneninformationen vom Master-Server. Die Zonendaten werden vom Master-Server übertragen und auf der lokalen Festplatte des Slave-Servers in einer Datei abgelegt. Diese Übertragung wird treffend als *Zonentransfer* bezeichnet. Ein Slave-Server hält eine vollständige Kopie aller Zoneninformationen vor und kann Anfragen zu dieser Zone mit der entsprechenden Autorität beantworten. Ein Slave-Server wird deshalb ebenfalls als autoritativer Server betrachtet.

Bei der Konfiguration eines Slave-Servers ist der Aufbau lokaler Zonendateien nicht notwendig, da diese vom Master-Server heruntergeladen werden. Allerdings werden andere Dateien (Boot-, Cache- und Loopback-Datei) verlangt.

Caching-Only

Ein Caching-Only-Server führt die Nameserver-Software aus, hält aber keine Zonendateien vor. Er lernt die Antworten auf alle Nameserver-Abfragen von irgendeinem entfernten Server. Sobald er eine Antwort gelernt hat, legt der Server diese in einem Cache ab und verwendet sie, um künftige Abfragen dieser Informationen zu beantworten. Alle Nameserver verwenden im Cache liegende Informationen auf diese Weise, aber nur Caching-Only-Server bauen die Beantwortung von Abfragen allein auf dieser Technik auf. Ein solcher Server wird nicht als autoritativ betrachtet, weil er alle Informationen aus zweiter Hand hat. Nur eine Boot- und eine Cache-Datei sind für die Konfiguration eines solchen Cache-Servers notwendig. Die gängigste Konfiguration schließt aber auch eine Loopback-Datei ein. Diese Konfiguration ist wahrscheinlich die häufigste und, abgesehen vom reinen Resolver-Betrieb, die am einfachsten zu konfigurierende.

Ein Nameserver kann in einer dieser Konfigurationen, oder, was noch häufiger der Fall ist, in einer Kombination dieser Konfigurationsarten betrieben werden. Alle Systeme müssen jedoch den Resolver ausführen, weshalb wir auch mit der Konfiguration der Client-Seite der DNS-Software beginnen wollen.

Die Konfiguration des Resolvers

Der Resolver wird mit Hilfe der Datei */etc/resolv.conf* konfiguriert. Der Resolver ist kein eigener Prozeß, sondern eine Bibliothek mit Routinen, die von Netzwerkprozessen aufgerufen werden. Die Datei *resolv.conf* wird eingelesen, wenn ein Prozeß startet, der den Resolver nutzt. Der Inhalt dieser Datei wird während der gesamten Lebensdauer dieses Prozesses zwischengespeichert. Wird die Konfigurationsdatei nicht gefunden, versucht der Resolver, die Verbindung zu dem auf dem lokalen Host laufenden named-Server herzustellen. Das mag zwar funktionieren, ich empfehle diese Vorgehensweise aber nicht. Indem Sie den Resolver in die Standardeinstellung schalten, geben Sie die Kontrolle über Ihr System auf und werden anfällig für die verschiedenen Techniken, die von den unterschiedlichen Systemen eingesetzt werden, um die Standardkonfiguration zu bestimmen. Aus diesen Gründen sollte die Resolver-Konfigurationsdatei auf jedem System eingerichtet werden, auf dem BIND läuft.

Die Resolver-Konfigurationsdatei

Die Konfigurationsdatei dokumentiert klar die Resolver-Konfiguration. Sie erlaubt die Angabe von bis zu drei Nameservern, von denen zwei als Backup dienen, falls der erste Server nicht antwortet. Sie definiert die Standard-Domain und verschiedene andere Verarbeitungsoptionen. Die Datei *resolv.conf* ist ein wichtiger Bestandteil der Konfiguration des Namensdienstes.

resolv.conf ist eine einfache, im Klartext vorliegende Datei. Es gibt systemspezifische Variationen bei den Befehlen, die in dieser Datei verwendet werden, aber die von den meisten Systemen unterstützten Einträge sind:

nameserver Adresse

Die *nameserver*-Einträge geben die IP-Adressen der Server an, die der Resolver nach Domain-Informationen abfragen soll. Die Nameserver werden in der Reihenfolge abgefragt, in der sie in der Datei auftauchen. Wird von einem Server keine Antwort empfangen, wird der nächste Server in der Liste probiert, so lange, bis die maximale Anzahl von Servern durchprobiert wurde.² Sind keine *nameserver*-Einträge in der Datei *resolv.conf* enthalten oder gibt es keine *resolv.conf*, dann werden alle Abfragen an den lokalen Host geschickt. Falls es jedoch eine Datei *resolv.conf* gibt und diese auch *nameserver*-Einträge enthält, wird der lokale Host *nur dann* abgefragt, wenn ein Eintrag auf ihn verweist. Geben Sie den lokalen Host mit seiner offiziellen IP-Adresse oder mit 0.0.0.0 an, nicht mit der Loopback-Adresse. Die Angabe der offiziellen Adresse vermeidet Probleme, die bei einigen Unix-Versionen aufgetreten sind, wenn die Loopback-Adresse eingesetzt wurde. Eine reine Resolver-Konfiguration enthält niemals einen *nameserver*-Eintrag, der auf den lokalen Host verweist.

² Bei den meisten BIND-Implementierungen ist drei die maximale Anzahl der Server, die durchprobiert wird.

domain Name

Der *domain*-Eintrag definiert den Standard-Domain-Namen. Der Resolver hängt den Standard-Domain-Namen an jeden Hostnamen an, der keinen Punkt enthält.³ Der so erweiterte Hostname wird dann in der Abfrage verwendet, die an den Nameserver geht. Wird zum Beispiel der Hostname *crab* (der keinen Punkt enthält) vom Resolver empfangen, wird der Standard-Domain-Name an *crab* angehängt, um die Abfrage zu formulieren. Wurde der *Name* eines *domain*-Eintrags mit *wrotethebook.com* angegeben, fragt der Resolver nach *crab.wrotethebook.com*. Ist die Umgebungsvariable *LOCALDOMAIN* gesetzt, überschreibt sie den *domain*-Eintrag, und der Wert von *LOCALDOMAIN* wird zur Erweiterung des Hostnamens verwendet.

search Domain ...

Der *search*-Eintrag definiert eine Reihe von Domains, die durchsucht werden, wenn ein Hostname keinen Punkt enthält. Nehmen wir zum Beispiel einmal den Eintrag *search essex.wrotethebook.com butler.wrotethebook.com*. Eine Abfrage nach dem Hostnamen *cookbook* wird zuerst als *cookbook.essex.wrotethebook.com* versucht. Führt das nicht zu einem erfolgreichen Ergebnis, fragt der Resolver nach *cookbook.butler.wrotethebook.com*. Falls auch diese Abfrage fehlschlägt, werden keine weiteren Versuche unternommen, den Hostnamen aufzulösen. Benutzen Sie entweder die Anweisung *search* oder die Anweisung *domain*. (Der Befehl *search* wird bevorzugt.) Benutzen Sie niemals beide in der gleichen Konfiguration. Ist die Umgebungsvariable *LOCALDOMAIN* gesetzt, überschreibt sie den *search*-Eintrag.

sortlist Netzwerk[/Netzmaske] ...

Adressen der im *sortlist*-Befehl angegebenen Netzwerke werden gegenüber anderen Adressen bevorzugt. Empfängt ein Resolver mehrere Adressen als Antwort auf eine Abfrage zu einem Multihomed-Host oder einem Router, sortiert er die Adressen so um, daß eine Adresse aus den in der *sortlist*-Anweisung enthaltenen Netzwerken vor den anderen Adressen plazierte wird. Normalerweise werden die Adressen vom Resolver in der Reihenfolge zurückgegeben, in der sie empfangen wurden.

Der *sortlist*-Befehl wird selten eingesetzt, weil er die Fähigkeit des Servers stört, Adressen für den Lastausgleich oder andere Zwecke umzusortieren. Die einzige Ausnahme ist hier, daß *sortlist* manchmal so konfiguriert wird, daß Adressen aus einem gemeinsam genutzten Netzwerk anderen Adressen vorgezogen werden. Ist der Computer, auf dem der Resolver läuft, also mit dem Netzwerk 172.16.0.0/16 verbunden, und eine der Adressen aus einer Mehrfachantwort stammt aus diesem Netzwerk, wird die Adresse aus 172.16.0.0 vor die anderen Adressen gesetzt.

options Option ...

Der *options*-Eintrag wird genutzt, um optionale Einstellungen für den Resolver zu treffen. Es gibt mehrere mögliche Optionen:⁴

- 3 Dies ist die übliche Art und Weise, auf die Standard-Domain-Namen verwendet werden, das läßt sich aber einstellen.
- 4 Diese Liste zeigt die Optionen auf Linux-Systemen, die BIND 8 ausführen. Die Solaris-Version von BIND 8 kennt die Optionen *rotate*, *no-check-names* oder *inet6* nicht.

`debug`

Aktiviert das Debugging, wodurch Debugging-Meldungen auf der Standardausgabe ausgegeben werden. `debug` funktioniert nur, wenn der Resolver mit der Option `-DDEBUG` kompiliert wurde, was meist nicht der Fall ist.

`ndots:n`

Gibt die Anzahl der Punkt an, die in einem Hostnamen enthalten sein müssen, um festzulegen, ob die Suchliste angewendet wird, bevor die Abfrage an den Nameserver gesandt wird. Der Standardwert ist 1. Daher wird an einen Hostnamen mit einem Punkt keine Domain angehängt, bevor er an den Nameserver geschickt wird. Ist `options ndots:2` festgelegt, wird einem Hostnamen mit einem Punkt die Suchlisten-Domain angehängt, bevor er weggeschickt wird; einem Hostnamen mit zwei oder mehr Punkten wird dagegen keine Domain angehängt.

