

Linux im Netzwerk - ein "Mini"-Tutorial (Teil 4)

[Jana Jaeger](#)

Inhaltsverzeichnis

- [... wo bist Du denn?](#)
- [Struktur des DNS und Ablauf einer Suchanfrage](#)
- [Konfiguration des DNS-Servers](#)
 - [named.conf - was hat es damit auf sich?](#)
 - [Klein, aber wichtig - noch mehr Konfigurationsdateien](#)
- [Literatur](#)

Bevor das in [&content/server/fire2.html">Teil 2](#) vorgestellte Basisnetz seinen Betrieb aufnehmen und vor allen Dingen mit dem restlichen Internet ungehindert kommunizieren kann, muß dringend noch ein Nameserver in unserem Netzwerk eingerichtet werden. Dieser Teil des Netzwerktutorials behandelt die wichtigsten Schritte zum Aufsetzen eines lokalen Nameservers und setzt sich mit der Bedeutung der einzelnen Konfigurationsdateien auseinander.

... wo bist Du denn?

Ein wichtiger Service, der unserem Beispielnetz noch dringend gefehlt hat, ist der Domain Name Service (kurz DNS) zur Auflösung der Rechnernamen in IP-Adressen.

Ohne DNS wäre das vernetzte Leben eine einzige Qual, weil man sich für jeden einzelnen Rechner eine numerische Adresse merken müßte, denn Rechner kommunizieren nur über ihre numerische Adresse miteinander. Ähnlich wie Telefonnummern kann man sich solche Dinge schlecht merken. Viel praktischer ist es, die Nummern einem charakteristischen Namen zuzuordnen. Die Nameserverdienste übernehmen die Umsetzung der Namen in die korrekte Adresse und umgekehrt. Die Fähigkeit, umgekehrt auch numerische Adressen in Namen umzusetzen, ist besonders wichtig, da sich hinter einem Namen wie `ftp.de.kernel.org` nicht nur eine Maschine (mit einer IP-Adresse), sondern gleich mehrere verbergen können (193.189.224.13, 132.230.1.90, 134.108.34.10, 131.159.72.23, um genau zu sein). Außerdem wechseln IP-Adressen relativ oft, die Namen der Maschinen sollen aber gleich bleiben - hier vermittelt DNS.

Der Nameserver in unserem kleinen Netzwerk wird lokal eingerichtet und zwar so, daß er die Namen und IP-Adressen von `home.nil` verwaltet und DNS-Anfragen nach Internetadressen und das **Cachen** (Abspeichern der zuletzt aus diesem Netz heraus angefragten Internetadressen) beherrscht. Wenn Sie mehrmals am Tag bei <http://portal.suse.de> vorbeischauen, wird das Ergebnis der ersten DNS-Abfrage im Cache abgelegt und in Zukunft darauf zurückgegriffen.

Um später mit den Konfigurationsdateien zurechtzukommen, fehlen noch die Definitionen einiger grundlegender Begriffe:

