DNS HOWTO

Nicolai Langfeldt (janl@math.uio.no) und Thomas »Balu« Walter (dlhp@b-a-l-u.de) v2.2, 18. September 2000

Dieses Dokument beschreibt, wie man mit wenig Aufwand ein DNS-Administrator wird.

Inhaltsverzeichnis

1 Vorwort

1.1 Copyrightinformationen

1.2 Danksagungen und die Bitte um Hilfe

Ich möchte mich hier vor allem bei Nikolai Langfeldt bedanken, ohne dessen Version dieser Text niemals möglich gewesen wäre.

Dieser Text wird immer unvollständig bleiben. Sollten Probleme oder Fragen beim Durcharbeiten dieses Textes auftreten, schreibt mir bitte. Dadurch helft Ihr mir diese HOWTO zu verbessern. Also schickt Vorschläge, Kommentare, Kritik, Pizza, Geld und/oder Fragen an mich. Sollten Eure Anregungen auch das englische Original betreffen, werde ich Nikolai darüber informieren. Die Übersetzung bringt es mit sich, dass einige Sätze holprig oder unverständlich klingen - wenn etwas überhaupt keinen Sinn macht, schreibt mir das bitte.

Wenn Ihr Antwort von mir wollt, achtet bitte darauf, dass der Absender in Eurer E-Mail korrekt eingetragen ist. Ich bekomme *sehr viele* E-Mails. Bitte lest auch den Abschnitt 8, bevor Ihr mir schreibt. Achso - ich verstehe nur die deutsche und die englische Sprache.

Sollte diese HOWTO in noch andere Sprachen übersetzt werden, würde ich gerne davon erfahren, damit ich eine Übersicht darüber bekomme, in welchen Ländern meine Texte verbreitet werden :-) - außerdem kann ich Euch dann eine Nachricht zukommen lassen, wenn Änderungen an der HOWTO gemacht wurden.

So - jetzt lasse ich Nikolai nochmal zu Wort kommen:

> I want to thank Arnt Gulbrandsen whom I cause to suffer through the drafts to this work and whom provided many useful suggestions. I also want to thank the numerous people that have e-mailed suggestions and notes.

1.3 Widmung

Hier noch eine Widmung von Nikolai - und dann geht's ab ins DNS-Getümmel:

> This HOWTO is dedicated to Anne Line Norheim Langfeldt. Though she will probably never read it since she's not that kind of girl.

2 Einleitung

Worum es geht - und worum nicht.

DNS steht für Domain Name System. Das DNS wandelt einen Rechnernamen in die jeweils zugeordnete IP-Adresse um, die jeder Computer im Netz braucht. Es wandelt Namen in IP-Adressen und umgekehrt und kann sogar noch ein wenig mehr. Diese HOWTO beschreibt, wie man einen DNS-Server auf einem Linuxsystem einrichtet und verwaltet.

DNS ist für den unerfahrenen Benutzer (Dich ;-)) eines der undurchsichtigeren Gebiete der Netzwerkadministration. Diese HOWTO versucht, alles etwas durchschaubarer zu machen. Sie beschreibt, wie ein *einfacher* DNS-Server aufgebaut wird. Zu Anfang wird ein »caching-only« Server eingerichtet, der dann zu einem primären DNS-Server für eine Domain ausgebaut wird. Um kompliziertere DNS-Server einzurichten, sollte man einen Blick in den Abschnitt 8 werfen. Falls das Problem dort nicht beschrieben wird, bleibt einem nichts anderes übrig, als die original Dokumentation zu lesen. Im Abschnitt 9 habe ich aufgelistet, woraus die echte Dokumentation besteht und wo man sie finden kann.

Bevor es nun endgültig losgeht, sollte der Computer so eingerichtet sein, dass man z.B. mit `telnet` Verbindungen zum und vom Rechner aufbauen kann und dass auch andere Netzwerkverbindungen funktionieren. Wichtig ist, dass ein

```
telnet 127.0.0.1
```

eine Verbindung zum eigenen Rechner aufbaut, was man jetzt testen sollte. Als Ausgangspunkt braucht man ausserdem richtig konfigurierte `/etc/nsswitch.conf`- (bzw. `/etc/host.conf`), `/etc/resolv.conf`- und `/etc/hosts`-Dateien; da ich deren Funktion hier nicht erklären werde, empfehle ich die *NET-3 HOWTO* und/oder die *PPP HOWTO*, in denen beschrieben wird, was dort eingestellt werden muss.

Wenn ich von dem »eigenen Rechner« spreche, meine ich den Computer, auf dem versucht wird, den Nameserver einzurichten und keinen anderen, der sich vielleicht im Netzwerk befindet.

Ausserdem gehe ich davon aus, dass der Rechner nicht hinter irgendeinem Firewall hängt, der eventuell Nameserveranfragen verhindert. In diesem Fall braucht man eine spezielle Konfiguration, die im Abschnitt 8 beschrieben wird.

Der Nameservice wird unter UNIX von einem Programm namens `named` bewerkstelligt. Dieses Programm ist Teil des »bind«-Paketes, das von Paul Vixie für das Internet Software Consortium verwaltet wird. `named` ist in den meisten Linux-Distributionen enthalten und wird üblicherweise als `/usr/sbin/named` installiert. Wenn bereits ein `named` vorhanden ist, kann er vermutlich auch genutzt werden; falls nicht, kann man ein Binärpaket von einem Linux-FTP-Server bekommen oder man holt sich den aktuellsten Source-Code von:

```
ftp.isc.org:/isc/bind/src/cur/bind-8/
```

Diese HOWTO beschreibt bind in der Version 8. Die alte Version für bind Version 4 ist nicht ins Deutsche übersetzt worden - die englische Version ist unter

```
http://www.math.uio.no/~janl/DNS/
```

zu finden. Wenn in der Manual Page zum named (`man named`) am Ende im FILES-Abschnitt eine Datei namens `named.conf` aufgelistet wird, ist bind 8 installiert. Wenn es sich um die Datei `named.boot` handelt, ist es bind 4. Wenn nur bind 4 installiert wurde und man sich Gedanken über die Sicherheit seines Computers macht - das sollte jeder -, sollte man auf jeden Fall auf bind 8 updaten.

Das DNS ist eine netzweite Datenbank. Man sollte darauf achten, womit man sie füttert. Wenn man sie mit Müll füllt bekommt man selbst - und andere auch - nur Müll von ihr geliefert. Wer das eigene DNS ordentlich pflegt, wird von ihm auch sinnvolle Daten erhalten. Jeder, der lernt, das DNS zu benutzen, zu verwalten und Fehler zu beheben, darf sich in die Riege der »guten« Systemadministratoren aufgenommen fühlen und hilft dabei, das Netz vor einem Zusammenbruch zu bewahren.

In diesem Dokument werde ich gelegentlich ein paar Dinge behaupten, die nicht unbedingt der Wahrheit entsprechen - obwohl sie mindestens die halbe Wahrheit darstellen -, um es etwas einfacher zu machen. Alles sollte funktionieren, wenn man meinen Ausführungen glaubt.

Von allen bereits existierenden Dateien, die im Laufe dieses Textes geändert werden, sollten Sicherheitskopien erstellt werden, um im Notfall wieder zur alten - funktionierenden - Konfiguration zurückkehren zu können.

3 Ein »caching-only«-Nameserver

Ein erster Ausblick auf die DNS-Konfiguration - sehr praktisch für Benutzer, die sich ins Internet einwählen.

Ein »caching-only« Nameserver (ich werde den englischen Begriff weiterverwenden, weil sich »nur-Zwischenspeicher« nicht besonders schön anhört) wird auf Namensanfragen antworten und sich bei der nächsten Anfrage an die alte Antwort erinnern. Dieses Vorgehen verkürzt vor allem die Wartezeit bei jeder weiteren Anfrage - besonders, wenn man nur eine langsame Verbindung hat.

Zuerst wird eine Datei namens `/etc/named.conf` gebraucht. Sie wird bei jedem Start vom named eingelesen. Erstmal sollte sie einfach nur das folgende enthalten:

```
// Konfigurationsdatei für einen caching-only Nameserver

options {
        directory "/var/named";

        // Wenn Verbindungen über eine Firewall gehen müssen und das nicht
        // so funktioniert, wie es sollte, hilft es vielleicht, die folgende
        // Zeile auszukommentieren.

        // query-source port 53;
};

zone "." {
        type hint;
        file "root.hints";
};

zone "0.0.127.in-addr.arpa" {
        type master;
        file "pz/127.0.0";
};
```

Die »`directory`«-Zeile sagt dem named, wo er nach Dateien suchen soll. Alle noch folgenden Dateien gehören in dieses Verzeichnis (oder einem Unterverzeichnis relativ hierzu). Also ist `pz` ein Unterverzeichnis in `/var/named`: `/var/named/pz`. `/var/named` ist nach dem *Linux File system Standard* das für den Nameserver vorgesehene Verzeichnis für die Konfigurationsdateien eines Nameservers.

Die Datei `/var/named/root.hints` sollte die folgenden Daten beinhalten:

```
;
; Wenn diese Datei bereits existiert, könnten hier
; einführende Kommentare stehen.
; Falls nicht - keine Panik ;-).
;
.                       6D IN NS        G.ROOT-SERVERS.NET.
.                       6D IN NS        J.ROOT-SERVERS.NET.
.                       6D IN NS        K.ROOT-SERVERS.NET.
.                       6D IN NS        L.ROOT-SERVERS.NET.
.                       6D IN NS        M.ROOT-SERVERS.NET.
.                       6D IN NS        A.ROOT-SERVERS.NET.
.                       6D IN NS        H.ROOT-SERVERS.NET.
.                       6D IN NS        B.ROOT-SERVERS.NET.
.                       6D IN NS        C.ROOT-SERVERS.NET.
.                       6D IN NS        D.ROOT-SERVERS.NET.
.                       6D IN NS        E.ROOT-SERVERS.NET.
.                       6D IN NS        I.ROOT-SERVERS.NET.
.                       6D IN NS        F.ROOT-SERVERS.NET.

G.ROOT-SERVERS.NET.     5w6d16h IN A    192.112.36.4
J.ROOT-SERVERS.NET.     5w6d16h IN A    198.41.0.10
K.ROOT-SERVERS.NET.     5w6d16h IN A    193.0.14.129
L.ROOT-SERVERS.NET.     5w6d16h IN A    198.32.64.12
M.ROOT-SERVERS.NET.     5w6d16h IN A    202.12.27.33
A.ROOT-SERVERS.NET.     5w6d16h IN A    198.41.0.4
H.ROOT-SERVERS.NET.     5w6d16h IN A    128.63.2.53
B.ROOT-SERVERS.NET.     5w6d16h IN A    128.9.0.107
C.ROOT-SERVERS.NET.     5w6d16h IN A    192.33.4.12
D.ROOT-SERVERS.NET.     5w6d16h IN A    128.8.10.90
E.ROOT-SERVERS.NET.     5w6d16h IN A    192.203.230.10
I.ROOT-SERVERS.NET.     5w6d16h IN A    192.36.148.17
F.ROOT-SERVERS.NET.     5w6d16h IN A    192.5.5.241
```

Diese Datei enthält die Hauptnameserver des Internet; auch Root-Nameserver genannt. Da sich diese Daten gelegentlich ändern, muss sie regelmässig erneuert werden. Mehr darüber gibt es im Abschnitt 6.