`ndots` kann sich als nützlich erweisen, falls ein Bestandteil Ihrer Domain mit einer Top-Level-Domain verwechselt werden könnte und Ihre Benutzer Hostnamen immer auf diese Domain abkürzen. In diesem Fall würden Abfragen immer zuerst an die Root-Server zur Auflösung in der Top-Level-Domain geschickt werden, bevor sie schließlich zurück an Ihren lokalen Server gelangen. Es ist allerdings ein ziemlich schlechtes Betragen, die Root-Server wegen nichts zu belästigen. Benutzen Sie `ndots`, um die Resolver zu zwingen, die fraglichen Hostnamen mit Ihrem lokalen Domain-Namen zu erweitern, damit diese aufgelöst werden, bevor sie die Root-Server erreichen.

`timeout:n`

Setzt den initialen Abfrage-Timeout für den Resolver. Der Timeout für die erste Abfrage beträgt für alle Server standardmäßig fünf Sekunden. Bei der Solaris-8-Version von BIND lautet die Syntax dieser Option `retrans:n`.

`attempts:n`

Definiert, wie oft der Resolver eine Abfrage wiederholt. Der Vorgabewert ist 2, das bedeutet, der Resolver wiederholt eine Abfrage zweimal bei jedem Server in seiner Server-Liste, bevor er eine Fehlermeldung an die Anwendung zurückschickt. Bei der Solaris-8-Version von BIND lautet die Syntax dieser Option `retry:n`, der Vorgabewert ist 4.

`rotate`

Aktiviert die Round-Robin-Auswahl der Nameserver. Normalerweise sendet der Resolver die Abfrage an den ersten Server in der Nameserver-Liste. Er wendet sich erst dann an einen anderen Server, wenn der erste nicht antwortet. Die Option `rotate` weist den Resolver an, die Nameserver-Last gleichmäßig auf alle Server zu verteilen.

no-check-names

Deaktiviert die Überprüfung der Domain-Namen auf Übereinstimmung mit RFC 952, *DOD Internet Host Table Specification*. Domain-Namen, die einen Unterstrich (`_`), Nicht-ASCII-Zeichen oder ASCII-Steuerzeichen enthalten, werden standardmäßig als fehlerhaft angesehen. Setzen Sie diese Option ein, falls Sie mit Hostnamen arbeiten müssen, die einen Unterstrich enthalten.

inet6

Veranlaßt den Resolver, nach IPv6-Adressen zu fragen. Im heutigen Internet ist IPv4 die eingesetzte Version des Internet Protocol (IP). IPv4 verwendet 32-Bit-Adressen. IPv6 erweitert diese auf 128-Bit-Adressen.

Die am weitesten verbreitete Konfiguration der *resolv.conf* definiert den lokalen Domain-Namen als Suchliste, den lokalen Host als ersten Nameserver und setzt einen oder zwei Backup-Nameserver ein. Hier ein Beispiel für diese Konfiguration:

```
# Domain Name Resolver Konfigurationsdatei
#
search wrotethebook.com
# zuerst probieren wir es bei uns selbst
nameserver 172.16.12.2
# danach bei crab
nameserver 172.16.12.1
# und schließlich bei ora
nameserver 172.16.1.2
```

Dieses Beispiel basiert auf unserem imaginären Netzwerk, der Standard-Domain-Name ist deshalb *wrotethebook.com*. Die Konfiguration ist für *rodent* gedacht und gibt sich selbst als ersten Nameserver an. Die Backup-Server sind *crab* und *ora*. Die Konfiguration enthält keine Sortierliste oder irgendwelche Optionen, da diese nur selten genutzt werden. Es ist ein Beispiel für eine durchschnittliche Resolver-Konfiguration.

Konfiguration eines reinen Resolvers

Die Konfiguration eines reinen Resolvers ist sehr einfach. Sie ist mit der Durchschnittskonfiguration identisch, mit der Ausnahme, daß es keinen `nameserver`-Eintrag für das lokale System gibt. Eine Beispiel-*resolv.conf*-Datei für ein reines Resolver-System sehen Sie hier:

```
# Domain Name Resolver Konfigurationsdatei
#
search wrotethebook.com
# crab probieren
nameserver 172.16.12.1
# danach ora
nameserver 172.16.1.2
```

Die Konfiguration weist den Resolver an, alle Abfragen an *crab* zu schicken. Schlägt das fehl, soll *ora* probiert werden. Abfragen werden niemals lokal aufgelöst. Diese einfache *resolv.conf*-Datei ist alles, was Sie zur Konfiguration eines reinen Resolver-Systems benötigen.

Die Konfiguration von named

Während die Resolver-Konfiguration maximal eine Konfigurationsdatei verlangt, werden bei der Konfiguration von `named` mehrere Dateien benötigt. Die vollständige Liste der `named`-Konfigurationsdateien umfaßt:

Die Konfigurationsdatei

Legt allgemeine `named`-Parameter fest und verweist auf die Quellen mit DNS-Datenbankinformationen, die von diesem Server genutzt werden. Bei diesen Quellen kann es sich um lokal vorliegende Dateien oder um entfernte Server handeln. Diese Datei heißt üblicherweise `named.conf`.

Die Root-Hints-Datei

Verweist auf die Root-Zonen-Server. Einige gebräuchliche Namen für diese Datei sind `named.ca`, `db.cache`, `named.root` oder `root.ca`.

Die Localhost-Datei

Wird benutzt, um die Loopback-Adresse lokal aufzulösen. Im allgemeinen wird der Name `named.local` für diese Datei eingesetzt.

Die Datei der Forward-Mapping-Zone

Die Zonendatei, die Hostnamen auf IP-Adressen abbildet. Das ist die Datei, die die meisten Informationen über die Zone enthält. Damit es einfacher wird, diese Datei zu besprechen, bezeichnen wir sie hier im allgemeinen als *Zonendatei* und lassen die nähere Beschreibung »Forward-Mapping« einfach weg. Die Zonendatei erhält üblicherweise einen aussagekräftigen Namen wie `wrotethebook.com.hosts`, der angibt, die Daten welcher Zone in der Datei enthalten sind.

Die Datei der Reverse-Mapping-Zone

Die Zonendatei, die IP-Adressen auf Hostnamen abbildet. Auch hier verkürzen wir den Namen zur Vereinfachung; wir sprechen von der *Reverse-Zonendatei*. Die Reverse-Zonendatei erhält im allgemeinen einen aussagekräftigen Namen wie `172.16.rev`, der angibt, welche IP-Adresse von dieser Datei abgebildet wird.

All diese Dateien können von Ihnen beliebige gewünschte Namen erhalten. Sie sollten jedoch für Ihre Zonendateien aussagekräftige Namen wählen, die Namen `named.conf` und `named.local` für die Boot-Datei und die Loopback-Adreßdatei beibehalten sowie einen der bekannten Namen für die Root-Hints-Datei einstellen, um anderen die Wartung Ihres Systems zu vereinfachen. In den folgenden Abschnitten schauen wir uns die einzelnen Dateien an. Wir beginnen mit `named.conf`.

Die Datei `named.conf`

Die Datei `named.conf` zeigt `named` die Quellen der DNS-Informationen. Einige dieser Quellen sind lokale Dateien, andere sind entfernte Server. Sie müssen nur die Dateien erzeugen, die in den `master`- und `cache`-Anweisungen referenziert werden. Wir werden uns ein Beispiel für alle Arten von Dateien ansehen, die Sie möglicherweise anlegen müssen.

Die Struktur der Konfigurationsbefehle in *named.conf* ist ähnlich der Struktur der Programmiersprache C. Anweisungen enden mit einem Semikolon (;), Literale sind in Anführungszeichen eingeschlossen (") und zusammenhängende Elemente werden mit Hilfe geschweifter Klammern gruppiert ({}). Ein Kommentar kann zwischen /* und */ stehen wie ein Kommentar in C, er kann mit // beginnen wie ein Kommentar in C++ oder mit # wie ein Shell-Kommentar. Diese Beispiele verwenden die C++-Nomenklatur für Kommentare, Ihnen steht es aber natürlich frei, sich für einen der anderen Stile zu entscheiden.

Tabelle 8-1 faßt die grundlegenden Konfigurationsanweisungen von *named.conf* zusammen. Sie enthält gerade genug Informationen, um die Beispiele verstehen zu können. Nicht alle Konfigurationsbefehle der *named.conf* werden in den Beispielen verwendet, und Sie werden wahrscheinlich auch nicht alle Beispiele in Ihrer Konfiguration einsetzen. Die Befehle sind so konzipiert, daß sie das gesamte Spektrum der Konfigurationen abdecken, selbst die Konfiguration von Root-Servern. Wenn Sie weitere Details zu den Konfigurationsanweisungen der *named.conf* benötigen, finden Sie in Anhang C eine vollständige Beschreibung aller Befehle.

Tabelle 8-1: Konfigurationsbefehle der *named.conf*

Befehl	Funktion
acl	Definiert eine Zugriffskontrollliste der IP-Adressen.
include	Fügt eine andere Datei in die Konfigurationsdatei ein.
key	Definiert Sicherheitsschlüssel für die Authentifizierung.
logging	Definiert, was protokolliert werden soll und wo dies abgelegt wird.
options	Definiert globale Konfigurationsoptionen und Vorgabewerte.
server	Definiert die Charakteristik eines entfernten Servers.
zone	Definiert eine Zone.

Die Art und Weise, wie Sie die Datei *named.conf* konfigurieren, legt fest, ob der Nameserver als Master-Server einer Zone, als Slave-Server oder als Caching-Only-Server fungiert. Am besten werden Sie die unterschiedlichen Konfigurationen verstehen, wenn Sie sich die Beispiel-*named.conf*-Dateien anschauen. Beispiele für die einzelnen Konfigurationsarten finden Sie in den nächsten Abschnitten.