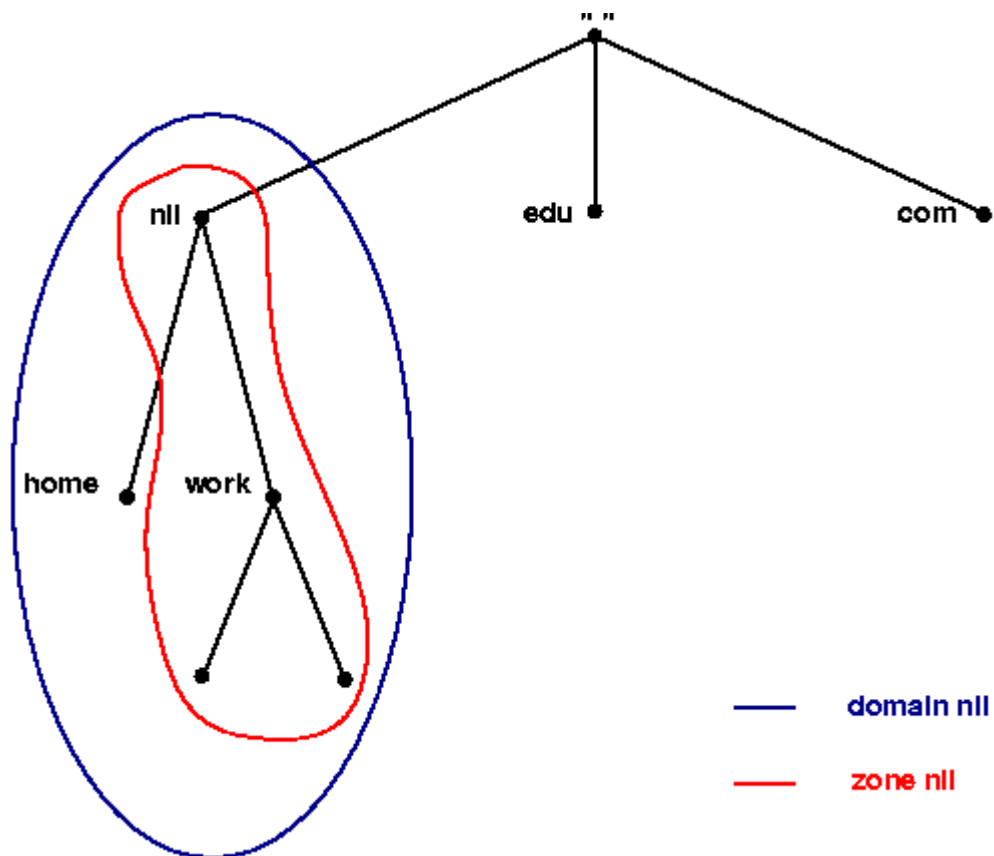
- **(Domain) Name Space** - platt gesagt ist dies der gesamte Adressraum der im Internet zugänglichen Adressen. Auf die Grafik übertragen würde dies den gesamten Baum mit allen

möglichen Verzweigungen bezeichnen.

- **Authority** - ein Nameserver, der exklusiv die für die Daten einer Domain verantwortlich ist, hat die "Authority".
- **Domain** - Untereinheiten des DNS Baums (in der Abbildung `nil`, `org`, `com`, `edu`, `home`, `work` . . . , die wiederum in Untereinheiten unterteilt sein können. Zum Beispiel ist die Domain `nil` wieder in die Subdomains `home` und `work` unterteilt.
- **Zone** - Untermenge einer Domain. Am konkreten Beispiel: Die `nil` Domain umfaßt die komplette Domain mit allen Subdomains (`home` und `work`), die Zone aber nur die Daten von `nil` und `work.nil`, da die Verantwortlichkeit (Authority) für `home` an den Netzbetreiber von `home` delegiert ist.
- **Zone Transfer** - wenn für einen Nameserverdienst nicht nur eine Maschine, sondern eine primäre (primary master) und eine sekundäre (secondary master) eingetragen sind, synchronisiert sich die sekundäre Maschine in periodischen Abständen mit der primären Maschine. Dabei werden alle Daten über die Zone, für die der primäre Nameserver die Verantwortung trägt, an den sekundären Nameserver übertragen. Dies nennt sich dann "Zone Transfer".

Struktur des DNS und Ablauf einer Suchanfrage

DNS Server gibt es nicht nur einen, sondern eine ganze Reihe überall über die ganze Welt verstreut. Alle stehen miteinander in Kontakt. Da DNS hierarchisch gegliedert sind, folgen die DNS-Abfragen quasi einer Kaskade von oben nach unten.