Der nächste zu erklärende Abschnitt in der `named.conf`-Datei ist die letzte `Zone`. Darauf werde ich aber erst später zurückkommen. Erstmal wird jetzt eine Datei namens `127.0.0` im Unterverzeichnis `pz` erstellt:

```
@               IN      SOA     ns.linux.test. hostmaster.linux.test. (
                                1       ; Serial
                                8H      ; Refresh
                                2H      ; Retry
                                1W      ; Expire
                                1D)     ; Minimum TTL
                        NS      ns.linux.test.
1                       PTR     localhost.
```

Der nächste Schritt ist die Erstellung einer `/etc/resolv.conf`-Datei, die so aussehen sollte:

```
search unterdomain.eigene-domain.de eigene-domain.de
nameserver 127.0.0.1
```

Die »search«-Zeile definiert die Suchreihenfolge der Domains, zu denen ein Rechner gehören könnte, der eine Anfrage an den named durchführt. Die »nameserver«-Zeile enthält die Adresse vom eigenen Nameserver. In diesem Fall ist das der eigene Computer, da dass der Rechner sein wird, auf dem der named eingerichtet wird (127.0.0.1 ist richtig, ganz egal, was die Maschine sonst noch für Adressen hat). Wenn man mehrere Nameserver auflisten möchte, macht man das, indem für jeden Server eine eigene »nameserver«-Zeile eingefügt wird. Der named selber wird diese Datei nie einlesen, sondern der resolver. Das ist ein Programm, welches die Anfragen an den named durchführt.

Eine kleine Beschreibung, was diese Datei bewirkt: Wenn ein Client-Programm versucht, einen Computer namens `bla` zu finden, dann wird zuerst nach `bla.unterdomain.eigene-domain.de` gesucht, anschliessend nach `bla.eigene-domain.de` und erst zuletzt nach `bla`. Wenn versucht wird, den Rechner `sunsite.unc.edu` zu finden, wird zuerst nach `sunsite.unc.edu.unterdomain.eigene-domain.de` (ja - das ist nicht sinnvoll, aber so funktioniert es nunmal...), dann nach `sunsite.unc.edu.eigene-domain.de` und erst zuletzt nach `sunsite.unc.edu` gesucht. Aus diesem Grund sollte man nicht zu viele Domains in die search-Zeile schreiben, da es lange dauern kann bis alle durchsucht wurden.

Das Beispiel geht davon aus, dass man zu der Domain `unterdomain.eigene-domain.de` gehört. Der eigene Rechner wird dann vermutlich `eigener-Rechner.unterdomain.eigene-domain.de` heissen. Auf keinen Fall sollte die search-Zeile die eigene TLD (Top Level Domain, in diesem Fall »de«) enthalten. Wenn man regelmässig Verbindungen zu Computern in anderen Domains aufbaut, dann wird diese wie folgt zusätzlich eingetragen:

```
search unterdomain.eigene-domain.de eigene-domain.de andere-domain.com
```

Und so weiter. Natürlich sollten existierende Domainnamen anstelle meiner ausgedachten benutzt werden. Ausserdem sollte man darauf achten, dass am Ende der Domainnamen keine Punkte stehen - wieso das wichtig ist, wird später erklärt.

Der nächste Schritt hängt von der verwendeten libc-Version ab. Entweder muss man die Datei `/etc/nsswitch.conf` oder die Datei `/etc/host.conf` anpassen. Wenn bereits eine `nsswitch.conf` existiert, muss diese geändert werden - falls nicht, passt man die `host.conf` an.

/etc/nsswitch.conf

In dieser etwas längeren Datei wird eingestellt, aus welcher Datei oder Datenbank Daten zu einem bestimmten Dienst geholt werden. Sie enthält normalerweise gleich am Anfang sehr hilfreiche Kommentare, die gelesen werden sollten. Die mit »hosts:« beginnende Zeile sollte wie folgt lauten:

```
hosts:      files dns
```

Wenn es noch keine Zeile mit »hosts:« am Anfang gibt, dann wird die oben gezeigte einfach hinzugefügt. Die Zeile besagt, dass Programme zuerst in der `/etc/hosts`-Datei nachschauen sollen und erst dann das DNS nach den Vorgaben in der `resolv.conf` zur Aufschlüsselung von Rechnernamen genutzt wird.

/etc/host.conf

Diese Datei enthält vermutlich mehrere Zeilen. Eine davon müsste mit `order` beginnen und sollte wie die folgende aussehen:

```
order hosts,bind
```

Wenn es keine »order«-Zeile gibt, wird sie hinzugefügt. Sie bewirkt, dass die Routinen, die die Computernamen aufschlüsseln, zuerst in der `/etc/hosts` nachschauen und dann beim Nameserver anfragen, der laut `resolv.conf` der Rechner mit der IP 127.0.0.1 ist.

3.1 Starten des named

Nachdem das alles erledigt ist, ist es an der Zeit, den named das erste Mal zu starten. Eventuell ist vorher noch die Verbindung ins Internet zu starten. Dann gibt man

```
ndc start/
```

ein. Sollte das nicht funktionieren, klappt statt dessen vielleicht der folgende Befehl:

```
/usr/sbin/ndc start
```

Wenn auch das nicht das gewünschte Ergebnis liefert, bleibt nur ein Blick in den Abschnitt 8. Ein Blick in die Datei, in der der syslog Nachrichten speichert (üblicherweise heisst diese `/var/adm/messages` - sie kann aber auch im Verzeichnis `/var/log` gefunden werden oder nicht `messages` sondern `syslog` heissen...), während der named gestartet wird (`tail -f /var/log/messages`) sollte etwas Ähnliches wie das folgende zeigen:

(Zeilen, die mit \ enden, werden in der nächsten Zeile fortgeführt)

```
Feb 15 01:26:17 roke named[6091]: starting.  named 8.1.1 Sat Feb 14 \
  00:18:20 MET 1998 janl@roke.uio.no:/var/tmp/bind-8.1.1/src/bin/named
Feb 15 01:26:17 roke named[6091]: cache zone "" (IN) loaded (serial 0)
Feb 15 01:26:17 roke named[6091]: master zone "0.0.127.in-addr.arpa" \
  (IN) loaded (serial 1)
Feb 15 01:26:17 roke named[6091]: listening [127.0.0.1].53 (lo)
Feb 15 01:26:17 roke named[6091]: listening [129.240.230.92].53 (ippp0)
Feb 15 01:26:17 roke named[6091]: Forwarding source address is [0.0.0.0].1040
Feb 15 01:26:17 roke named[6092]: Ready to answer queries.
```

Wenn der named Fehler in der Konfiguration findet, meldet er im Logfile unter anderem den Name der Datei, die den Fehler enthält: entweder `named.conf` oder `root.hints` - hoffe ich :-). Dann wird der named beendet und die Datei muss überarbeitet werden.

Ansonsten können die Einstellungen jetzt getestet werden. Mit `nslookup` kann man seine Arbeit kontrollieren.

```
$ nslookup
Default Server:  localhost
Address:  127.0.0.1

>
```

Wenn dieses auf dem Bildschirm erscheint, läuft es. Hoffentlich. Wenn etwas anderes auftaucht, müssen alle Einstellungen von Anfang an kontrolliert werden. Jedesmal, wenn die `named.conf`-Datei verändert wurde, muss der named mit dem Befehl

```
named restart
```

neu gestartet werden.

Ansonsten kann jetzt eine erste Anfrage gemacht werden. Zuerst versucht man einen Rechner herauszufinden, der sich »in der Nähe« befindet. `pat.uio.no` befindet sich in meiner Nähe - an der Universität in Oslo:

```
> pat.uio.no
Server:  localhost
Address:  127.0.0.1

Name:    pat.uio.no
Address:  129.240.130.16
```

`nslookup` hat jetzt den gerade konfigurierten named nach dem Rechner `pat.uio.no` gefragt. Um darauf antworten zu können, fragt dieser einen der Nameserver aus der `root.hints`-Datei und bahnt sich von dort aus seinen Weg zum vollen Namen des Rechners. Vielleicht dauert es ein wenig, bis das Ergebnis gezeigt wird, weil zuerst alle Domains aus der `/etc/resolv.conf` durchsucht werden.

Macht man die gleiche Anfrage nochmal, bekommt man dieses Ergebnis:

```
> pat.uio.no
Server:  localhost
Address:  127.0.0.1

Non-authoritative answer:
Name:    pat.uio.no
Address:  129.240.2.50
```

Man beachte die »`Non-authoritative answer:`«-Zeile, die dieses mal zusätzlich erscheint. Sie bedeutet, dass der named dieses Mal nicht im Netz nachgefragt hat, sondern dass die Informationen aus seinem Cache stammen. Durch die »`Non-authorative answer:`« wird man über die (sehr kleine) Möglichkeit informiert, dass diese zwischengespeicherten Daten *eventuell* nicht mehr aktuell sind. Wenn `nslookup` diese Zeile bei der zweiten Anfrage ausgibt, ist das ein sicheres Zeichen dafür, dass die Daten zwischengespeichert werden und dass der named wie gewünscht funktioniert. `nslookup` wird beendet, indem man »`exit`« eingibt.

3.2 Was noch verbessert werden kann

In grossen, gut organisierten wissenschaftlichen oder ISP (Internet Service Provider)-Netzen sieht man gelegentlich, dass die Netzwerkadministratoren sogenannte »forwarder«-DNS-Server eingerichtet haben, die dazu beitragen, dass die interne und externe Netzwerkbelastung klein bleibt. Es ist nicht leicht, herauszufinden, ob man sich in so einem Netz befindet - oder nicht. Es ist aber auch nicht wichtig und indem man den DNS-Server vom eigenen Provider als »forwarder« benutzt, kann man die Antwortzeiten auf Anfragen erheblich verkürzen und die Netzwerkbelastung sehr niedrig halten. Gerade für Leute mit Modem ist das ein nicht unerheblicher Vorteil. Um ein Beispiel nennen zu können, gehe ich davon aus, dass der Netz-Provider zwei Nameserver hat, die man benutzen kann und die die IPs `10.0.0.1` und `10.1.0.1` haben. Dann werden in die Datei `named.conf` in den Anfangsabschnitt »options« die folgenden Zeilen eingefügt:

```
forward first;
forwarders {
     10.0.0.1;
     10.1.0.1;
 };
```

Jetzt den Nameserver neu starten und mit `nslookup` testen. Er sollte die gleichen Ergebnisse wie vorher liefern.

3.3 Gratulation

Damit ist die Installation eines »caching-only« Nameservers beendet. Zeit für ein Bier, eine Milch oder was auch immer man zur Feier des Tages trinken möchte.

4 Eine »einfache« Domain

Dieser Abschnitt beschreibt, wie man seine eigene Domain einrichtet.

4.1 Aber zuerst etwas trockene Theorie

Bevor es *wirklich* mit diesem Abschnitt losgehen kann, soll anhand eines Beispiels ein wenig auf die theoretische Funktionsweise des DNS eingegangen werden. Diese wichtigen und interessanten Informationen sollten auf jeden Fall gelesen werden. Wer nicht alles lesen möchte, kann es wenigstens einmal überfliegen - bis zu der Beschreibung des Inhalts der `named.conf`-Datei.