Konfiguration eines Caching-Only-Servers

Die Konfiguration eines Caching-Only-Servers ist einfach. Sie brauchen lediglich eine *named.conf*- und eine *named.ca*-Datei, obwohl normalerweise auch noch eine *named.local*-Datei eingesetzt wird. Eine mögliche *named.conf*-Datei für einen Caching-Only-Server sieht so aus:

```
$ cat /etc/named.conf
options {
    directory "/var/named";
};
```

```

//
// eine Caching-Only-Nameserver-Konfiguration
//
zone "." {
    type hint;
    file "named.ca";
};

zone "0.0.127.in-addr.arpa" {
    type master;
    file "named.local";
};

```

Die `options`-Anweisung legt das Standardverzeichnis für `named` fest. In der Beispieldatei ist dies `/var/named`. Alle nachfolgenden Dateiverweise in `named.conf` sind relativ zu diesem Verzeichnis zu sehen.

Die beiden `zone`-Anweisungen in dieser Caching-Only-Konfiguration gibt es in allen Server-Konfigurationen. Die erste `zone`-Anweisung definiert die Hints-Datei, die dem Nameserver helfen soll, während des Starts die Root-Server zu finden. Die zweite `zone`-Anweisung macht den Server zum Master für seine eigene Loopback-Adresse und besagt, daß die Informationen für die Loopback-Domain in der Datei `named.local` gespeichert sind. Die Loopback-Domain ist eine *in-addr.arpa*-Domain⁵ die die Adresse 127.0.0.1 auf den Namen `localhost` abbildet. Der Gedanke, die eigene Loopback-Adresse aufzulösen, erscheint den meisten Leuten sinnvoll, und die `named.conf`-Dateien sollten diesen Eintrag enthalten. Die Hints-Datei und die lokale Host-Datei werden zusammen mit der `named.conf`-Datei für jede Server-Konfiguration benötigt.⁶

Diese `zone`- und `options`-Anweisungen sind die einzigen Anweisungen, die in den meisten Caching-Only-Server-Konfigurationen verwendet werden, die `options`-Anweisung kann allerdings komplexer sein. Manchmal werden auch eine `forwarders`- und eine `forward only`-Option eingesetzt. Die Option `forwarders` veranlaßt den Caching-Only-Server, alle Abfragen, die er nicht mit Hilfe der in seinem Cache vorliegenden Daten auflösen kann, an bestimmte Server zu senden:

```

options {
    directory "/var/named";
    forwarders { 172.16.12.1; 172.16.1.2; };
};

```

Diese `forwarders`-Option leitet jede Abfrage, die nicht aus dem lokalen Cache heraus beantwortet werden kann, an 172.16.12.1 und 172.16.1.2 weiter. Die `forwarders`-Option baut auf ausgewählten Servern im lokalen Netzwerk einen großen DNS-Cache auf. Damit reduziert sich die Zahl der Abfragen, die ins WAN geschickt werden müssen, was sich vor allem dann als sinnvoll erweist, wenn Ihnen nur eine beschränkte Bandbreite zum WAN zur Verfügung steht oder Sie für deren Nutzung bezahlen müssen.

⁵ In Kapitel 4 finden Sie eine Beschreibung der *in-addr.arpa*-Domains.

⁶ BIND 8 fordert die Root-Hints-Datei, bei BIND 9 dagegen sind Hints bereits mit kompiliert und können verwendet werden, falls es keine Root-Hints-Datei gibt.

Ist die Netzwerkverbindung zur Außenwelt stark eingeschränkt, benutzen Sie die Option `forward only`, um den lokalen Server zu zwingen, immer den Forwarder zu benutzen:

```
options {
    directory "/var/named";
    forwarders { 172.16.12.1; 172.16.1.2; };
    forward only;
};
```

Steht diese Option in Ihrer Konfigurationsdatei, versucht der lokale Server nicht, eine Abfrage selbst aufzulösen, selbst dann nicht, wenn er keine Antwort von den Forwardern erhält.

Die Verwendung weiterer Optionen mit der Anweisung `options` ändert nichts daran, daß es sich um eine reine Cache-Server-Konfiguration handelt. Das wäre nur mit zusätzlichen Master- und Slave-zone-Befehlen der Fall.

Konfigurationen für Master- und Slave-Server

Die imaginäre *wrotethebook.com*-Domain ist die Grundlage für unsere beispielhaften Master- und Slave-Server-Konfigurationen. Nachfolgend sehen Sie die *named.conf*-Datei, mit der *crab* zum Master-Server der Domain *wrotethebook.com* gemacht wird:

```
options {
    directory "/var/named";
};

// eine Master-Nameserver-Konfiguration
//
zone "." {
    type hint;
    file "named.ca";
};

zone "0.0.127.in-addr.arpa" {
    type master;
    file "named.local";
};

zone "wrotethebook.com" {
    type master;
    file "wrotethebook.com.hosts";
};

zone "16.172.in-addr.arpa" {
    type master;
    file "172.16.rev";
};
```

Die `directory`-Option erspart uns bei den nachfolgenden Dateinamen einiges an Schreibarbeit. Sie teilt `named` mit, daß alle Dateinamen, die nicht mit `/` anfangen, relativ zum Verzeichnis */var/named* liegen, unabhängig davon, ob sie in der `named`-Konfiguration auf-

tauchen oder nicht. Diese Option weist `named` außerdem an, wohin die verschiedenen Dateien, wie etwa die Dump-Datei, zu schreiben sind.

Die ersten beiden `zone`-Anweisungen in der Beispielkonfiguration sind die `zone`-Anweisungen für die Loopback-Adresse und die Hints-Datei. Auf diese Anweisungen sind wir bereits früher im Zusammenhang mit den Caching-Only-Konfigurationen eingegangen. Sie haben immer die gleiche Funktion und sind in fast jeder Konfiguration zu finden.

Die erste neue `zone`-Anweisung legt fest, daß dies der Master-Server für die Domain *wrotethebook.com* ist und daß die Daten für diese Domain aus der Datei *wrotethebook.com.hosts* geladen werden.

Die zweite neue `zone`-Anweisung verweist auf die Datei, die IP-Adressen aus 172.16.0.0 auf Hostnamen abbildet. Diese Anweisung sagt aus, daß der lokale Server der Master-Server für die Reverse-Domain *16.172.in-addr.arpa* ist und die Daten für diese Domain aus der Datei *172.16.rev* geladen werden.

Die Konfiguration eines Slave-Servers unterscheidet sich von der eines Master-Servers nur in der Struktur der `zone`-Anweisungen. Die `zone`-Anweisungen des Slave-Servers verweisen auf entfernte Server als Quelle der Domain-Informationen und nicht auf lokal abgelegte Dateien. Außerdem definieren sie die Zone als `type slave`. Im Gegensatz zur `file`-Klausel einer Master-`zone`-Anweisung enthält die `file`-Klausel in einer Slave-`zone`-Anweisung den Namen einer lokalen Datei, in der die Informationen gespeichert werden, die vom entfernten Server empfangen wurden – und nicht den Namen einer Datei, aus der die Domain geladen wird. Die folgende *named.conf*-Datei konfiguriert *ora* als Slave-Server für die Domain *wrotethebook.com*:

```
options {
    directory "/var/named";
};

// eine Slave-Server-Konfiguration
//
zone "." {
    type hint;
    file "named.ca";
};

zone "0.0.127.in-addr.arpa" {
    type master;
    file "named.local";
};

zone "wrotethebook.com" {
    type slave;
    file "wrotethebook.hosts";
    masters { 172.16.12.1; };
};
```

```

zone "16.172.in-addr.arpa" {
    type slave;
    file "172.16.rev";
    masters { 172.16.12.1; };
};

```

Die erste zone-Anweisung macht aus diesem Server einen Slave-Server für die Domain *wrotethebook.com* (sie setzt den Typ auf *slave*). Die Anweisung teilt *named* mit, daß es die Daten für *wrotethebook.com* vom Server unter der IP-Adresse 172.16.12.1 herunterladen und in der Datei */var/named/wrotethebook.hosts* speichern soll. Wenn die Datei *wrotethebook.hosts* nicht existiert, erzeugt *named* sie, holt die Zonendaten vom entfernten Server und schreibt die Daten in die neu erzeugte Datei. Existiert die Datei bereits, prüft *named* beim entfernten Server, ob dessen Daten neueren Datums sind als die Daten in der Datei. Haben sich die Daten geändert, lädt *named* die aktualisierten Daten herunter und überschreibt den Inhalt der Datei mit den neuen Daten. Sind die Daten unverändert, lädt *named* den Inhalt der Datei auf der lokalen Festplatte und führt keinen Zonentransfer durch.⁷ Das Vorhalten einer Kopie der Datenbank auf der lokalen Festplatte macht den Transfer der Zonendatei bei jedem Neustart des lokalen Hosts unnötig. Die Zone muß nur dann übertragen werden, wenn sich die Daten geändert haben.

Die letzte zone-Anweisung in dieser Konfiguration besagt, daß der lokale Server außerdem ein Slave-Server für die Reverse-Domain *16.172.in-addr.arpa* ist und die Daten für diese Domain von 172.16.12.1 heruntergeladen werden sollen. Die Daten der Reverse-Domain werden lokal in einer Datei namens *172.16.rev* abgelegt. Dabei werden die gleichen Regeln befolgt, wie zuvor beim Erzeugen und Überschreiben von *wrotethebook.hosts* besprochen.