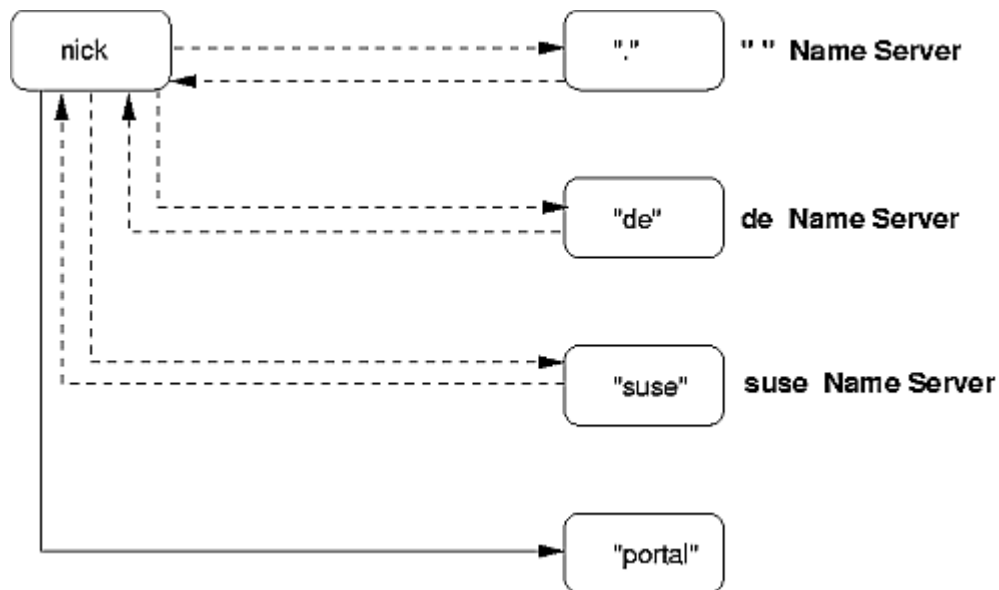


Die DNS-Hierarchie ist baumartig aufgebaut. Die oberste aller Domains ist ".". Ihr unterstehen wiederum zahlreiche Top Level Domains (TLD) wie `nil`, `org`, `com`, `edu`, `net`, `de`. Von

diesen Ästen aus verzweigen sich wieder weitere... .

Angenommen aus dem lokalen Netz läuft eine Anfrage nach einer bestimmten Internetadresse, hier zum Beispiel `portal.suse.de`. Zuerst schaut der Nameserver in seinen eigenen Cache. Wenn die Adresse schon hier gespeichert wurde, ist die Anfrage zuende, ehe sie richtig begonnen hat :-). Weiß `named` aber noch komplett gar nichts damit anzufangen, beginnt es damit, am linken Ende der Adresse Angaben "abzuschneiden". Als nächstes versucht es, über `suse.de` etwas herauszufinden. Wenn also im Cache schon einmal eine Adresse für `www.suse.de` liegt, wird die Suche dadurch abgekürzt, daß jetzt im Bereich `suse.de` nur noch die Maschine `portal` gesucht werden muß. War die Suche nach `suse.de` nicht erfolgreich, wird weiter nach `de` gesucht. Schlußendlich gelangt man dann zu `."`. `."` oder `root` ist ein Analog zum Wurzelverzeichnis `/"` im Unix-Dateibaum.

Die zu `."` gehörige Adresse ist dem System bekannt, im System liegt unter `/var/named/root.hint` eine Aufzählung der weltweit bekannten Rootserver. Der angefragte Rootserver wird jetzt noch keine Antwort bezüglich `portal.suse.de` wissen. Aber er verfügt über eine Datenbank mit den zuständigen Servern für die Top Level Domains. An den anfragenden `nick` kann er jetzt sein Wissen um die Adresse des für `"de"` zuständigen Nameservers weiterleiten. `nick` fragt nun den `"de"` Nameserver nach dem Zuständigen für `"suse.de"`. Nachdem diese Auskunft wiederum zu `nick` durchgedrungen ist, fragt `nick` den Nameserver von `"suse"` nach der Adresse von `portal.suse.de`. Sobald `nick` dies erfahren hat, kann er endlich mit dem Server `portal.suse.de` in Kontakt treten. Da alle diese Zwischenstationen einer solchen Anfrage wiederum über einen Cache verfügen, werden spätere gleichlautende Anfragen ungleich viel schneller beantwortet werden, da der "Dienstweg" jetzt wesentlich kürzer ist.



Konfiguration des DNS-Servers

Die Namens- und Adressauflösung wird unter Unix mit einem Programm namens `named` vorgenommen. `named` ist Teil des BIND Pakets (für SuSE 7.1 `bind8` in der Serie `n`).

Vorraussetzung für eine erfolgreiche `named`-Konfiguration ist ein korrekt konfiguriertes Netzwerk und folgende Einträge in den Konfigurationsdateien `/etc/hosts`, `/etc/resolv.conf` und `/etc/nsswitch.conf`, die auf den entsprechenden Nameserver hinweisen.