Das DNS ist ein hierarchisches System, in der Form eines Baumes. Die Spitze wird als »`.`« geschrieben und wird »root« (Wurzel) genannt. Unter dem »`.`« existieren einige Top Level Domains (TLDs) - die bekanntesten sind `ORG`, `COM`, `EDU`, `NET` und natürlich `DE` - es gibt aber noch viele mehr. Wie bei einem Baum gibt es neben der Wurzel auch Äste. Jeder, der sich etwas in den Computerwissenschaften auskennt, erkennt im DNS einen Suchbaum, an dem sich Knoten, Verzweigungen und mehr befinden.

Wenn nach einem Computernamen gesucht wird, taucht die Abfrage an der Spitze rekursiv nach unten wandernd in die Hierarchie ein. Wenn man die Adresse von `prep.ai.mit.edu` herausfinden will, muss der Nameserver erst einmal herausfinden, welcher Nameserver für die Domain `edu` zuständig ist. Dazu befragt er einen `.`-Server, den er dank der `root.hints`-Datei kennt. Der root-Server listet dann die Server für `edu` auf:

```
$ nslookup
Default Server:  localhost
Address:  127.0.0.1
```

Am Anfang wird ein root-Server befragt:

```
> server c.root-servers.net.
Default Server:  c.root-servers.net
Address:  192.33.4.12
```

Hierzu wird die Abfrageart auf NS gesetzt (Nameserver-Einträge):

```
> set q=ns
```

Nach `edu` fragen:

```
> edu.
```

Der abschliessende ».« dieser Anfrage ist sehr wichtig. `nslookup` weiß dadurch, dass `edu` direkt unter der Wurzel zu finden ist (und nicht unter irgendeiner `search`-Domain. So wird die Abfrage beschleunigt).

```
edu     nameserver = A.ROOT-SERVERS.NET
edu     nameserver = H.ROOT-SERVERS.NET
edu     nameserver = B.ROOT-SERVERS.NET
edu     nameserver = C.ROOT-SERVERS.NET
edu     nameserver = D.ROOT-SERVERS.NET
edu     nameserver = E.ROOT-SERVERS.NET
edu     nameserver = I.ROOT-SERVERS.NET
edu     nameserver = F.ROOT-SERVERS.NET
edu     nameserver = G.ROOT-SERVERS.NET
A.ROOT-SERVERS.NET      internet address = 198.41.0.4
H.ROOT-SERVERS.NET      internet address = 128.63.2.53
B.ROOT-SERVERS.NET      internet address = 128.9.0.107
C.ROOT-SERVERS.NET      internet address = 192.33.4.12
D.ROOT-SERVERS.NET      internet address = 128.8.10.90
E.ROOT-SERVERS.NET      internet address = 192.203.230.10
I.ROOT-SERVERS.NET      internet address = 192.36.148.17
F.ROOT-SERVERS.NET      internet address = 192.5.5.241
G.ROOT-SERVERS.NET      internet address = 192.112.36.4
```

Diese Auflistung bedeutet, dass alle `ROOT-SERVERS.NET`-Server für `EDU.` zuständig sind, also kann einer von denen befragt werden. Wir fragen bereits den `C`-Root-Nameserver. Dann kann die nächste Ebene des Domainnamens herausgefunden werden: `mit.edu`:

```
> mit.edu.
Server:  c.root-servers.net
Address:  192.33.4.12

Non-authoritative answer:
mit.edu nameserver = W20NS.mit.edu
mit.edu nameserver = BITSY.mit.edu
mit.edu nameserver = STRAWB.mit.edu

Authoritative answers can be found from:
W20NS.mit.edu   internet address = 18.70.0.160
BITSY.mit.edu   internet address = 18.72.0.3
STRAWB.mit.edu  internet address = 18.71.0.151
```

`strawb`, `w20ns` und `bitsy` sind alle für `mit.edu` zuständig. Einer davon wird ausgewählt und es geht eine Ebene tiefer: `ai.mit.edu`:

```
> server W20NS.mit.edu.
```

Die Gross-/Kleinschrift ist bei Rechnernamen unwichtig, aber da ich die Daten per Maus kopiere und hier einfüge, sieht die Ausgabe genau wie auf dem Bildschirm aus.

```
Server:  W20NS.mit.edu
Address:  18.70.0.160

> ai.mit.edu.
Server:  W20NS.mit.edu
Address:  18.70.0.160

Non-authoritative answer:
ai.mit.edu      nameserver = ALPHA-BITS.AI.MIT.EDU
```

```
ai.mit.edu      nameserver = GRAPE-NUTS.AI.MIT.EDU
ai.mit.edu      nameserver = TRIX.AI.MIT.EDU
ai.mit.edu      nameserver = MUESLI.AI.MIT.EDU
ai.mit.edu      nameserver = LIFE.AI.MIT.EDU
ai.mit.edu      nameserver = BEET-CHEX.AI.MIT.EDU
ai.mit.edu      nameserver = MINI-WHEATS.AI.MIT.EDU
ai.mit.edu      nameserver = COUNT-CHOCULA.AI.MIT.EDU
ai.mit.edu      nameserver = MINTAKA.LCS.MIT.EDU

Authoritative answers can be found from:
AI.MIT.EDU      nameserver = ALPHA-BITS.AI.MIT.EDU
AI.MIT.EDU      nameserver = GRAPE-NUTS.AI.MIT.EDU
AI.MIT.EDU      nameserver = TRIX.AI.MIT.EDU
AI.MIT.EDU      nameserver = MUESLI.AI.MIT.EDU
AI.MIT.EDU      nameserver = LIFE.AI.MIT.EDU
AI.MIT.EDU      nameserver = BEET-CHEX.AI.MIT.EDU
AI.MIT.EDU      nameserver = MINI-WHEATS.AI.MIT.EDU
AI.MIT.EDU      nameserver = COUNT-CHOCULA.AI.MIT.EDU
AI.MIT.EDU      nameserver = MINTAKA.LCS.MIT.EDU
ALPHA-BITS.AI.MIT.EDU   internet address = 128.52.32.5
GRAPE-NUTS.AI.MIT.EDU   internet address = 128.52.36.4
TRIX.AI.MIT.EDU internet address = 128.52.37.6
MUESLI.AI.MIT.EDU       internet address = 128.52.39.7
LIFE.AI.MIT.EDU internet address = 128.52.32.80
BEET-CHEX.AI.MIT.EDU    internet address = 128.52.32.22
MINI-WHEATS.AI.MIT.EDU  internet address = 128.52.54.11
COUNT-CHOCULA.AI.MIT.EDU        internet address = 128.52.38.22
MINTAKA.LCS.MIT.EDU     internet address = 18.26.0.36
```

Also ist `museli.ai.mit.edu` ein Nameserver für `ai.mit.edu`:

```
> server MUESLI.AI.MIT.EDU
Default Server:  MUESLI.AI.MIT.EDU
Address:  128.52.39.7
```

Jetzt wird der Typ der Anfrage gewechselt. Der Nameserver wurde gefunden, also wird `muesli` gefragt, was er alles über `prep.ai.mit.edu` weiss.

```
> set q=any
> prep.ai.mit.edu.
Server:  MUESLI.AI.MIT.EDU
Address:  128.52.39.7

prep.ai.mit.edu CPU = dec/decstation-5000.25    OS = unix
prep.ai.mit.edu
        inet address = 18.159.0.42, protocol = tcp
          ftp  telnet  smtp  finger
prep.ai.mit.edu preference = 1, mail exchanger = gnu-life.ai.mit.edu
prep.ai.mit.edu internet address = 18.159.0.42
ai.mit.edu      nameserver = beet-chex.ai.mit.edu
ai.mit.edu      nameserver = alpha-bits.ai.mit.edu
ai.mit.edu      nameserver = mini-wheats.ai.mit.edu
ai.mit.edu      nameserver = trix.ai.mit.edu
ai.mit.edu      nameserver = muesli.ai.mit.edu
```

```
ai.mit.edu      nameserver = count-chocula.ai.mit.edu
ai.mit.edu      nameserver = mintaka.lcs.mit.edu
ai.mit.edu      nameserver = life.ai.mit.edu
gnu-life.ai.mit.edu     internet address = 128.52.32.60
beet-chex.ai.mit.edu    internet address = 128.52.32.22
alpha-bits.ai.mit.edu   internet address = 128.52.32.5
mini-wheats.ai.mit.edu  internet address = 128.52.54.11
trix.ai.mit.edu internet address = 128.52.37.6
muesli.ai.mit.edu       internet address = 128.52.39.7
count-chocula.ai.mit.edu        internet address = 128.52.38.22
mintaka.lcs.mit.edu     internet address = 18.26.0.36
life.ai.mit.edu internet address = 128.52.32.80
```

Angefangen bei ».« wurden so der Reihe nach die Nameserver für jede einzelne Ebene des Domainnamens gefunden. Wenn hierzu anstelle der vielen einzelnen Nameserver nur der eigene Nameserver benutzt worden wäre, hätte dieser natürlich alle Informationen zwischengespeichert, die auf der Suche nach dem Rechnernamen aufgetaucht wären; so bräuchte er für eine Weile nicht mehr im Netz nachfragen.

Analog zu der Baumstruktur ist jeder ».« im Rechnername eine Verzweigung. Jeder Teil zwischen zwei Punkten ist der Name eines Astes im Baum.

Man »klettert« auf den Baum, indem man den Rechnernamen nimmt (`prep.ai.mit.edu`) und bei der Wurzel (`.`) beginnend die Zweige hinabsteigt. Der nächste Zweig in diesem Beispiel wäre `edu`. Die nächste Ebene erreicht man durch das Umschalten zu dem Nameserver, der diesen Teil des Namens kennt. Als nächstes folgt der `mit`-Zweig, der an den `edu`-Zweig anknüpft (zusammen ergibt das `mit.edu`) und besteigt diesen Zweig wiederum durch Umschalten zu dem Server der die Informationen für `mit.edu` bereitstellt. Der nächste Zweig lautet `ai.mit.edu` und erneut wird der Server gewechselt. Jetzt wurde der richtige Server an der richtigen Abzweigung erreicht. Die letzte Aufgabe ist die Abfrage der Informationen zu `prep.ai.mit.edu` - ziemlich einfach, oder?

Eine Domain, über die wenig gesprochen wird, die aber mindestens genauso wichtig wie alle anderen ist, ist `in-addr.arpa`. Sie ist wie die »normalen« Domains aufgebaut und erlaubt es uns, den Rechnernamen herauszufinden, wenn nur die IP-Adresse bekannt ist. Eine wichtige Eigenheit muss man allerdings kennen: Die IP-Adressen werden in der `in-addr.arpa`-Domain verkehrt herum aufgelistet. Wenn man die IP-Adresse `192.128.52.43` hat, arbeitete der named genauso wie im `prep.ai.mit.edu`-Beispiel: Er sucht nach den `arpa.`-Servern und dann nach `in-addr.arpa.`-Servern, `192.in-addr.arpa.`-Servern, `128.192.in-addr.arpa.`-Servern und nach `52.128.192.in-addr.arpa.` Servern. Letztendlich findet er dann die Einträge für `43.52.128.192.in-addr.arpa.` Clever, oder? Die Antwort sollte »ja« lauten ;-). Trotzdem wird diese Umkehrung der IP-Nummern jahrelang für Verwirrung sorgen.