Standard-Resource-Records

Die gerade besprochenen und in Tabelle 8-1 aufgeführten Konfigurationsbefehle werden nur in der Datei *named.conf* eingesetzt. Alle anderen zur Konfiguration von *named* eingesetzten Dateien (die Zonendatei, die Reverse-Zone-Datei, *named.local* und *named.ca*) speichern DNS-Datenbank-Informationen. Diese Dateien haben das gleiche grundlegende Format und benutzen die gleiche Art von Datenbank-Records. Sie verwenden Standard-Resource-Records, sogenannte RRs. Diese sind in RFC 1033, dem *Domain Administrators Operations Guide*, und in anderen RFCs definiert. Tabelle 8-2 faßt alle Standard-Resource-Records zusammen, die in diesem Kapitel zum Einsatz kommen. Diese Records werden in Anhang C ausführlich behandelt.

Die Syntax der Resource Records wird in Anhang C beschrieben. Ein gewisses Verständnis für die Struktur dieser Records ist allerdings notwendig, um die Beispielkonfigurationsdateien in diesem Kapitel zu lesen.

⁷ Anhang C (im Abschnitt »SOA-Record (Start of Authority)«) zeigt, wie *named* feststellt, ob die Daten aktualisiert wurden.

Tabelle 8-2: Standard-Resource-Records

Resource Record (ausgeschriebener Name)	Record-Typ	Funktion
Start of Authority	SOA	Markiert den Anfang der Daten einer Zone und definiert Parameter, die die gesamte Zone betreffen.
Nameserver	NS	Bestimmt den Nameserver einer Domain.
Address	A	Konvertiert einen Hostnamen in eine Adresse.
Pointer	PTR	Konvertiert eine Adresse in einen Hostnamen.
Mail Exchange	MX	Bestimmt, wohin die Mail für einen angegebenen Domain-Namen ausgeliefert werden soll.
Canonical Name	CNAME	Definiert einen Alias für einen Hostnamen.
Text	TXT	Speichert beliebige Text-Strings ab.

Das Format der DNS-Resource-Records ist folgendermaßen:

`[name] [ttl] IN typ daten`

name

Der Name des Domain-Objekts, das über das Resource Record angesprochen wird. Es kann sich hierbei um einen einzelnen Host oder eine ganze Domain handeln. Der für das *name*-Feld eingegebene String ist relativ zur aktuellen Domain, es sei denn, er endet mit einem Punkt. Bleibt das *name*-Feld leer, d. h., enthält es nur Whitespace, wird das Record auf das zuletzt genannte Domain-Objekt angewandt. Falls beispielsweise dem A-Record für *rodent* ein MX-Record mit einem leeren *name*-Feld folgt, gelten sowohl das A-Record als auch das MX-Record für *rodent*.

ttl

Der TTL-Wert (Time-to-Live, also »Lebensdauer«) definiert die Zeitspanne in Sekunden, für die die Informationen dieses Resource Record im Cache eines entfernten Systems vorgehalten werden sollen. Dieses Feld bleibt üblicherweise leer, und es wird die Standard-*ttl* verwendet, die mittels der \$TTL-Direktive für die gesamte Zone gesetzt wird.⁸

IN

Identifiziert das Record als Internet-DNS-Resource-Record. Es gibt andere Record-Klassen, die aber selten eingesetzt werden. Neugierig? In Anhang C finden Sie die anderen, Nicht-Internet-Klassen.

type

Bestimmt die Art des Resource Records. Tabelle 8-2 gibt diese Record-Typen unter der Überschrift *Record-Typ* an. Sie müssen einen dieser Werte im Feld *type* festlegen.

data

Die für diesen Typ von Resource Record spezifischen Daten. Zum Beispiel enthält das Feld in einem A-Record die tatsächliche IP-Adresse.

⁸ Die Beschreibung der \$TTL-Direktive finden Sie weiter hinten in diesem Kapitel.

Weiter hinten in diesem Kapitel schauen wir uns die verbleibenden Konfigurationsdateien an. Denken Sie bei der Betrachtung dieser Dateien daran, daß alle Standard-Resource-Records in diesen Dateien dem oben beschriebenen Format folgen.

Der größte Teil einer Zonendatei ist aus Standard-Resource-Records zusammengesetzt. Darüber hinaus stellt BIND einige Zonendatei-Direktiven bereit, die verwendet werden, um eine DNS-Datenbank aufzubauen.

Zonendatei-Direktiven

BIND stellt vier Direktiven zur Verfügung, die die Konstruktion einer Zonendatei vereinfachen oder einen Wert definieren, der von den Resource Records in der Datei benutzt wird. Zwei der vier Direktiven sind Befehle, die die Konstruktion einer Zonendatei vereinfachen: \$INCLUDE und \$GENERATE. Zwei definieren Werte, die von den Resource Records benutzt werden: \$ORIGIN und \$TTL.

Die Direktive \$TTL

Die Direktive \$TTL definiert die Standard-TTL für Resource Records, bei denen keine explizite Lebensdauer angegeben ist. Die Zeit kann in Form der Anzahl an Sekunden oder als eine Kombination aus Zahlen und Buchstaben angegeben werden. Wenn Sie den Zeitraum von einer Woche als Standard-TTL im numerischen Format definieren wollen, schreiben Sie folgendes:

```
$TTL 604800
```

Eine Woche entspricht 604.800 Sekunden. Im alphanumerischen Format sieht eine Woche so aus:

```
$TTL 1w
```

Folgende Werte sind für das alphanumerische Format möglich:

- w für Woche
- d für Tag
- h für Stunde
- m für Minute
- s für Sekunde

Die Direktive \$ORIGIN

Die Direktive \$ORIGIN stellt den aktuellen Ursprung ein, also den Domain-Namen, der zur Vervollständigung aller relativen Domain-Namen verwendet wird. Ein relativer Domain-Name ist jeder Name, der nicht mit einem Punkt endet. Standardmäßig geht \$ORIGIN von dem Domain-Namen aus, der in der zone-Anweisung definiert ist. Benutzen Sie die Direktive \$ORIGIN, um die Einstellung zu ändern.

Die Direktive \$INCLUDE

Die Direktive \$INCLUDE liest eine externe Datei und fügt diese als Bestandteil der Zonendatei ein. Die externe Datei wird in die Zonendatei an der Stelle aufgenommen, an der die Direktive \$INCLUDE auftritt.

Die Direktive \$GENERATE

Die Direktive \$GENERATE wird benutzt, um eine Reihe von Resource Records zu erzeugen. Diese Resource Records sind nahezu identisch und unterscheiden sich nur um eine numerische Laufvariable. Zum Beispiel:

```
$ORIGIN 20.16.172.in-addr.arpa.  
$GENERATE 1-4 $ CNAME $.1to4
```

Dem Schlüsselwort \$GENERATE folgt ein Bereich von Records, die erzeugt werden sollen. In unserem Beispiel bewegt sich der Bereich von 1 bis 4. Hinter dem Bereich steht das Template der zu erzeugenden Resource Records. Das Template ist in diesem Fall \$ CNAME \$.1to4. Das \$-Zeichen im Template wird durch die aktuelle Laufvariable ersetzt. Im Beispiel läuft der Wert von 1 bis 4. Diese \$GENERATE-Direktive erzeugt die folgenden Resource Records:

```
1 CNAME 1.1to4  
2 CNAME 2.1to4  
3 CNAME 3.1to4  
4 CNAME 4.1to4
```

Vorausgesetzt, daß *20.16.172.in-addr.arpa.* der Wert des aktuellen Ursprungs ist, sind diese Resource Records identisch mit:

```
1.20.16.172.in-addr.arpa. CNAME 1.1to4.20.16.172.in-addr.arpa.  
2.20.16.172.in-addr.arpa. CNAME 2.1to4.20.16.172.in-addr.arpa.  
3.20.16.172.in-addr.arpa. CNAME 3.1to4.20.16.172.in-addr.arpa.  
4.20.16.172.in-addr.arpa. CNAME 4.1to4.20.16.172.in-addr.arpa.
```

Diese eigenartig aussehenden Records sind hilfreich für das Delegieren von reversen Subdomains. Auf das Delegieren von Domains gehen wir später ein.

Mit Ausnahme von *named.conf* werden alle BIND-Konfigurationsdateien aus Standard-Records und Direktiven zusammengesetzt. Alle vier verbleibenden Konfigurationsdateien sind Datenbankdateien. Zwei dieser Dateien, *named.ca* und *named.local*, werden ungeachtet des Server-Typs auf allen Servern verwendet.

Die Cache-Initialisierungsdatei

Die zone-Anweisung in *named.conf*, bei der der Typ auf *hints* gesetzt ist, verweist auf die Cache-Initialisierungsdatei. Jeder Server, der einen Cache pflegt, besitzt eine solche Datei. Sie enthält die Informationen, die beim Start des Nameservers zum Aufbau eines Caches mit Domain-Daten benötigt werden. Die Root-Domain wird in der zone-Anweisung durch einen einzelnen Punkt im Domain-Namen-Feld gekennzeichnet, da die Cache-Initialisierungsdatei die Namen und Adressen der Root-Server enthält.

Die Datei *named.ca* wird auch als »Hints«-Datei (Hinweisdatei) bezeichnet, da sie Hinweise enthält, die *named* verwendet, um den Cache zu initialisieren. Bei diesen Hinweisen handelt es sich um die Namen und Adressen der Root-Server. Die Hints-Datei hilft dem lokalen Server, während des Starts einen Root-Server zu finden. Sobald ein Root-Server ermittelt wurde, wird eine verbindliche Liste von Root-Servern von diesem Server heruntergeladen. Auf die Hinweise wird erst bei einem Neustart des lokalen Servers wieder zugegriffen. Die Informationen in der Datei *named.ca* werden nicht oft benutzt, sie sind aber wichtig für das Booten eines *named*-Servers.