Ich habe gerade ein wenig geflunkert. Das DNS arbeitet nicht so wie ich es gerade erklärt habe. Aber die Beschreibung ist nahe genug an der Wahrheit.

4.2 Die eigene Domain

Um eine eigene Domain zu erstellen, wird in dieser HOWTO als Beispiel der Domainname `linux.test` benutzt. Einzelne Rechner werden zu dieser Domain hinzugefügt. Es wird eine »test«-TLD benutzt, um sicherzustellen, dass niemand »da draussen« durch diese Domain gestört wird.

Eins noch, bevor es losgeht: Es sind nicht alle Zeichen in einem Rechnernamen erlaubt. Diese Einschränkung beinhaltet, dass nur die Buchstaben des englischen Alphabets benutzt werden dürfen: a-z, Ziffern (0-9) und der Bindestrich (»-«). Nochmal, es sollten und dürfen keine anderen Zeichen benutzt werden. Im DNS gibt es keinen Unterschied zwischen Groos- und Kleinbuchstaben - `pat.uio.no` is identisch mit `Pat.UiO.No`.

Die erste wichtige Einstellung wurde bereits durch die folgende Zeile in der `named.conf` gemacht:

```
zone "0.0.127.in-addr.arpa" {
        type master;
        file "pz/127.0.0";
};
```

Zu beachten ist, dass der »`.`« am Ende der Domainnamen in dieser Datei fehlt. Die Einstellungen definieren die Zone `0.0.127.in-addr.arpa`, dass unser Server der Haupt-(Master)Server dafür ist und dass die Daten in einer Datei mit dem Namen `pz/127.0.0` gespeichert werden. Auch diese Datei existiert bereits:

```
@       IN    SOA     ns.linux.test. hostmaster.linux.test. (
                      1       ; Serial
                      8H      ; Refresh
                      2H      ; Retry
                      1W      ; Expire
                      1D)     ; Minimum TTL
                      NS      ns.linux.test.
1                     PTR     localhost.
```

Wichtig ist der »`.`« am Ende von jedem kompletten Domainnamen in dieser Datei - im Gegensatz zu der `named.conf`-Datei von oben. Es gibt Leute, die jede Zonendatei mit einem `$ORIGIN`-Statement beginnen, aber das ist überflüssig. Der Ursprung (origin) einer Zonendatei (der Punkt an dem sich die Domain in der DNS-Hierarchie befindet) wird durch den Zonen-Abschnitt in der `named.conf`-Datei definiert. In diesem Fall ist das `0.0.127.in-addr.arpa`.

Die gerade erstellte Zonen-Datei enthält 3 Eintragungen (»resource records« (RRs)). Einen NS RR und einen PTR RR. SOA ist die Abürzung für »Start Of Authority«. Der Klammeraffe »@« ist ein spezielles Zeichen und steht für den Ursprung dieser Domain. Die »domain«-Zeile für diese Datei definiert den Ursprung als 0.0.127.in-addr.arpa, also steht in der ersten Zeile ausgeschrieben:

```
0.0.127.in-addr.arpa.   IN      SOA ...
```

NS ist der Name-Server RR. Am Anfang der Zeile fehlt das »@«. Man kann sich diese »Tipparbeit« sparen, weil bereits die vorhergehende Zeile mit »@« anfing. Die NS-Zeile könnte also auch so geschrieben werden:

```
0.0.127.in-addr.arpa.   IN      NS      ns.linux.test
```

Hierdurch erfärt das DNS den Rechner, der als Nameserver für die Domain `0.0.127.in-addr.arpa` fungiert: `ns.linux.test`. »ns« ist der übliche Name für einen Nameserver - genauso wie Web-Server normalerweise `www.`*irgendetwas* heissen. Trotzdem kann natürlich auch jeder beliebige andere Name benutzt werden.

Als letzter Eintrag bedeutet die PTR-Zeile, dass der Rechner mit der Adresse 1 im Subnetz `0.0.127.in-addr.arpa` (also 127.0.0.1) `localhost` heisst.

Der SOA-Eintrag steht am Anfang von *jeder* Zonendatei. Ihn sollte es in jeder Datei nur ein einziges mal geben. Dieser Eintrag beschreibt die Ursprungszone (`linux.test`), wer für den Inhalt der Domaindaten dieser Zone verantwortlich ist (`hostmaster@linux.test`) - man sollte hier seine eigene E-Mail-Adresse einfügen oder besser den üblichen Alias »hostmaster« erzeugen. Ausserdem ist die Versionsnummer dieser Zonendatei enthalten (Seriennummer: 1) und diverse weitere Einstellungen, die mit dem Caching und Secondary (Zweit-)Nameservern zu tun haben. Wenn die Einstellungen der Felder (refresh, retry, expire und minimum) aus der HOWTO übernommen werden, sollte man auf der sicheren Seite sein.

Jetzt wird der named neu gestartet (der Befehl hierfür ist `ndc restart`) und die Einstellungen werden mit `nslookup` kontrolliert:

```
$ nslookup

Default Server:  localhost
Address:  127.0.0.1

> 127.0.0.1
Server:  localhost
Address:  127.0.0.1

Name:    localhost
Address:  127.0.0.1
```

Also ist `nslookup` in der Lage, den Namen `localhost` mit der IP-Adresse 127.0.0.1 herauszufinden - hervorragend.

Um jetzt zurück zur Hauptsache - der `linux.test`-Domain zu kommen, wird ein neuer »Zonen«-Abschnitt in die `named.conf` eingefügt:

```
zone "linux.test" {
        notify no;
        type master;
        file "pz/linux.test";
};
```

Am Ende des Domainnamens in der `named.conf`-Datei wird kein »`.`« geschrieben.

In die `linux.test`-Zonendatei werden jetzt irgendwelche Testdaten eingefügt:

```
;
; Zonendatei für linux.test
;
; Die komplette Zonendatei
;
@       IN      SOA     ns.linux.test. hostmaster.linux.test. (
                        199802151       ; Datum + Seriennummer #
                        8H              ; refresh, Sekunden
                        2H              ; retry, Sekunden
                        1W              ; expire, Sekunden
                        1D )            ; minimum,
;
                NS      ns              ; Rechnername des Nameserver
                MX      10 mail.linux.test      ; erster Mailserver
                MX      20 mail.friend.test.    ; zweiter Mailserver
;
localhost       A       127.0.0.1
ns              A       192.168.196.2
mail            A       192.168.196.4
```

Zwei Anmerkungen müssen über den SOA-Eintrag gemacht werden. `ns.linux.test` *muss* ein Rechner mit einem gültigen A-Record sein. Es ist nicht erlaubt, einen CNAME-Eintrag im SOA-Abschnitt zu benutzen. Der Name muss übrigens nicht »ns« lauten, es kann auch jeder andere gültige Rechnername sein.

Ausserdem liest sich hostmaster.linux.test als hostmaster@linux.test - dies sollte ein Alias oder eine Mailbox sein, die von den DNS-Administratoren regelmässig gelesen wird. Alle E-Mails, die sich auf das DNS beziehen, werden an diese Adresse geschickt. Genauso wie der Rechnername nicht unbedingt `ns` sein muss, kann auch hier eine andere gültige E-Mail-Adresse benutzt werden, aber oft wird davon ausgegangen, dass auch die »hostmaster«-Adresse existiert und gelesen wird.

Es gibt in dieser Datei noch einen neuen RR-Typen - den MX oder Mail eXchanger. Dieser Eintrag liefert Mailservern die Adresse zurück, an die E-Mails geliefert werden sollen, die nach `jemandem@linux.test` geschickt werden. In diesem Fall nach `mail.linux.test` oder `mail.friend.test`. Die Nummer vor den Mail-Einträgen ist die Priorität, mit der die Mailserver ausgewählt werden. Der RR mit der kleinsten Nummer (10) ist der erste, an den versucht wird, E-Mails zu schicken. Wenn das nicht klappt, kann die E-Mail an einen Server mit einer höheren Prioritätsnummer geschickt werden - zum Beispiel einen Not-Mailserver `mail.friend.test`, der hier die Priorität 20 hat.

Der named wird mit Hilfe von `ndc restart` neu gestartet und die Ergebnisse werden wie immer mit `nslookup` begutachtet:

```
$ nslookup
> set q=any
> linux.test
Server:  localhost
Address:  127.0.0.1

linux.test
        origin = ns.linux.test
        mail addr = hostmaster.linux.test
        serial = 199802151
        refresh = 28800 (8 hours)
        retry   = 7200 (2 hours)
        expire  = 604800 (7 days)
        minimum ttl = 86400 (1 day)
linux.test     nameserver = ns.linux.test
linux.test     preference = 10, mail exchanger = mail.linux.test.linux.test
linux.test     preference = 20, mail exchanger = mail.friend.test
linux.test     nameserver = ns.linux.test
ns.linux.test  internet address = 192.168.196.2
mail.linux.test        internet address = 192.168.196.4
```

Bei genauerer Betrachtung sollte man einen Fehler entdecken. Die Zeile

```
linux.test     preference = 10, mail exchanger = mail.linux.test.linux.test
```

ist falsch. Sie sollte

```
linux.test     preference = 10, mail exchanger = mail.linux.test
```

lauten.

Ich habe diesen Fehler mit Absicht eingebaut, um davon zu lernen :-). In der Zonendatei findet man den Fehler in der Zeile

```
                MX      10 mail.linux.test      ; Erster Mailserver
```

Es fehlt ein Punkt oder das »linux.test« ist zuviel. Wenn ein Rechnername in der Zonendatei nicht mit einem Punkt endet, wird die Ursprungszone an den Namen angehängt. Dadurch entsteht die Verdoppelung `linux.test.linux.test`. Also ist die richtige Schreibweise entweder

```
                MX      10 mail.linux.test.     ; Erster Mailserver
```

oder

```
                MX      10 mail                 ; Erster Mailserver
```

Ich benutze die zweite Variante - es ist weniger Tipparbeit. Es gibt einige Bind-Experten, denen diese Arbeitsweise nicht gefällt und andere, die es ebenso machen. Nochmal: In einer Zonendatei sollte die Domain entweder ausgeschrieben und mit einem Punkt beendet werden oder sie sollte komplett weggelassen werden - in diesem Fall wird der Ursprung angehängt.

Ich muss noch einmal betonen, dass in der `named.conf`-Datei *kein* »`.`« hinter dem Domainnamen sein darf. Man glaubt garnicht, wie oft ein Punkt zuviel oder zuwenig Fehler verursacht und die Leute an den Rand des Wahnsinns bringt.

Und jetzt folgt eine neue Zonendatei mit einigen zusätzlichen Einträgen:

```
;
; Zonendatei fürlinux.test
;
; die komplette Zonendatei
;
@       IN      SOA     ns.linux.test. hostmaster.linux.test. (
                        199802151       ; aktuelles Datum + Seriennummer
                        8H              ; refresh, Sekunden
                        2H              ; retry, Sekunden
                        1W              ; expire, Sekunden
                        1D )            ; minimum, Sekunden
;
                TXT     "Linux.Test, die DNS-Einführung"
                NS      ns              ; Rechnername des Nameservers
                NS      ns.friend.test.
                MX      10 mail         ; erster Mailserver
                MX      20 mail.friend.test. ; zweiter Mailserver

localhost       A       127.0.0.1

gw              A       192.168.196.1
                HINFO   "Cisco" "IOS"
                TXT     "Der Router"

ns              A       192.168.196.2
                MX      10 mail
                MX      20 mail.friend.test.
                HINFO   "Pentium" "Linux 2.0"
www             CNAME   ns

donald          A       192.168.196.3
                MX      10 mail
                MX      20 mail.friend.test.
                HINFO   "i486"  "Linux 2.0"
                TXT     "DEK"

mail            A       192.168.196.4
                MX      10 mail
```

```
                MX      20 mail.friend.test.
                HINFO   "386sx" "Linux 1.2"

ftp             A       192.168.196.5
                MX      10 mail
                MX      20 mail.friend.test.
                HINFO   "P6" "Linux 2.1.86"
```

Es gib hier wieder einige neue RRs: HINFO (Host INFOrmation - Rechnerinformation) besteht aus zwei Teilen - es gehört zum guten Ton, beide Teile anzugeben. Der erste Teil ist die Hardare bzw. die CPU des Systems, der zweite Teil enthält den Namen der Software oder des Betriebssystems des Rechners. Der Rechner mit dem Namen »ns« enthält eine Pentium CPU und läuft mit Linux 2.0. CNAME (Canonical Name) ist eine Möglichkeit, einem Rechner mehrere Namen zuzuweisen. Also ist www ein Alias für ns.

Die Benutzung von CNAME-Einträgen ist ein wenig kontrovers. Unter Berücksichtigung der folgenden Regeln, sollte es keine Probleme geben. MX, CNAME oder SOA-Einträge dürfen *niemals* auf einen CNAME-Eintrag verweisen, sondern nur auf A-Records. Aus diesem Grund ist das folgende nicht erlaubt:

```
blabar          CNAME   www                     ; NEIN!
```

Die richtige Variante lautet:

```
blabar          CNAME   ns                      ; JA!
```

Man kann davon ausgehen, dass ein CNAME kein gültiger Rechnername für eine E-Mail-Adresse ist: `webmaster@www.linux.test` ist nach den obigen Einstellungen eine ungültige Adresse. Auch wenn ein Test auf dem eigenen Rechner keine Schwierigkeiten macht, kann man mit Sicherheit erwarten, dass es immer ein paar Mail-Admins gibt, die diese Regel durchboxen. Ein Weg zur Vermeidung dieses Problems ist es, nur A-Einträge zu benutzen (und natürlich andere wie zum Beispiel MX,...):

```
www             A       192.168.196.2
```

Eine Handvoll der alten Bind-Magier empfehlen ebenfalls, *niemals* CNAME zu benutzen. Die Grundsatzdiskussion »warum oder warum nicht« gehört aber nicht in diese HOWTO.

Was man aber auch sehen kann, ist dass diese HOWTO und viele andere Sites diese Regeln nicht befolgen.

Die neuen Einstellungen der Bind-Datenbank werden mit `ndc reload` geladen - der named liest daraufhin seine Konfigurationsdateien neu ein.

```
$ nslookup
Default Server:  localhost
Address:  127.0.0.1

> ls -d linux.test
```

Das bewirkt eine Auflistung aller Einträge. Als Ergebnis erhält man:

```
[localhost]
$ORIGIN linux.test.
@                       1D IN SOA       ns hostmaster (
                                        199802151       ; Seriennummer
                                        8H              ; refresh
```

```
                                        2H              ; retry
                                        1W              ; expiry
                                        1D )            ; minimum

                        1D IN NS        ns
                        1D IN NS        ns.friend.test.
                        1D IN TXT       "Linux.Test, die DNS-Einfuehrung"
                        1D IN MX        10 mail
                        1D IN MX        20 mail.friend.test.
gw                      1D IN A         192.168.196.1
                        1D IN HINFO     "Cisco" "IOS"
                        1D IN TXT       "The router"
mail                    1D IN A         192.168.196.4
                        1D IN MX        10 mail
                        1D IN MX        20 mail.friend.test.
                        1D IN HINFO     "386sx" "Linux 1.0.9"
localhost               1D IN A         127.0.0.1
www                     1D IN CNAME     ns
donald                  1D IN A         192.168.196.3
                        1D IN MX        10 mail
                        1D IN MX        20 mail.friend.test.
                        1D IN HINFO     "i486" "Linux 1.2"
                        1D IN TXT       "DEK"
ftp                     1D IN A         192.168.196.5
                        1D IN MX        10 mail
                        1D IN MX        20 mail.friend.test.
                        1D IN HINFO     "P6" "Linux 1.3.59"
ns                      1D IN A         192.168.196.2
                        1D IN MX        10 mail
                        1D IN MX        20 mail.friend.test.
                        1D IN HINFO     "Pentium" "Linux 1.2"
```

Sehr schön - man erkennt sofort die Ähnlichkeit zu der Zonendatei selbst. Mal sehen, was nur für `www` ausgegeben wird:

```
> set q=any
> www.linux.test.
Server:  localhost
Address:  127.0.0.1

www.linux.test canonical name = ns.linux.test
linux.test     nameserver = ns.linux.test
linux.test     nameserver = ns.friend.test
ns.linux.test  internet address = 192.168.196.2
```

In anderen Worten: Der richtige Name von `www.linux.test` ist `ns.linux.test`. Außerdem werden zusätzliche Informationen über ns ausgegeben - genug um eine Verbindung aufzubauen, wenn man ein Programm wäre.

4.3 Die umgekehrte Zone (Reverse Zone)

Jetzt können Programme also die Rechnernamen der Domain linux.test in die für sie nötigen IP-Adressen umwandeln. Zusätzlich wird aber noch eine Zone benötigt, die die Umwandlung von IP-Adressen in

Rechnernamen ermögicht - eine umgekehrte oder »reverse« Zone. Die Rechnernamen werden von vielen Programmen (z.B. FTP, IRC, WWW und anderen) zur Entscheidung darüber herangezogen, ob sie mit einem reden wollen oder nicht - und falls sie es möchten, welche Priorität man erhält. Um vollen Zugang zu allen Internetdiensten zu bekommen ist eine umgekehrte Zone nötig.

Hierzu wird noch etwas in die `named.conf` eingefügt:

```
zone "196.168.192.in-addr.arpa" {
        notify no;
        type master;
        file "pz/192.168.196";
};
```

Dieser Eintrag ist dem `0.0.127.in-addr.arpa`-Eintrag sehr ähnlich - auch der Inhalt der Zonendatei ist gleich:

```
@       IN      SOA     ns.linux.test. hostmaster.linux.test. (
                        199802151 ; Seriennummer, Datum + Seriennnummer
                        8H      ; Refresh
                        2H      ; Retry
                        1W      ; Expire
                        1D)     ; Minimum TTL
                NS      ns.linux.test.

1               PTR     gw.linux.test.
2               PTR     ns.linux.test.
3               PTR     donald.linux.test.
4               PTR     mail.linux.test.
5               PTR     ftp.linux.test.
```

Jetzt wird der named neu gestartet (`ndc restart`) und die gerade gemachten Einstellungen werden mit `nslookup` überprüft:

```
> 192.168.196.4
Server:  localhost
Address:  127.0.0.1

Name:    mail.linux.test
Address:  192.168.196.4
```

So, das sieht OK aus. Jetzt werden nochmal alle Einstellungen ausgegeben und gecheckt:

```
> ls -d 196.168.192.in-addr.arpa
[localhost]
$ORIGIN 196.168.192.in-addr.arpa.
@                       1D IN SOA       ns.linux.test. hostmaster.linux.test. (
                                        199802151       ; serial
                                        8H              ; refresh
                                        2H              ; retry
                                        1W              ; expiry
                                        1D )            ; minimum

                        1D IN NS        ns.linux.test.
1                       1D IN PTR       gw.linux.test.
```

```
2                       1D IN PTR       ns.linux.test.
3                       1D IN PTR       donald.linux.test.
4                       1D IN PTR       mail.linux.test.
5                       1D IN PTR       ftp.linux.test.
@                       1D IN SOA       ns.linux.test. hostmaster.linux.test. (
                                        199802151       ; serial
                                        8H              ; refresh
                                        2H              ; retry
                                        1W              ; expiry
                                        1D )            ; minimum
```

Das sieht sehr gut aus. Wenn die Ausgabe allerdings nicht mit der obigen übereinstimmt, muss man in der syslog-Datei nach Fehlermeldungen suchen. Wie das gemacht wird, habe ich ganz am Anfang dieses Abschnitts erklärt.

4.4 Warnungen

Es gibt noch ein paar Dinge, die gesagt werden sollten. Die IP-Adressen, die in den obigen Beispielen benutzt wurden, stammen aus einem der »privaten Netze«. Sie dürfen nicht öffentlich im Internet benutzt werden. Darum sind sie ideal für die Benutzung in einer HOWTO. Der zweite Punkt ist die `notify no;`-Zeile. Sie veranlaßt den named dazu, seine Zweit-(Secondary)-Server nicht über Änderungen in den Zonendateien zu informieren. Denn der Bind Version 8 kann die anderen Server, die in den NS-Einträgen einer Zonendatei stehen, über Neuerungen in einer Zone informieren. Das ist für den täglichen Gebrauch sehr praktisch, aber für private Zonen-Experimente sollte diese Option abgeschaltet werden. Wir wollen doch nicht das Internet mit unserem Experiment belästigen, oder?

Ausserdem ist die Domain natürlich nur ein Test - genauso wie alle Adressen, die sie beinhaltet. Ein reales Beispiel aus einer existierenden Domain ist im nächsten Kapitel zu sehen.

4.5 Warum Reverse Lookups nicht funktionieren.

Es gibt ein paar Probleme mit Namensauflösungen, wenn die Adressen rückwärts aufgeschlüsselt werden sollen, die zwar selten aber dennoch auftreten. Bevor ich darauf eingehe, sollte überprüft werden, ob die Reverse Lookups auf dem eigenen Rechner funktionieren. Wenn nicht - zurück an den Anfang und den Fehler beheben.

Ich werde hier im Detail auf zwei Probleme eingehen, wie sie auftreten können, wenn von ausserhalb des eigenen Netzes auf den Nameserver zugegriffen wird.

4.5.1 Die reverse-Zone wurde nicht delegiert.

Wenn man von einen Serviceprovider einen Netzwerkbereich und einen Domainnamen bekommt, wird der Domainname normalerweise automatisch delegiert. Eine Delegierung ist der NS-Eintrag, der einen von einem Nameserver zum nächsten weiterleitet, wie es in dem oberen Theorie-Abschnitt erklärt wurde. Das wurde doch gelesen und verstanden? Wenn die reversen-Zonen noch nicht funktionieren, zurück an den Anfang und nochmals genau lesen.