Eine grundlegende *named.ca*-Datei enthält NS-Records, die die Root-Server benennen, und A-Records, die die Adressen der Root-Server bereitstellen. Hier sehen Sie ein Beispiel für eine *named.ca*-Datei:

```

;
.           3600000   IN   NS   A.ROOT-SERVERS.NET.
A.ROOT-SERVERS.NET. 3600000   IN   A   198.41.0.4
;
.           3600000   NS   B.ROOT-SERVERS.NET.
B.ROOT-SERVERS.NET. 3600000   IN   A   128.9.0.107
;
.           3600000   NS   C.ROOT-SERVERS.NET.
C.ROOT-SERVERS.NET. 3600000   IN   A   192.33.4.12
;
.           3600000   NS   D.ROOT-SERVERS.NET.
D.ROOT-SERVERS.NET. 3600000   IN   A   128.8.10.90
;
.           3600000   NS   E.ROOT-SERVERS.NET.
E.ROOT-SERVERS.NET. 3600000   IN   A   192.203.230.10
;
.           3600000   NS   F.ROOT-SERVERS.NET.
F.ROOT-SERVERS.NET. 3600000   IN   A   192.5.5.241
;
.           3600000   NS   G.ROOT-SERVERS.NET.
G.ROOT-SERVERS.NET. 3600000   IN   A   192.112.36.4
;
.           3600000   NS   H.ROOT-SERVERS.NET.
H.ROOT-SERVERS.NET. 3600000   IN   A   128.63.2.53
;
.           3600000   NS   I.ROOT-SERVERS.NET.
I.ROOT-SERVERS.NET. 3600000   IN   A   192.36.148.17
;
.           3600000   NS   J.ROOT-SERVERS.NET.
J.ROOT-SERVERS.NET. 3600000   IN   A   198.41.0.10
;
.           3600000   NS   K.ROOT-SERVERS.NET.
K.ROOT-SERVERS.NET. 3600000   IN   A   193.0.14.129
;
.           3600000   NS   L.ROOT-SERVERS.NET.
L.ROOT-SERVERS.NET. 3600000   IN   A   198.32.64.12
;
.           3600000   NS   M.ROOT-SERVERS.NET.
M.ROOT-SERVERS.NET. 3600000   IN   A   202.12.27.33

```

Diese Datei enthält nur Nameserver- und Adreß-Records. Jedes NS-Record gibt einen Nameserver für die Root-Domain (.) an. Das dazugehörige A-Record liefert die Adresse des jeweiligen Root-Servers. Der TTL-Wert für alle Records liegt bei 3.600.000 – ein sehr großer Wert, der ungefähr 42 Tagen entspricht.

Legen Sie die *named.ca*-Datei an, indem Sie die Datei *domain/named.root* über anonymes ftp von *ftp.rs.internic.net* herunterladen. Die dort gespeicherte Datei liegt in einem für Unix-Systeme geeigneten Format vor. Das folgende Beispiel zeigt den Superuser beim Download von *named.root* direkt in die Datei *named.ca* des lokalen Systems. Die Datei muß nicht einmal bearbeitet werden, sondern ist bereit für den Einsatz.

```
# ftp ftp.rs.internic.net
Connected to rs.internic.net.
220-****Welcome to the InterNIC Registration Host ****
****Login with username "anonymous"
****You may change directories to the following:
    policy      - Registration Policies
    templates   - Registration Templates
    netinfo     - NIC Information Files
    domain      - Root Domain Zone Files
220 And more!
Name (ftp.rs.internic.net:craig): anonymous
331 Guest login ok, send your complete e-mail address as password.
Password: craig@wrotethebook.com
230 Guest login ok, access restrictions apply.
Remote system type is Unix.
Using binary mode to transfer files.
ftp> get /domain/named.root /var/named/named.ca
local: /var/named/named.ca remote: /domain/named.root
200 PORT command successful.
150 Opening BINARY mode data connection for /domain/named.root (2769 bytes).
226 Transfer complete.
2769 bytes received in 0.998 secs (2.7 Kbytes/sec)
ftp> quit
221 Goodbye.
```

Laden Sie die Datei *named.root* alle paar Monate herunter, um Ihren Cache mit korrekten Informationen über die Root-Server zu versorgen. Ein fehlerhafter Root-Server-Eintrag kann Probleme mit Ihrem lokalen Server verursachen. Die obigen Daten waren zum Zeitpunkt der Veröffentlichung korrekt, können sich aber jederzeit ändern.

Ist Ihr System nicht mit dem Internet verbunden, kann es auch nicht mit den Root-Servern kommunizieren. Die Initialisierung Ihrer Hints-Datei mit den angegebenen Servern wäre sinnlos. In diesem Fall müssen Sie Ihre Hints-Datei mit Einträgen initialisieren, die auf die Haupt-Nameserver Ihres lokalen Netzwerks verweisen. Diese Server müssen ebenfalls so konfiguriert sein, daß sie Anfragen nach der »Root«-Domain beantworten können. Diese Root-Domain enthält aber nur NS-Records, die auf die Domain-Server in Ihrem lokalen Netzwerk verweisen. Nehmen Sie beispielsweise an, *wrotethebook.com* wäre nicht mit dem Internet verbunden und *crab* sowie *horseshoe* würden als Root-Server für diese isolierte Domain fungieren. *crab* wird in seiner *named.conf*-Datei zum Master-Server für die

Root-Domain erklärt. *horseshoe* dient als Slave-Server für die Root-Domain. Sie laden die Root aus einer Zonendatei, die mit einem SOA-Record beginnt, das *crab* als Server identifiziert und einen netzwerkeigenen Kontaktpunkt angibt. Auf das SOA-Record folgen NS- und A-Records, die besagen, daß *crab* und *horseshoe* die Autorität für die Root-Zone haben und die Domains *wrotethebook.com* und *16.172.in-addr.arpa* an die lokalen Nameserver delegiert werden, die diese Domains bedienen. (Wie die Delegation von Domains funktioniert, besprechen wir weiter hinten in diesem Kapitel.) Einzelheiten über diese Art der Konfiguration finden Sie in *DNS und BIND* von Liu und Albitz (O'Reilly Verlag Köln, 2001).

Die Datei *named.local*

Die Datei *named.local* wird verwendet, um die Adresse 127.0.0.1 (die »Loopback-Adresse«) auf den Namen *localhost* abzubilden. Es handelt sich um die Zonendatei für die Reverse-Domain *0.0.127.IN-ADDR.ARPA*. Da alle Systeme 127.0.0.1 als »Loopback«-Adresse benutzen, ist diese Datei praktisch auf allen Servern identisch. Hier ein Beispiel für *named.local*:

```
$TTL      86400
@         IN  SOA      crab.wrotethebook.com. alana.crab.wrotethebook.com. (
                        1                ; Serial
                        360000           ; Refresh alle 100 Stunden
                        3600             ; Retry nach 1 Stunde
                        3600000          ; Expire nach 1000 Stunden
                        3600             ; Negativ-Cache ist 1 Stunde
                        )
          IN  NS      crab.wrotethebook.com.
0         IN  PTR     loopback.
1         IN  PTR     localhost.
```

Die meisten Zonendateien beginnen so wie diese Datei mit einer \$TTL-Direktive. Diese Direktive setzt die Standard-TTL für alle Resource Records in dieser Zone. Sie kann in einem einzelnen Record überschrieben werden, indem für dieses Record eine spezielle TTL definiert wird.

Das SOA- und das NS-Record identifizieren die Zone und den Nameserver für die Zone. Das erste PTR-Record bildet das Netzwerk 127.0.0.0 auf den Namen *loopback* ab. Das Record ist eine Alternative zur Abbildung des Netzwerknamens in der Datei */etc/networks*. Das zweite PTR-Record stellt das Kernstück dieser Datei dar. Es bildet die Hostadresse 1 in Netzwerk 127.0.0 auf den Namen *localhost* ab.

Die Datenfelder des SOA-Records und das NS-Record, das den Hostnamen des Computers enthält, unterscheiden sich von System zu System. Das Beispiel-SOA-Record legt *crab.wrotethebook.com.* als den Server fest, der den Ursprung der Zone bildet. Die E-Mail-Adresse *alana.crab.wrotethebook.com.* ist die Kontaktstelle für alle Fragen, die die Zone betreffen. (Beachten Sie, daß in der E-Mail-Adresse in einem SOA-Record ein Punkt den Namen des Empfängers vom Hostnamen trennt: *alana* ist der Benutzer, und *crab.wrote*

thebook.com ist der Host. Die Domain-Namen enden mit einem Punkt, wodurch angegeben wird, daß sie voll qualifiziert sind und kein Standard-Domain-Name angehängt werden soll.) Das NS-Record enthält außerdem den Hostnamen des Computers. Passen Sie diese drei Felder an, und Sie können die gleiche Datei auf jedem Host einsetzen.

Die bislang behandelten Dateien *named.conf*, *named.ca* und *named.local* sind die einzigen Dateien, die Sie zur Konfiguration von Caching-Only- und Slave-Servern benötigen. Die meisten Ihrer Server werden nur diese Dateien nutzen, und die Dateien werden auf jedem Server nahezu identische Informationen enthalten. Die einfachste Möglichkeit, diese drei Dateien zu erzeugen, besteht darin, eine Beispieldatei zu kopieren und sie für Ihr System anzupassen. Die meisten Systeme werden mit Beispieldateien geliefert. Ist das bei Ihnen nicht der Fall, holen Sie sich von einem laufenden Server Beispiel-Konfigurationen.

Die verbleibenden *named*-Konfigurationsdateien sind komplexer, dafür ist auch die relative Anzahl von Systemen, die diese Dateien benötigen, gering. Nur der Master-Server braucht all die Konfigurationsdateien, und jede Zone sollte nur einen Master-Server besitzen.

Die Datei für die Reverse-Zone

Die Datei für die Reverse-Zone ist in ihrer Struktur ähnlich der Datei *named.local*. Beide Dateien übersetzen IP-Adressen in Hostnamen, deshalb enthalten beide Dateien PTR-Records.