Die reverse Zone muss nämlich ebenfalls delegiert werden. Wenn man das `192.168.196`-Netz mit der Domain `linux.test` vom Provider zugeteilt bekommen hat, muss dieser auch `NS`-Einträge für die reversen-Zonen eintragen - genauso wie für die »normalen« Zonen. Wenn man die Kette von `in-addr.arpa` zurück zum eigenen Netz verfolgt, wird man ansonsten ein Loch in der Kette finden. Üblicherweise ist diese

Lücke beim eigenen Serviceprovider zu finden. Wenn dieses Loch gefunden wird, kontaktiert man am besten seinen Provider, damit dieser den Fehler behebt.

4.5.2 Man besitzt ein klassenloses Subnetz.

Dieser Punkt ist ein etwas schwieriger Punkt, aber Subnetze ohne Klasse sind in der letzten Zeit sehr wichtig geworden und vermutlich hat man davon eins, es sei denn man gehört zu einer mittelgrossen Firma.

Klassenlose Subnetze halten das Internet in diesen Tagen überhaupt noch am Leben. Bereits vor einigen Jahren gab es viele Probleme damit, dass die IP-Adressen knapp wurden. Die netten Leute im IETF (der Internet Engineering Task Force, die das Internet »verwalten") steckten Ihre Köpfe zusammen und lösten das Problem. Und mussten dafür einen Preis bezahlen - den Preis, dass man kleinere Netze als die der Klasse »C« bekommen kann - und dadurch kann einiges schiefgehen. Dazu kann ich nur raten unter

```
http://www.acmebw.com/askmrdns/00007.htm
```

nach einer guten Erklärung dieses Problems und wie man damit umgeht, zu fragen.

Gelesen? Ich werde hier nicht weiter darauf eingehen.

Der erste Teil des Problems ist, dass der eigene ISP die Techniken verstanden haben muss, die dort erklärt werden. Nicht alle kleinen ISPs verstehen es und setzen es auch um. Wenn dem so ist, muss man ihnen das eventuell erklären und vor allem sehr ausdauernd sein. Damit das funktioniert, sollte man allerdings sicher sein, dass man selber das Ganze verstanden hat ;-). Erst dann wird der Provider auch eine schöne reverse Zone auf seinem Server einrichten, die man mit `nslookup` überprüfen kann.

Der zweite und letzte Teil dieses Problems ist, dass man die Technik, die dahinter steht, richtig verstehen muss. Jeder der noch unsicher ist, sollte auf die Seite zurückgehen und es noch einmal lesen. Dann kann man sich daran machen, die eigene Klassenlose reverse-Zone einzurichten, wie es dort beschrieben wird.

Eine Falle wartet hier allerdings immernoch. Alte Resolver-Programme werden *nicht* in der Lage sein, dem `CNAME`-Trick in der resolve-Kette zu folgen und werden die Rechner nicht aufschlüsseln können. Das kann dann dazu führen, dass der Dienst den Rechner in eine falsche Zugriffsklasse einordnet und einem den Zugriff verweigert oder ähnliches. Wenn man auf so einen Service stößt, bleibt als einzige Möglichkeit (die ich kenne), dass der ISP einen PTR-Eintrag direkt in die klassenlose Zonen-Datei schreibt - als Ersatz für den CNAME-Trick.

Einige ISPs bieten andere Wege, um dieses Problem zu behandeln, an. Ein Beispiel können WWW-basierte Formulare sein, in die man seine reversen-Einträge machen kann oder andere automatisierte Systeme.

5 Eine existierendes Domainbeispiel

Leser haben mir vorgeschlagen, dass ich ein existierendes Beispiel einer funktionierenden Domain zusätzlich zu dem Lehrbeispiel hinzufüge.

Ich benutze dieses Beispiel mit Erlaubnis von David Bullock von LAND-5. Diese Dateien stammen vom 24. September 1996 und sind nachträglich von mir editiert worden, damit sie die Bind8-Restriktionen und Erweiterungen enthalten. Also unterscheiden sich diese Daten ein wenig von denen, die man erhält, wenn einer der LAND-5 Nameserver befragt wird.

5.1 /etc/named.conf (bzw. /var/named/named.conf)

Hier findet man die Zonenabschnitte, für die zwei benötigten reversen Zonen: Das 127.0.0-Netz und auch das LAND-5 `206.6.177`-Subnetz. Ausserdem ist eine primary-Zeile für die LAND-5-Forward-Zone `land-5.com` enthalten. Ausserdem ist zu beachten, dass David die Dateien nicht in ein Verzeichnis mit dem Namen `pz` legt, wie ich das in dieser HOWTO gemacht habe. Er benutzt ein Verzeichnis namens `zone`.

```
// Boot-Datei fuer den LAND-5 Nameserver

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

zone "." {
        type hint;
        file "root.hints";
};

zone "0.0.127.in-addr.arpa" {
        type master;
        file "zone/127.0.0";
};

zone "land-5.com" {
        type master;
        file "zone/land-5.com";
};

zone "177.6.206.in-addr.arpa" {
        type master;
        file "zone/206.6.177";
};
```

Sofern man dieses Beispiel in die eigene `named.conf`-Datei packt, wovon abzuraten ist, sollte man bitte auch noch ein »`notify no;`« in die Zonenabschnitte für die zwei `land-5`-Zonen schreiben, um Konflikte zu vermeiden.

5.2 /var/named/root.hints

Hierbei ist zu beachten, dass diese Datei relativ dynamisch ist. Die hier aufgelistete ist alt. Jetzt sollte besser eine mit dig neu erzeugte benutzt werden, wie bereits beschrieben wurde.

```
; <<>> DiG 8.1 <<>> @A.ROOT-SERVERS.NET.
; (1 server found)
;; res options: init recurs defnam dnsrch
;; got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 10
;; flags: qr aa rd; QUERY: 1, ANSWER: 13, AUTHORITY: 0, ADDITIONAL: 13
;; QUERY SECTION:
;;      ., type = NS, class = IN

;; ANSWER SECTION:
.                       6D IN NS        G.ROOT-SERVERS.NET.
```

```
.                        6D IN NS        J.ROOT-SERVERS.NET.
.                        6D IN NS        K.ROOT-SERVERS.NET.
.                        6D IN NS        L.ROOT-SERVERS.NET.
.                        6D IN NS        M.ROOT-SERVERS.NET.
.                        6D IN NS        A.ROOT-SERVERS.NET.
.                        6D IN NS        H.ROOT-SERVERS.NET.
.                        6D IN NS        B.ROOT-SERVERS.NET.
.                        6D IN NS        C.ROOT-SERVERS.NET.
.                        6D IN NS        D.ROOT-SERVERS.NET.
.                        6D IN NS        E.ROOT-SERVERS.NET.
.                        6D IN NS        I.ROOT-SERVERS.NET.
.                        6D IN NS        F.ROOT-SERVERS.NET.

;; ADDITIONAL SECTION:
G.ROOT-SERVERS.NET.      5w6d16h IN A    192.112.36.4
J.ROOT-SERVERS.NET.      5w6d16h IN A    198.41.0.10
K.ROOT-SERVERS.NET.      5w6d16h IN A    193.0.14.129
L.ROOT-SERVERS.NET.      5w6d16h IN A    198.32.64.12
M.ROOT-SERVERS.NET.      5w6d16h IN A    202.12.27.33
A.ROOT-SERVERS.NET.      5w6d16h IN A    198.41.0.4
H.ROOT-SERVERS.NET.      5w6d16h IN A    128.63.2.53
B.ROOT-SERVERS.NET.      5w6d16h IN A    128.9.0.107
C.ROOT-SERVERS.NET.      5w6d16h IN A    192.33.4.12
D.ROOT-SERVERS.NET.      5w6d16h IN A    128.8.10.90
E.ROOT-SERVERS.NET.      5w6d16h IN A    192.203.230.10
I.ROOT-SERVERS.NET.      5w6d16h IN A    192.36.148.17
F.ROOT-SERVERS.NET.      5w6d16h IN A    192.5.5.241

;; Total query time: 215 msec
;; FROM: roke.uio.no to SERVER: A.ROOT-SERVERS.NET.  198.41.0.4
;; WHEN: Sun Feb 15 01:22:51 1998
;; MSG SIZE  sent: 17  rcvd: 436
```

5.3 /var/named/zone/127.0.0

Hier ist nur das Minimum enthalten: Der obligatorische SOA-Eintrag und ein Eintrag, der 127.0.0.1 in `localhost` umwandelt. Beide Einträge sind nötig, allerdings sollte diese Datei nicht mehr enthalten. Sie wird vermutlich niemals mehr geändert werden müssen, es sei denn der Nameserver oder die hostmaster-Adresse ändert sich.

```
@               IN      SOA     land-5.com. root.land-5.com. (
                                199609203       ; Serial
                                28800   ; Refresh
                                7200    ; Retry
                                604800  ; Expire
                                86400)  ; Minimum TTL
                        NS      land-5.com.

1                       PTR     localhost.
```

5.4 /var/named/zone/land-5.com

Hier sieht man den notwendigen SOA-Eintrag und NS-Einträge. Man kann erkennen, dass er einen Zweit (Secondary)-Nameserver mit der Adresse `ns2.psi.net` hat. Es sollte übrigens *immer* ein zweiter Nameserver als Backup-Server zur Verfügung stehen. Außerdem kann man sehen, dass es einen Hauptrechner gibt, der `land-5` heißt und viele verschiedene Internetdienste anbietet - und dass dieser als Alternative zu A-Einträgen mit CNAMEs realisiert wurde.

Der SOA-Eintrag enthält die Informationen über den Ursprung der Zonendatei (`land-5.com`) und dass die Kontaktperson `root@land-5.com` ist. `hostmaster` ist die meistens benutzte Adresse für die Kontaktperson. Die Seriennummer ist in dem Format jjjjmmtt, wobei noch die Seriennummer des laufenden Tages angehängt wird; vermutlich handelt es sich um die sechste Version der Zonendatei vom 20. September 1996. Man sollte immer daran denken, dass die Seriennummer monoton ansteigen *muss*. Da dieses Beispiel eine *einstellige* Nummer für die Seriennnummer nutzt, können maximal neun Änderungen an einem Tag gemacht werden können. Darum empfehle ich die Verwendung von zwei Stellen für die Seriennummer.

```
@       IN      SOA     land-5.com. root.land-5.com. (
                        199609206       ; serial, todays date + todays serial #
                        8H              ; refresh, seconds
                        2H              ; retry, seconds
                        1W              ; expire, seconds
                        1D )            ; minimum, seconds
                NS      land-5.com.
                NS      ns2.psi.net.
                MX      10 land-5.com.  ; Primary Mail Exchanger
                TXT     "LAND-5 Corporation"

localhost       A       127.0.0.1

router          A       206.6.177.1

land-5.com.     A       206.6.177.2
ns              A       206.6.177.3
www             A       207.159.141.192

ftp             CNAME   land-5.com.
mail            CNAME   land-5.com.
news            CNAME   land-5.com.

funn            A       206.6.177.2

;
;       Workstations
;
ws-177200       A       206.6.177.200
                MX      10 land-5.com.   ; Primary Mail Host
ws-177201       A       206.6.177.201
                MX      10 land-5.com.   ; Primary Mail Host
ws-177202       A       206.6.177.202
                MX      10 land-5.com.   ; Primary Mail Host
ws-177203       A       206.6.177.203
                MX      10 land-5.com.   ; Primary Mail Host
ws-177204       A       206.6.177.204
                MX      10 land-5.com.   ; Primary Mail Host
ws-177205       A       206.6.177.205
```

```
                MX      10 land-5.com.   ; Primary Mail Host
; {Many repetitive definitions deleted - SNIP}
ws-177250       A       206.6.177.250
                MX      10 land-5.com.   ; Primary Mail Host
ws-177251       A       206.6.177.251
                MX      10 land-5.com.   ; Primary Mail Host
ws-177252       A       206.6.177.252
                MX      10 land-5.com.   ; Primary Mail Host
ws-177253       A       206.6.177.253
                MX      10 land-5.com.   ; Primary Mail Host
ws-177254       A       206.6.177.254
                MX      10 land-5.com.   ; Primary Mail Host
```

Bei genauerer Betrachtung der Nameserver von LAND-5, fällt auf, dass die Rechnernamen immer die Form `ws_`*nummer* haben. Seit einer der letzten der Bind-4-Versionen achtet der named auf Einschränkungen, welche Zeichen in einem Rechnernamen vorkommen dürfen. Da die angegeben Namen mit Bind-8 auf keinen Fall funktionieren würden, habe ich in der HOWTO die Unterstriche »_« durch einen Bindestrich »-« ersetzt.

Ich sollte noch darauf hinweisen, dass die Workstations keine individuellen Namen haben, sondern aus einem Prefix mit den angehängten letzten beiden Teilen der IP-Adresse bestehen. Das vereinfacht die Wartung zwar erheblich, es ist aber sehr unpersönlich.

Man kann außerdem sehen, dass `funn.land-5.com` ein Alias für `land-5.com` ist, dass aber ein A-Eintrag anstelle eines CNAME-Eintrages gewählt wurde. Wie ich oben bereits sagte, ist diese Vorgehensweise der Nutzung von CNAMES vorzuziehen.

5.5 /var/named/zone/206.6.177

Diese Datei wird weiter unten beschrieben.

```
@               IN      SOA     land-5.com. root.land-5.com. (
                                199609206       ; Serial
                                28800   ; Refresh
                                7200    ; Retry
                                604800  ; Expire
                                86400)  ; Minimum TTL
                        NS      land-5.com.
                        NS      ns2.psi.net.
;
;       Servers
;
1       PTR     router.land-5.com.
2       PTR     land-5.com.
2       PTR     funn.land-5.com.
;
;       Workstations
;
200     PTR     ws-177200.land-5.com.
201     PTR     ws-177201.land-5.com.
202     PTR     ws-177202.land-5.com.
203     PTR     ws-177203.land-5.com.
204     PTR     ws-177204.land-5.com.
205     PTR     ws-177205.land-5.com.
```

```
; {Many repetitive definitions deleted - SNIP}
250     PTR     ws-177250.land-5.com.
251     PTR     ws-177251.land-5.com.
252     PTR     ws-177252.land-5.com.
253     PTR     ws-177253.land-5.com.
254     PTR     ws-177254.land-5.com.
```

Die reverse Zone ist der unbeliebteste Teil der Einrichtung eines Nameservers. Sie dient dazu, den Rechnernamen herauszufinden, wenn nur die IP-Adresse bekannt ist. Wenn man zum Beispiel einen IRC-Server in Norwegen betreibt und nur Clients aus Norwegen und anderen Skandinavischen Ländern zulassen möchte, hat man die Möglichkeit, durch eine C-Bibliothek die IP-Nummer der gerade verbundenen Maschine zu bekommen, da diese mit jedem Datenpaket über das Netz geschickt wird. Jetzt kann man eine Funktion namens `gethostbyaddr()` aufrufen, die den Namen anhand einer übergebenen IP-Nummer herausfindet. `gethostbyaddr()` fragt dazu einen DNS-Server, der dann das DNS-System nach einem zu der IP-Nummer gehörigen Namen durchsucht. Wenn die Verbindung zum Beispiel von ws-177200.land-5.com ausgeht, liefert die C-Funktion die IP-Adresse 206.6.177.200. Um den Namen des Rechners herauszufinden, müssen wir `200.177.6.206.in-addr.arpa` finden. Der DNS-Server geht dann über die `arpa.`-Server, dann findet er `in-addr.arpa.`-Server, folgt der Route über 206, 6 und zum Schluss findet er den Server für die `177.6.206.in-addr.arpa`-Zone bei LAND-5. Dort wird er schlussendlich die Antwort bekommen, dass ein »`PTR ws-177200.land-5.com`«-Eintrag existiert. Das bedeutet, dass der zu der IP-Nummer `206.6.177.200` gehörende Name `ws-177200.land-5.com` ist. Genau wie die obige Beschreibung ist auch das ein wenig fiktiv - entspricht aber ungefähr dem tatsächlichen Ablauf.

Zurück zum IRC-Server-Beispiel. Der IRC-Server nimmt nur Verbindungen aus Skandinavischen Ländern wie `*.no`, `*.se` oder `*.dk` an. Der Rechner `ws-177200.land-5.com` passt zu keinem der genannten Länder, darum wird der Server die Verbindung ablehnen. Wenn es *keine* reverse Umwandlung für `206.6.177.200` durch die `inn-addr.arpa`-Zone gäbe, würde der Server den Rechnernamen nicht herausfinden können und müsste die IP-Nummer `206.6.177.200` mit `*.no`, `*.se` und `*.dk` vergleichen - keine Domain würde zutreffen.

Es gibt viele Leute, die behaupten, dass reverse Mappings nur für Server wichtig sind oder überhaupt keinen Sinn haben. Das ist falsch: Viele FTP, News, IRC und sogar einige HTTP (WWW)-Server akzeptieren *keine* Verbindungen von Rechnern ohne Namen. Also sind reverse Mappings ein *muss* bei der Verwaltung eines eigenen Nameservers.

6 Wartung

Es gibt einen Punkt, der bei der Administration von Nameservern beachtet werden muss - wenn man mal davon absieht, dass der Nameserver funktionieren sollte - und das ist die Aktualisierung der `root.hints`-Datei. Der einfachste Weg hierfür ist die Benutzung von dig. Zuerst startet man dig ohne Parameter, dann bekommt man die Datei `root.hints`, die zum eigenen Server gehört. Dann wird einer der aufgelisteten root-Server mit `dig @rootserver` befragt. Die Ausgabe dieses Befehls sieht erstaunlicherweise nach einer `root.hints`-Datei aus. Die Ausgabe wird in einer Datei gespeichert (`dig @e.root-servers.net . ns >root.hints.new`) und ersetzt dann die alte `root.hints`-Datei.

Nicht vergessen, im Anschluss an den Austausch der Cache-Datei den named neu zu laden.

Al Longyear hat mir das folgende Skript geschickt, das automatisch die `root.hints`-Datei aktualisieren kann. Man erstellt einen Eintrag in die Crontab, um es einmal im Monat zu starten und kann es dann vergessen. Das Skript geht davon aus, dass der E-Mail-Versand funktioniert und dass es einen Alias namens »hostmaster« gibt. Es muss noch verändert werden, damit es mit dem eigenen Setup übereinstimmt.

```
#!/bin/sh
```

```
#
# Dieses Skript aktualisiert einmal im Monat die Nameserver-Cache-Datei.
# Es wird durch einen Cron-Eintrag gestartet.
#
# Original von Al Longyear
# An bind8 angepasst von Nicolai Langfeldt
# Verschiedene Problem-Reports durch David A. Ranch
# Ping-Test vorgeschlagen von Martin Foster
# Übersetzung von Thomas "Balu" Walter
#
(
 echo "To: hostmaster <hostmaster>"
 echo "From: system <root>"
 echo "Subject: Automatisches Update der root.hints Datei"
 echo

 PATH=/sbin:/usr/sbin:/bin:/usr/bin:
 export PATH
 cd /var/named

 # Sind wir online? Ein Server beim Provider anpingen...
 case `ping -qnc some.machine.net` in
   *'100% packet loss'*)
        echo "Das Netz ist DOWN. root.hints wurde NICHT upgedated"
        echo
        exit 0
        ;;
 esac

 dig @rs.internic.net . ns >root.hints.new 2>&1

 case `cat root.hints.new` in
   *NOERROR*)
        # Funktioniert...
        :;;
   *)
        echo "Das Update der root.hints Datei ist FEHLGESCHLAGEN."
        echo "Dig hatte die folgende Ausgabe:"
        echo
        cat root.hints.new
        exit 0
        ;;
 esac

 echo "Die root.hints Datei wurde mit den folgenden Einstellungen aktualisiert:"
 echo
 cat root.hints.new

 chown root.root root.hints.new
 chmod 444 root.hints.new
 rm -f root.hints.old
 mv root.hints root.hints.old
 mv root.hints.new root.hints
 ndc restart
 echo
```

```
 echo "Der Nameserver wurde neu gestartet, um das Update abzuschliessen."
 echo "Die ursprüngliche root.hints Datei heisst jetzt /var/named/root.hints.old."
) 2>&1 | /usr/lib/sendmail -t
exit 0
```

Einige Leser haben mir geschrieben, dass die `root.hints`-Datei auch per FTP von internic zu bekommen ist. Man sollte aber nicht FTP benutzen, um die Datei zu aktualisieren. Die oben gezeigte Version ist »netter« gegenüber dem Netz und Internic.

7 Umstieg von der Version 4 zu Version 8

Dieser Abschnitt war ursprünglich eine Beschreibung, wie man bind 8 benutzt. Er stammte von David E. Smith (`dave@bureau42.ml.org`). Ich habe ihn an die aktuellen Gegebenheiten angepasst.

Es gibt hierzu nicht viel zu tun. Ausser dass die Datei `named.boot` jetzt `named.conf` heisst, hat sich nichts geändert. Bind 8 beinhaltet ausserdem ein Perl-Skript, dass die Dateien alten Stils in den neuen Stil umwandelt. Als Beispiel eine `named.boot`-Datei im alten Stil - geschrieben für einen Caching-only Nameserver:

```
directory /var/named
cache   .                                root.hints
primary 0.0.127.IN-ADDR.ARPA             127.0.0.zone
primary localhost                        localhost.zone
```

Man tippt in der Kommandozeile, im Verzeichnis `bind8/src/bin/named` (*es wird davon ausgegangen, dass man einen bind8 im Quellcode hat. Im Falle eines Binärpaketes sollte das Skript irgendwo anders liegen - aber ich bin mir nicht sicher wo...*):

```
./named-bootconf.pl < named.boot > named.conf
```

Dieses erzeugt eine `named.conf`-Datei:

```
// generated by named-bootconf.pl

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

zone "." {
        type hint;
        file "root.hints";
};

zone "0.0.127.IN-ADDR.ARPA" {
        type master;
        file "127.0.0.zone";
};

zone "localhost" {
        type master;
        file "localhost.zone";
};
```

Dieses funktioniert für alle Einstellungen, die in einer `named.boot`-Datei gemacht werden. Leider werden neue und verbesserte Konfigurationsoptionen von bind 8 nicht unterstützt. Hier ist eine erweiterte `named.conf`-Datei, die dasselbe macht, allerdings um ein paar Optionen erweitert wurde.

```
// Dies ist eine Konfigurationsdatei für den named
// (für BIND 8.1 oder später). Sie sollte normalerweise
// als /etc/named.conf installiert werden. Der einzige
// Unterschied zu der einfachsten named.conf-Version
// (zusätzlich zu diesem Kommentar :) ist, dass der
// Kommentar der directory-Zeile entfernt wurde, da
// ich bereits die Zonendateien in /var/named hatte.

options {
        directory "/var/named";
        datasize 20M;
};

zone "localhost" IN {
        type master;
        file "localhost.zone";
};

zone "0.0.127.in-addr.arpa" IN {
        type master;
        file "127.0.0.zone";
};

zone "." IN {
        type hint;
        file "root.hints";
};
```

In der Bind 8 Distribution findet man im Verzeichnis `bind8/src/bin/named/test` diese und andere Dateien, die die meisten Leute nur noch anpassen brauchen und die dann sofort benutzt werden können.