Die Datei *172.16.rev* in unserem Beispiel ist die Reverse-Zone-Datei für die Domain *16.172.in-addr.arpa*. Der Domain-Administrator erzeugt diese Datei auf *crab*. Jeder andere Host, der diese Informationen benötigt, holt sie sich von dort.

```
$TTL 86400
;
;      Abbildung von Adressen auf Hostnamen.
;
@      IN      SOA      crab.wrotethebook.com. jan.crab.wrotethebook.com. (
                                2001061401 ; Serial
                                21600      ; Refresh
                                1800       ; Retry
                                604800    ; Expire
                                900 )      ; Negative Cache-TTL
                                IN      NS      crab.wrotethebook.com.
                                IN      NS      ora.wrotethebook.com.
                                IN      NS      bigserver.isp.com.
1.12   IN      PTR     crab.wrotethebook.com.
2.12   IN      PTR     rodent.wrotethebook.com.
3.12   IN      PTR     horseshoe.wrotethebook.com.
4.12   IN      PTR     jerboas.wrotethebook.com.
2.1    IN      PTR     ora.wrotethebook.com.
6      IN      NS      linuxuser.articles.wrotethebook.com.
      IN      NS      horseshoe.wrotethebook.com.
```

Wie alle Zonendateien ist das erste Resource Record in dieser Datei ein SOA-Record. Das @ im Namensfeld des SOA-Records verweist auf den aktuellen Ursprung. Da diese Zonendatei keine \$ORIGIN-Direktive enthält, die explizit den Ursprung definiert, ist der aktuelle Ursprung die Domain *16.172.in-addr.arpa*, die durch die zone-Anweisung für diese Datei in unserer Beispiel-*named.conf*-Datei definiert wird:

```
zone "16.172.in-addr.arpa" {
    type master;
    file "172.16.rev";
};
```

Das @ im SOA-Record erlaubt der zone-Anweisung, die Zonendatei-Domain zu definieren. Das gleiche SOA-Record wird in jeder Zone eingesetzt; es verweist immer auf den richtigen Domain-Namen, weil es auf die Domain verweist, die für diese spezielle Zonendatei in *named.conf* definiert ist. Ändern Sie den Hostnamen (*crab.wrotethebook.com.*) und die E-Mail-Adresse des Verwalters (*jan.crab.wrotethebook.com.*), und benutzen Sie dieses SOA-Record in Ihren Zonendateien.

Die NS-Records, die dem SOA-Record folgen, definieren die Nameserver für die Domain. Im allgemeinen werden die Nameserver unmittelbar nach dem SOA-Record aufgeführt und enthalten ein leeres Namensfeld. Erinnern Sie sich daran, daß ein leeres Namensfeld bedeutet, daß der letzte Domain-Name immer noch gilt. Die NS-Records gelten also für die gleiche Domain wie das SOA-Record.

PTR-Records dominieren die Datei für die Reverse-Zone, weil sie zur Abbildung von Adressen auf Hostnamen eingesetzt werden. Die PTR-Records in unserem Beispiel übernehmen die Adresse-auf-Name-Abbildung für die Hosts 12.1, 12.2, 12.3, 12.4 und 2.1 im Netzwerk 172.16. Da sie nicht mit Punkten enden, sind die Werte in den Namensfeldern dieser PTR-Records relativ zur aktuellen Domain zu sehen. Der Wert 3.12 wird beispielsweise als *3.12.16.172.in-addr.arpa* interpretiert. Der Hostname im Datenfeld des PTR-Records ist voll qualifiziert, um zu verhindern, daß er relativ zum aktuellen Domain-Namen ist (deshalb endet er mit einem Punkt). Mit Hilfe der Information in diesem PTR übersetzt *named* *3.12.16.172.in-addr.arpa* in *horseshoe.wrotethebook.com*.

Die beiden letzten Zeilen in dieser Datei sind zusätzliche NS-Records. Wie bei jeder Domain können in einer *in-addr.arpa*-Domain Subdomains erzeugt werden. Das erledigen die beiden letzten NS-Records. Diese NS-Records verweisen auf *horseshoe* und *linux-user* als Nameserver für die Subdomain *6.16.172.in-addr.arpa*. Jede Anfrage nach Informationen in der Subdomain *6.16.172.in-addr.arpa* wird an sie weitergereicht. NS-Records, die auf die Server für eine Subdomain verweisen, müssen in der darüberliegenden Domain plaziert werden, bevor Sie diese Subdomain benutzen können.

Domain-Namen und IP-Adressen sind nicht gleich und haben auch nicht die gleiche Struktur. Wenn eine IP-Adresse in einen *in-addr.arpa*-Domain-Namen umgewandelt wird, werden die vier Byte der Adresse als vier eigene Bestandteile des Namens behandelt. In Wirklichkeit besteht die IP-Adresse aus 32 aufeinanderfolgenden Bit, nicht aus vier getrennten Byte. Subnetze teilen den IP-Adreßraum auf, und Subnetz-Masken sind

bit-orientiert, weshalb sie nicht auf die Byte-Grenzen beschränkt sind. Werden Subdomains auf Byte-Grenzen beschränkt, sind sie weniger flexibel als die Subnetze, die sie unterstützen müssen. Unsere Beispiel-Domain *in-addr.arpa* delegiert die Subdomain an einer Byte-Grenze, wodurch jedes Byte der Adresse als eigenständiger »Name« betrachtet wird. Dies ist die einfachste Art der Delegierung einer Reverse-Subdomain, möglicherweise ist sie aber für Ihre Situation nicht flexibel genug.

Das weiter vorn gezeigte \$GENERATE-Beispiel hilft bei der Schaffung flexiblerer Delegierungen für Reverse-Domains. Die Direktive \$GENERATE erzeugt CNAME-Records, um einen Adreßbereich in einer *in-addr.arpa*-Domain auf eine andere Domain abzubilden, die flexiblere Domain-Namen-Regeln besitzt. Echte *in-addr.arpa*-Domain-Namen *müssen* aus vier numerischen Feldern bestehen, die den vier Byte der IP-Adresse entsprechen, gefolgt vom String *in-addr.arpa*. Im \$GENERATE-Beispiel haben wir diese Namen auf längere Namen abgebildet, die uns eine größere Flexibilität boten. Hier ist ein umfangreicheres Beispiel des \$GENERATE-Befehls:

```
$ORIGIN 30.168.192.in-addr.arpa.  
$GENERATE 0-63 $ CNAME $.1ST64  
$GENERATE 63-127 $ CNAME $.2ND64  
$GENERATE 128-191 $ CNAME $.3RD64  
$GENERATE 192-255 $ CNAME $.4TH64
```

Diese vier \$GENERATE-Befehle bilden die 256 numerischen Namen in der Domain *30.168.192.in-addr.arpa* auf vier andere Domains ab, die jeweils aus 64 numerischen Namen bestehen. Wenn ein entfernter Server das PTR-Record für *52.30.168.192.in-addr.arpa* sucht, wird ihm gesagt, daß der kanonische Name für diesen Host *52.1st64.30.168.192.in-addr.arpa* lautet und der Server das PTR-Record für diesen Host von dem Server für die Domain *1st64.30.168.192.in-addr.arpa* holen muß. Im Prinzip erlaubt es uns die Direktive \$GENERATE, die einzelne *30.168.192.in-addr.arpa*-Domain in mehrere Domains aufzuteilen. Sobald die Aufteilung erfolgt ist, kann jedes Teil an einen anderen Server delegiert werden.

Die Subdomain-Delegierung kann Reverse-Domains komplizierter machen.⁹ In den meisten Fällen sind jedoch Dateien für die Reverse-Zone einfacher als die Datei für die Forward-Mapping-Zone.

Die Datei für die Forward-Mapping-Zone

Die Datei für die Forward-Mapping-Zone enthält den größten Teil der Domain-Informationen. Diese Datei wandelt Hostnamen in IP-Adressen um, A-Records sind also vorherrschend. Sie enthält aber auch MX-, CNAME- und andere Records. Die Zonendatei wird ebenso wie die Reverse-Zone-Datei nur auf dem Master-Server erzeugt. Alle anderen Server beziehen diese Informationen vom Master-Server.

⁹ Noch kompliziertere Beispiele finden Sie in *DNS und BIND* von Albitz und Liu.

```

$TTL 86400
;
; Adressen und andere Host-Informationen.
;
@ IN SOA crab.wrotethebook.com. jan.crab.wrotethebook.com. (
    2001061401 ; Serial
    21600 ; Refresh
    1800 ; Retry
    604800 ; Expire
    900 ) ; Negative Cache-TTL
; Definition der Nameserver und der Mailserver
    IN NS crab.wrotethebook.com.
    IN NS ora.wrotethebook.com.
    IN NS bigserver.isp.com.
    IN MX 10 crab.wrotethebook.com.
    IN MX 20 horseshoe.wrotethebook.com.
;
; Definition von localhost
;
localhost IN A 127.0.0.1
;
; Definition der Hosts in dieser Zone
;
crab IN A 172.16.12.1
loghost IN CNAME crab.wrotethebook.com.
rodent IN A 172.16.12.2
    IN MX 5 crab.wrotethebook.com.
mouse IN CNAME rodent.wrotethebook.com.
horseshoe IN A 172.16.12.3
jerboas IN A 172.16.12.4
ora IN A 172.16.1.2
; Die Host-Tabelle enthält sowohl Host- als auch Gateway-Einträge für 10.104.0.19
wtb-gw IN A 10.104.0.19
;
; Glue-Records für Server in dieser Domain
;
linuxmag.articles IN A 172.16.18.15
24seven.events IN A 172.16.6.1
;
; Definition der Subdomains
;
articles IN NS linuxmag.articles.wrotethebook.com.
    IN NS horseshoe.wrotethebook.com.
events IN NS 24seven.events.wrotethebook.com.
    IN NS linuxmag.articles.wrotethebook.com.