Das Format für die Zonendateien und die `root.hints`-Datei ist identisch zur alten Version. Genauso wie die Befehle um sie zu aktualisieren.

8 Fragen und Antworten

Lest bitte diesen Abschnitt, bevor Ihr mir mailt.

1. Mein named verlangt eine `named.boot`-Datei

   Du liest die falsche HOWTO. Benutze den Bind Version 8 oder lies die alte englischsprachige HOWTO, die sich mit Bind 4 beschäftigt unter folgender Adresse:

   ```
   http://www.mathh.uio.no/~janl/DNS/
   ```

2. Wie benutze ich DNS aus einer Firewall-geschützten Umgebung?

   Ein Tipp: `forward only;`, eventuell wird auch noch

   ```
   query-source port 53;
   ```

innerhalb des Optionen-Teils der `named.conf`-Datei benötigt, wie bereits im Beispiel in dem 3-Abschnitt beschrieben wurde.

3. Wie bringe ich den DNS dazu, durch eine Anzahl verfügbarer Adressen für einen Service zu rotieren, um einen Load-Balancing-Effekt oder ähnliches zu erzielen (zum Beispiel für www.ausgelastete.site)?

   Hierzu macht man mehrere A-Einträge für www.ausgelastete.site und benutzt bind 4.9.3 oder später. Dann wird der Nameserver automatisch durch die Adressen rotieren. Das funktioniert *nicht* mit älteren Versionen des Bind.

4. Ich möchte DNS in einem (geschlossenen) Intranet benutzen. Wie macht man das?

   Man läßt die `root.hints`-Datei weg und erstellt nur Zonendateien. Das bedeutet natürlich auch, dass man nicht andauernd eine neue hints-Datei besorgen muss.

5. Wie erstelle ich einen Secondary (Slave) Nameserver?

   Wenn der primäre (Master-)Server die Adresse 127.0.0.1 hat, fügt man eine Zeile wie die folgende in die `named.conf`-Datei des Secondary-Servers ein:

```
zone "linux.test" {
      type slave;
      file "sz/linux.test";
      masters { 127.0.0.1; };
};
```

   Man kann auch mehrere Master-Server für die Zone angeben, indem die zusätzlichen Server durch Semikolons getrennt zur `masters`-Liste hinzugefügt werden.

6. Bind soll auch laufen, wenn keine Verbindung zum Netz besteht.

   Es gibt drei Antworten zu diesem Problem:

   - Diese Mail stammt von Ian Clark (`ic@deakin.edu.au`); er beschreibt seine Vorgehensweise:

     Bei mir läuft der named auf einer »Masquerading«-Maschine. Ich habe zwei `root.hints`-Dateien. Eine davon heisst `root.hints.real` und enthält die richtigen root-Servernamen, während die zweite `root.hints.fake` heisst und wie folgt aussieht:

```
; root.hints.fake
; Diese Datei enthält keine Informationen
```

     Wenn ich offline gehe, wird die `root.hints.fake`-Datei über die Datei `root.hints` kopiert und der named neu gestartet.

     Sobald ich online bin, kopiere ich die Datei `root.hints.real` über die Datei `root.hints` und starte den named neu.

     Dieses geschieht automatisch in den Dateien `ip-down` und `ip-up`.

     Bei der ersten Namensanfrage im offline-Modus zu einer Domain, zu der der named keine Daten hat, wird die folgende Nachricht in die messages geschrieben:

```
Jan 28 20:10:11 hazchem named[10147]: No root nameserver for class IN
```

Aber damit kann ich leben.

So funktioniert es bei mir. Ich kann den Nameserver für lokale Rechner benutzen, wenn ich offline bin - ohne die timeout-Wartepause für externe Domainnamen und wenn ich online gegangen bin, arbeitet die Auflösung der externen Domainnamen wie erwartet.

- Ausserdem habe ich von Karl-Max Waner Informationen darüber erhalten, wie sich die Kombination aus Bind, NFS und dem Portmapper bei einem Rechner verhält, der meistens offline ist:

  Ich benutze üblicherweise einen eigenen named auf jeder Maschine, die nicht durchgehend über ein Modem mit dem Internet verbunden ist. Der Nameserver fungiert hierbei ausschliesslich als Cache, er besitzt keine »area of authority« und erfragt alle seine Informationen von den Servern aus der `root.cache`-Datei. Bei einer Slackware-Distribution wird er üblicherweise vor dem `nfsd` und dem `mountd` gestartet.

  Auf einem meiner Computer (einem Libretto 30 Notebook) hatte ich das Problem, dass ein Mount-Versuch von einem anderen System in meinem lokalen Netz gelegentlich - aber meistens eher nicht funktionierte. Ganz egal, ob ich PLIP, eine PCMCIA-Ethernet-Karte oder PPP über ein serielles Interface benutzte.

  Nach einiger Zeit des Ratens und Experimentierens fand ich heraus, dass der named in Kombination mit dem `nfsd` und dem `mountd`, die beim Hochfahren den Portmapper starten, Probleme erzeugte. Als ich den named nach dem `nfsd` und dem `mountd` startete, trat dieser Fehler nicht wieder auf.

  Da keine Nachteile bei dieser veränderten Boot-Reihenfolge zu erwarten sind, empfehle ich jedem dasselbe zu machen, um eventuelle Schwierigkeiten zu vermeiden.

- Als Abschluss sollte noch gesagt werden, dass es hierüber eine HOWTO unter der folgenden Adresse gibt:

  `http://www.acmebw.com/askmrdns/#linux-ns`

  Hier wird allerdings noch der Bind4 beschrieben; also sollte man das, was er dort schreibt, nach Bind8 übersetzen.

7. Wo speichert der caching-Nameserver seine Daten? Gibt es eine Möglichkeit, die Größe dieses Caches zu verändern?

   Der Cache wird komplett im Speicher gehalten und *niemals* auf die Festplatte geschrieben. Jedesmal, wenn der named beendet wird, gehen auch die gecachten Daten verloren. Der Cache ist *nicht* steuerbar. Der named verwaltet ihn nach einigen einfachen Regeln - mehr nicht. Man kann den Cache oder dessen Größe in keinster Weise kontrollieren. Wer das dennoch möchte, kann diesen »Fehler« durch Handarbeit im named-Sourcecode ändern. Natürlich wird dieses nicht empfohlen.

8. Speichert der named den Cache bei einem Neustart? Gibt es eine Möglichkeit, diese Option zu aktivieren?

   Nein, der named speichert den Cache *nicht*, wenn er beendet wird. Bei jedem Neustart muss der named daher den Cache von Grund auf neu aufbauen. Es gibt *keine* Möglichkeit, ihn zum Speichern seiner Cache-Daten in einer Datei zu bewegen. Wer das möchte, kann diesen »Fehler« durch Handarbeit im named-Sourcecode ändern, was natürlich nicht empfohlen wird.

9. Wie bekomme ich eine Domain? Ich möchte eine eigene Domain mit dem Namen (zum Beispiel) `linux-ist-das-beste-betriebssystem.de` erstellen. Wie kann ich diese Domain für mich registrieren lassen?

   Hierfür sollte man seinen Netzwerkprovider um Rat bitten. Er sollte in der Lage sein, hierbei zu helfen. Als Hinweis sei noch gesagt, dass in den meisten Teilen der Welt für Domains Geld bezahlt werden muss.

9 Wie man ein besserer DNS-Administrator wird.

Es gibt »richtige« Dokumentation - sowohl online als auch gedruckt. Einige dieser Texte sollte man gelesen haben, um den Schritt von einem kleinen zu einem grossen DNS-Administrator machen zu können. Das gedruckte Standardwerk ist:

DNS und BIND
Paul Albitz, Cricket Liu
O'Reilly
ISBN 3-89721-160-2

Ich habe es gelesen - es ist ausgezeichnet. Es beschreibt sowohl bind4.9 als auch bind9.

Einen Abschnitt über DNS gibt es auch in:

TCP/IP Netzwerk-Administration
Craig Hunt
O'Reilly
ISBN 3-89721-110-6

Ein anderes Muss für gute DNS-Administratoren ist

Zen and the Art of Motorcycle Maintenance
Robert M. Pirsig
ISBN 0-68805-230-4

Online wird man Informationen unter folgenden Adressen finden

- `http://www.dns.net/dnsrd/` (DNS Quellen-Verzeichnis)
- `http://www.isc.org/bind.html`

Hier findet man eine FAQ, das Referenzhandbuch »Bind Operations Guide« sowie viele Artikel, Protokolldefinitionen und DNS-Hacks. Diese und die meisten anderen, der unten noch aufgelisteten RFCs, sind ausserdem im Bind-Paket enthalten. Ich habe die wenigsten von ihnen gelesen, aber ich bin auch kein grosser DNS-Admin. Arnt Gulbrandsen auf der anderen Seite hat den BOG gelesen und findet ihn atemberaubend :-).

Die Newsgruppe

`comp.protocols.tcp-ip.domains`

enthält Diskussionen über das DNS.

Ausserdem gibt es eine Reihe von RFCs, deren wichtigsten vermutlich die folgenden sind:

RFC 2052

A. Gulbrandsen, P. Vixie, *A DNS RR for specifying the location of services (DNS SRV)*, October 1996

RFC 1918

Y. Rekhter, R. Moskowitz, D. Karrenberg, G. de Groot, E. Lear, *Address Allocation for Private Internets*, 02/29/1996.

RFC 1912

D. Barr, *Common DNS Operational and Configuration Errors*, 02/28/1996.

RFC 1912 Errors

B. Barr *Errors in RFC 1912*, verfügbar unter folgender Adresse:

`http://www.cis.ohio-state.edu/~barr/rfc1912-errors.html`

RFC 1713

A. Romao, *Tools for DNS debugging*, 11/03/1994.

RFC 1712

C. Farrell, M. Schulze, S. Pleitner, D. Baldoni, *DNS Encoding of Geographical Location*, 11/01/1994.

RFC 1183

R. Ullmann, P. Mockapetris, L. Mamakos, C. Everhart, *New DNS RR Definitions*, 10/08/1990.

RFC 1035

P. Mockapetris, *Domain names - implementation and specification*, 11/01/1987.

RFC 1034

P. Mockapetris, *Domain names - concepts and facilities*, 11/01/1987.

RFC 1033

M. Lottor, *Domain administrators operations guide*, 11/01/1987.

RFC 1032

M. Stahl, *Domain administrators guide*, 11/01/1987.

RFC 974

C. Partridge, *Mail routing and the domain system*, 01/01/1986.