```

Ebenso wie die Reverse-Zone-Datei beginnt auch die Zonendatei mit einem SOA-Record und einigen NS-Records, die die Domain und ihre Server definieren. Die Zonendatei enthält allerdings eine größere Vielfalt an Resource-Records als eine Datei für die Reverse-Zone. Wir werden uns die einzelnen Records in der Reihenfolge ihres Auftretens in der Beispieldatei anschauen. Sie können dann selbst anhand dieser Datei vorgehen.

Das erste MX-Record bestimmt einen Mailserver für die gesamte Domain. Dieses Record besagt, daß *crab* der Mailserver für *wrotethebook.com* ist und einen Präferenzwert von 10 besitzt. An *benutzer@wrotethebook.com* adressierte Mail wird zur Auslieferung an *crab* umgeleitet. Damit *crab* die E-Mails erfolgreich ausliefern kann, muß er natürlich richtig als Mailserver konfiguriert sein. Das MX-Record ist nur ein Teil des Ganzen. Wir behandeln die Konfiguration von *sendmail* in Kapitel 10.

Das zweite MX-Record gibt *horseshoe* als Mailserver für *wrotethebook.com* mit einem Präferenzwert von 20 an. Präferenzwerte erlauben die Definition alternativer Mailserver. Je niedriger der Präferenzwert ist, desto »besser« ist der Server. Unsere beiden MX-Records besagen also, daß die an die Domain *wrotethebook.com* gerichtete Mail zuerst an *crab* geschickt werden soll. Ist *crab* nicht zu erreichen, soll versucht werden, die Mail an *horseshoe* zu senden. Statt von einem einzigen Mailserver abhängig zu sein, erlauben Präferenzwerte die Nutzung von Backup-Servern. Ist der Haupt-Mailserver nicht erreichbar, wird die Mail für diese Domain statt dessen an einen der Backup-Server geschickt.

Diese beispielhaften MX-Records leiten an *wrotethebook.com* adressierte Mail um, E-Mail für *benutzer@jerboas.wrotethebook.com* wird weiterhin direkt an *jerboas.wrotethebook.com* geschickt – nicht an *crab* oder *horseshoe*. Diese Konfiguration erlaubt denjenigen, die das wünschen, eine vereinfachte Mail-Adressierung in der Form *benutzer@wrotethebook.com*, gleichzeitig ist es aber auch jedem möglich, die Mails direkt an einzelne Hosts zu verschicken.

Das erste A-Record in diesem Beispiel definiert die Adresse für *localhost*. Es ist das Gegenstück zum PTR-Eintrag in der Datei *named.local*. Damit wird Benutzern in der Domain *wrotethebook.com* die Auflösung des Namens *localhost* in die Adresse 127.0.0.1 durch den lokalen Nameserver ermöglicht.

Das nächste A-Record definiert die IP-Adresse für *crab*, den Master-Server für diese Domain. Diesem A-Record folgt ein CNAME-Record, das *loghost* als Alias für *crab* festlegt.

Dem A-Record von *rodent* folgen ein MX- und ein CNAME-Record. (Beachten Sie, daß die Records, die zu einem einzelnen Host gehören, zusammengefaßt sind. Dies ist die gebräuchlichste Struktur in einer Zonendatei.) Das MX-Record von *rodent* leitet alle Mails, die an *benutzer@rodent.wrotethebook.com* adressiert sind, an *crab* weiter. Dieses MX-Record ist notwendig, weil die MX-Records am Anfang der Zonendatei die Mail nur dann weiterleiten, wenn sie an *benutzer@wrotethebook.com* gerichtet ist. Wollen Sie nun, daß auch die für *rodent* gedachte Mail umgeleitet wird, brauchen Sie ein »rodent-spezifisches« MX-Record.

Das Namensfeld des CNAME-Records enthält einen Alias für den offiziellen Hostnamen. Der offizielle Name, der sogenannte *kanonische Name*, ist im Datenfeld des Records angegeben. Wegen dieser Records kann auf *crab* auch über den Namen *loghost* und auf *rodent* über den Namen *mouse* zugegriffen werden. Der Alias *loghost* ist ein generischer Hostname, mit dessen Hilfe die Ausgabe des *syslogd* an *crab* gerichtet wird.¹⁰ Hostnamen-

¹⁰ In Kapitel 3 finden Sie eine weitergehende Diskussion generischer Hostnamen.

Aliase dürfen *nicht* in anderen Resource Records benutzt werden.¹¹ Verwenden Sie beispielsweise keinen Alias als Namen eines Mailservers in einem MX-Record. Benutzen Sie *ausschließlich* den kanonischen (offiziellen) Namen, der in einem A-Record definiert ist.

Ihre Zonendatei könnte viel größer sein als das vorgestellte Beispiel, sie würde aber im Prinzip die gleichen Records enthalten. Falls Sie die Namen und Adressen der Hosts in Ihrer Domain kennen, besitzen Sie bereits die meisten der Informationen, die zur Erstellung der named-Konfiguration notwendig sind.

Den named-Prozeß kontrollieren

Nachdem Sie die *named.conf*-Datei und die erforderlichen Zonendateien zusammengestellt haben, starten Sie named. named wird üblicherweise während des Bootens durch ein Start-Skript gestartet. Auf einem Solaris-8-System erfolgt der Start von named durch das Skript */etc/init.d/inetsvc*. Auf einem Red-Hat-Linux-System heißt das Skript zum Start von named */etc/rc.d/init.d/named*. Das Red-Hat-Skript kann auf der Kommandozeile mit optionalen Argumenten ausgeführt werden. Zum Beispiel können Sie auf einem Red-Hat-System den folgenden Befehl einsetzen, um den Nameserver zu stoppen:

```
# /etc/rc.d/init.d/named stop
```

Um den Namensdienst wieder aufzunehmen, setzen Sie folgenden Befehl ein:

```
# /etc/rc.d/init.d/named start
```

Start-Skripte tun zwar ihren Dienst, allerdings ist das named-Kontrollprogramm (*ndc*) ein effektiveres Werkzeug zur Verwaltung des named-Prozesses. Es wird mit BIND 8 geliefert und bietet eine Vielzahl an Funktionen, die Sie bei der Verwaltung des named unterstützen. BIND 9 besitzt ein ähnliches Werkzeug namens *rndc*. In Tabelle 8-3 sehen Sie die Optionen von *ndc* sowie ihre jeweiligen Aufgaben.¹²

Tabelle 8-3: *ndc*-Optionen

Option	Funktion
status	Gibt den Prozeßstatus von named aus.
dumpdb	Schreibt den Cache in <i>named_dump.db</i> . ^a
reload	Lädt den Nameserver erneut.
stats	Schreibt Statistiken in <i>named.stats</i> .
trace	Aktiviert das Tracing nach <i>named.run</i> .
notrace	Deaktiviert das Tracing und schließt <i>named.run</i> .
querylog	Ein- und Ausschalten der Abfrageprotokollierung, wobei alle eingehenden Abfragen durch <i>syslogd</i> protokolliert werden.
start	Startet named.

¹¹ In Anhang C finden Sie zusätzliche Informationen über den Einsatz von CNAME-Records in der Zonendatendatei.

¹² Zu dem Zeitpunkt, als dies geschrieben wurde, waren die Befehle *status*, *trace* und *restart* noch nicht für *rndc* implementiert.

Tabelle 8-3: *ndc*-Optionen (Fortsetzung)

Option	Funktion
stop	Stoppt named.
restart	Stoppt den aktuellen named-Prozeß und startet einen neuen.

- a. Diese Datei wird in dem Verzeichnis gespeichert, das durch die *directory*-Option in der *named.conf*-Datei angegeben wird.

ndc-Optionen sind einfach zu verstehen und einzusetzen. Die folgenden Befehle würden den *named*-Prozeß stoppen und dann neu starten:

```
# ndc stop  
# ndc start  
new pid is 795
```

Diese Befehlsfolge geht davon aus, daß zwischen dem Stoppen des alten *named*-Prozesses und dem Starten eines neuen eine gewisse Zeitspanne verstreicht. Falls Sie den *named*-Prozeß wirklich schnell beenden und neu starten wollen, benutzen Sie die Option *restart*:

```
# ndc restart  
new pid is 798
```

Wenn Sie *named* das erste Mal ausführen, achten Sie auf Fehlermeldungen. *named* protokolliert Fehler in der Datei *messages*.¹³ Läuft *named* dann zu Ihrer Zufriedenheit, verwenden Sie *nslookup*, um den Nameserver abzufragen und festzustellen, ob er die richtigen Informationen bereitstellt.

Arbeiten mit *nslookup*

nslookup ist ein Debugging-Werkzeug, das als Teil des BIND-Software-Pakets geliefert wird. Es erlaubt die direkte Abfrage eines Nameservers und aller dem DNS-System bekannten Informationen. Dabei ist es sehr hilfreich, wenn es darum geht zu ermitteln, ob der Server richtig läuft und sauber konfiguriert ist. Es ist aber auch zur Abfrage von Informationen geeignet, die von entfernten Servern angeboten werden.

Das *nslookup*-Programm ist so konzipiert, daß es Abfragen entweder interaktiv oder direkt über die Kommandozeile verarbeitet. Nachfolgend sehen Sie ein Beispiel für die Nutzung von *nslookup* über die Kommandozeile. Es wird die IP-Adresse eines Hosts abgefragt:

```
% nslookup crab.wrotethebook.com  
Server: rodent.wrotethebook.com  
Address: 172.16.12.2  
  
Name: crab.wrotethebook.com  
Address: 172.16.12.1
```

¹³ Diese Datei finden Sie in */usr/adm/messages* auf unserem Solaris-System und in */var/log/messages* auf unserem Red-Hat-System. Auf Ihrem System kann sie sich auch woanders befinden. Schauen Sie in Ihrer Dokumentation nach.

Hier fragt ein Benutzer nslookup nach der Adresse von *crab.wrotethebook.com*. nslookup gibt den Namen und die Adresse des Servers aus, der die Anfrage auflösen soll, und zeigt anschließend die Antwort auf die Anfrage. Das ist nützlich, weit häufiger wird nslookup jedoch interaktiv benutzt.

Die wirkliche Stärke von nslookup zeigt sich im interaktiven Modus. Um den interaktiven Modus zu starten, geben Sie ohne weitere Argumente auf der Kommandozeile nslookup ein. Sie beenden eine interaktive Sitzung, indem Sie Strg-D (^D) tippen oder den Befehl exit am nslookup-Prompt eingeben. Im interaktiven Modus sieht die vorgestellte Anfrage so aus:

```
% nslookup
Default Server:  rodent.wrotethebook.com
Address:  172.16.12.2

> crab.wrotethebook.com
Server:  rodent.wrotethebook.com
Address:  172.16.12.2

Name:   crab.wrotethebook.com
Address: 172.16.12.1
> ^D
```

nslookup fragt standardmäßig nach A-Records. Sie können aber den Befehl set type verwenden, um die Anfrage nach einem anderen Typ von Resource Record oder dem speziellen Typ ANY durchzuführen. ANY wird eingesetzt, um alle verfügbaren Resource Records für den angegebenen Host zu erfragen.¹⁴

Das folgende Beispiel sucht die MX-Records für *crab* und *rodent* heraus. Beachten Sie, daß der Abfragetyp MX bleibt, sobald er einmal festgelegt wurde. Er kehrt nicht wieder zum Standardwert A-Record zurück. Zum Zurücksetzen ist ein weiterer set-type-Befehl notwendig.

```
% nslookup
Default Server:  rodent.wrotethebook.com
Address:  172.16.12.2

> set type=MX
> crab.wrotethebook.com
Server:  rodent.wrotethebook.com
Address:  172.16.12.2

crab.wrotethebook.com  preference = 5, mail exchanger = crab.wrotethebook.com
crab.wrotethebook.com  inet address = 172.16.12.1

> rodent.wrotethebook.com
Server:  rodent.wrotethebook.com
Address:  172.16.12.2
```

¹⁴ »Alle verfügbaren« Records kann entsprechend dem antwortenden Server variieren. Ein Server, der die Autorität über die Zone besitzt, die die Records des Hosts enthält, antwortet mit allen Records. Ein nicht-autoritativer Server, der Informationen über den Host in seinem Cache abgelegt hat, antwortet mit den Records, die in seinem Cache zur Verfügung stehen. Das sind möglicherweise nicht alle Records, die zu dem Host gehören.

```
rodent.wrotethebook.com preference = 5, mail exchanger = rodent.wrotethebook.com
rodent.wrotethebook.com inet address = 172.16.12.2
> exit
```

Sie können mit dem `server`-Befehl steuern, welcher Server zur Auflösung der Anfragen genutzt wird. Das erweist sich vor allem dann als nützlich, wenn Sie sich direkt an einen autoritativen Server wenden wollen, um Informationen zu erlangen. Im folgenden Beispiel sehen Sie, wie das funktioniert. Dieses Beispiel enthält sogar einige interessante Befehle:

- Zuerst setzen wir `set type=NS` und beziehen die NS-Records für die Domain *zoo.edu*.
- Aus den Informationen, die wir bei dieser Anfrage erhalten haben, wählen wir einen Server aus und verwenden dann den Befehl `server`, um `nslookup` auf diesen Server zu richten.
- Als nächstes setzen wir mit dem Befehl `set domain` die Standard-Domain auf *zoo.edu*. `nslookup` benutzt diesen Standard-Domain-Namen, um die Hostnamen in seinen Anfragen auf die gleiche Weise zu erweitern, wie das auch der Resolver mit dem in *resolv.conf* definierten Standard-Domain-Namen macht.
- Wir setzen den Anfragetyp auf ANY. Wenn der Typ nicht zurückgesetzt wird, fragt `nslookup` weiterhin nach NS-Records.
- Schließlich fragen wir nach Informationen über den Host *tiger.zoo.edu*. Da als Standard-Domain *zoo.edu* eingestellt ist, geben wir am Prompt einfach *tiger* ein.

Das Beispiel sieht so aus:

```
% nslookup
Default Server: rodent.wrotethebook.com
Address: 172.16.12.2

> set type=NS
> zoo.edu
Server: rodent.wrotethebook.com
Address: 172.16.12.2

Non-authoritative answer:
zoo.edu nameserver = NOC.ZOO.EDU
zoo.edu nameserver = NI.ZOO.EDU
zoo.edu nameserver = NAMESERVER.AGENCY.GOV
Authoritative answers can be found from:
NOC.ZOO.EDU inet address = 172.28.2.200
NI.ZOO.EDU inet address = 172.28.2.240
NAMESERVER.AGENCY.GOV inet address = 172.21.18.31
> server NOC.ZOO.EDU
Default Server: NOC.ZOO.EDU
Address: 172.28.2.200

> set domain=zoo.edu
> set type=any
> tiger
Server: NOC.ZOO.EDU
```

```

Address: 172.28.2.200

tiger.zoo.edu  inet address = 172.28.172.8
tiger.zoo.edu  preference = 10, mail exchanger = tiger.ZOO.EDU
tiger.zoo.edu  CPU=ALPHA OS=Unix
tiger.zoo.edu  inet address = 172.28.172.8, protocol = 6
                7 21 23 25 79
tiger.ZOO.EDU  inet address = 172.28.172.8
> exit

```

Das letzte Beispiel zeigt, wie Sie eine ganze Domain von einem autoritativen Server herunterladen und auf Ihrem lokalen System untersuchen. Der Befehl `ls` fordert einen Zonentransfer an und gibt den Inhalt der empfangenen Zone aus.¹⁵ Ist die Zonendatei länger als nur ein paar Zeilen, leiten Sie die Ausgabe in eine Datei um und verwenden den Befehl `view`, um den Inhalt der Datei zu betrachten. (`view` sortiert eine Datei und gibt sie mit Hilfe des Unix-Befehls `more` aus.) Die Kombination aus `ls` und `view` ist hilfreich, wenn Sie einen entfernten Hostnamen untersuchen. In diesem Beispiel bezieht der Befehl `ls` die Zone `big.com` und speichert die Informationen in der Datei `temp.file`. Anschließend wird `view` benutzt, um `temp.file` zu untersuchen.

```

rodent% nslookup
Default Server: rodent.wrotethebook.com
Address: 172.16.12.2

> server minerals.big.com
Default Server: minerals.big.com
Address: 192.168.20.1

> ls big.com > temp.file
[minerals.big.com]
#####
Received 406 records.
> view temp.file
acmite             192.168.20.28
adamite            192.168.20.29
adelite            192.168.20.11
agate              192.168.20.30
alabaster          192.168.20.31
albite             192.168.20.32
allanite           192.168.20.20
altaite            192.168.20.33
alum               192.168.20.35
aluminum           192.168.20.8
amaranth           192.168.20.85
amethyst           192.168.20.36
andorite           192.168.20.37
apatite            192.168.20.38
beryl              192.168.20.23
--More--q
> exit

```

¹⁵ Aus Sicherheitsgründen antworten viele Nameserver nicht auf den Befehl `ls`. Die Option `allow-transfer` in Anhang C zeigt Ihnen, wie Sie den Zugriff auf Zonentransfers beschränken.

Diese Beispiele zeigen, daß `nslookup` Ihnen folgendes erlaubt:

- Abfragen jedes Typs von Standard-Resource-Record
- Direkte Abfragen der autoritativen Server einer Domain
- Herunterladen der vollständigen Domain-Daten in eine Datei zur späteren Untersuchung

Weitere Möglichkeiten von `nslookup` zeigt Ihnen der Befehl `help`. Aktivieren Sie das Debugging (mit `set debug`) und untersuchen Sie die zusätzlich gewonnenen Informationen. Beim Herumspielen mit diesem Werkzeug werden Sie viele nützliche Funktionen entdecken.

Zusammenfassung

Das Domain Name System (DNS) ist ein sehr wichtiger Benutzerdienst, der auf jedem System, das an das Internet angeschlossen ist, bereitgestellt werden sollte. Die große Mehrheit der Unix-Implementierungen des DNS basiert auf der BIND-Software (Berkeley Internet Name Domain). BIND stellt sowohl einen DNS-Client als auch einen DNS-Server zur Verfügung.

Der BIND-Client führt Namensanfragen aus und ist in Form von Bibliotheksroutinen implementiert. Man nennt ihn Resolver. Der Resolver wird in der Datei `resolv.conf` konfiguriert. Alle Systeme führen den Resolver aus.

Der BIND-Server beantwortet die Namensanfragen und läuft als Dämon. Er wird als `named` bezeichnet. `named` wird mit Hilfe der Datei `named.conf` konfiguriert, die definiert, woher der Server die DNS-Datenbankinformationen bezieht und welche Art von Server konfiguriert wird. Es gibt folgende Server-Arten: Master, Slave und Cache. Da alle Server auch Cache-Server sind, umfaßt eine einzige Konfiguration oft mehr als einen Server-Typ.

Die Original-DNS-Datenbankdateien sind auf dem Master-Server zu finden. Die DNS-Datenbankdatei heißt auch Zonendatei. Die Zonendatei wird aus Standard-Resource-Records (RRs) aufgebaut, die in RFCs definiert sind. Die RRs haben eine gemeinsame Struktur und werden verwendet, um alle DNS-Datenbankinformationen zu definieren.

Der DNS-Server kann mit `nslookup` getestet werden. Dieses Testwerkzeug ist im BIND-Paket enthalten.

In diesem Kapitel haben wir gesehen, wie DNS konfiguriert und getestet wird. Im nächsten Kapitel konfigurieren wir verschiedene andere Dienste.