`/etc/hosts`

In den einzelnen `/etc/hosts` Dateien auf sämtlichen Maschinen im Netz können alle Einträge

bis auf

```
127.0.0.1 localhost
```

wegfallen, da jetzt die Verwaltung aller Rechnernamen und IP-Adressen zentral über den Nameserver erfolgt.

```
/etc/resolv.conf
```

... ist das Netzwerk wie in [&content/server/fire2.html">Teil 2](#) konfiguriert, müssen hier keine Änderungen gemacht werden. Der Inhalt sollte genau so aussehen:

```
search home.nil
nameserver 192.168.17.1
```

```
/etc/nsswitch.conf
```

In dieser Datei muß eine Zeile vorkommen, die ungefähr so lautet:

```
hosts: files dns
```

Diese Einstellung wurde aber bei der Netzwerkkonfiguration mit *YaST2* automatisch vorgenommen.

named.conf - was hat es damit auf sich?

Die zentrale Konfigurationsdatei des Nameservers ist die `named.conf`. Unter SuSE Linux liegt sie im Pfad `/etc/named.conf`, bei anderen Distributionen (Debian) kann sie auch unter `/etc/bind/named.conf` liegen. Die unten in Auszügen gezeigte Konfigurationsdatei ist die Beispielkonfiguration für `named`, wie sie mit SuSE Linux ausgeliefert wird. Vom Prinzip her unterscheidet sie sich durch nichts von anderen, allenfalls sind einige Kommentare anders. Kommentare sind wie immer mit `"#"` markiert. Sollen die Einstellungen aktiviert werden, genügt es, das einleitende `#` Zeichen zu entfernen und wenn nötig, die IP-Adressen der Wahl einzutragen. In diesem Beispiel werden nur die elementarsten Einträge vorgestellt, die schon einen funktionsfähigen Nameserver garantieren. Außer der Weiterleitung von Anfragen an einen anderen Nameserver (Forwarding) sollten in unserem einfachen Fall keine besonders fortgeschrittenen Funktionen aktiviert werden.

```
#
# overall options of the server
#
options {
    directory "/var/named";
    # the default is to fail, if the master file is not correct
    check-names master warn;
```

Der Eintrag `directory` weist auf das Verzeichnis, in dem alle relevanten weiteren Konfigurationsdateien liegen.

```
pid-file "/var/run/named.pid";
statistics-interval 0;
cleaning-interval 720;

datasize default;
stacksize default;
coresize default;
files unlimited;
recursion yes;
```

```
multiple-cnames no;
```

Diese Einstellungen sind eher etwas für fortgeschrittene Konfigurationen, sie stören so nicht, sollten aber auch nicht einfach verändert werden, wenn es dafür nicht einen guten Grund gibt.

```
# list of DNS servers to ask
#forwarders {
#    192.168.0.10;
#    192.168.0.20;
#    192.168.0.30;
#};
```

```
};
```

Unter `forwarders` werden die Nameserver aufgelistet, die als nächstes angefragt werden sollen, wenn Ihr lokaler Nameserver eine Adresse nicht von selbst auflösen kann. Im Normalfall wird hier der Nameserver Ihres Internet Service Providers angegeben. Es müssen nicht unbedingt mehrere Server angegeben werden, einer reicht. Aber die Angabe mehrerer Forwarder fängt eventuelle Ausfälle des einen oder anderen Servers ab. In diesem Fall lieber ein wenig mehr Redundanz als zuwenig einplanen, um im Zweifelsfall einen funktionierenden Nameservice zu gewährleisten.

```
# the default is to listen on port 53 on all available interfaces
# you can also give a detailed list:
listen-on { 127.0.0.1; 192.168.17.1; };
#listen-on port 1234 { !1.2.3.4; 1.2/16; };
```

`listen-on` bietet die Möglichkeit, Datenabfragen nur auf bestimmten Interfaces zuzulassen. Sobald der einzurichtende Nameserver, wie in unserem Fall, auch die Namensauflösung im lokalen Netz übernehmen soll, müssen die lokalen Konfigurationsdaten vor externem Einblick geschützt werden. Eine sinnvolle Einstellung, um die Daten des internen Netzes vor externem Zugriff zu schützen, ist die ausschließliche Angabe der nach innen gerichteten Interfaces, also `nick`'s interner Adresse und seiner Loopbackadresse. Jetzt wird `nick` ausschließlich DNS-Abfragen beantworten, die aus dem lokalen Netz kommen.

```
#
# predefined access control lists (acl):
# "any"          allows all hosts
# "none"         denies all hosts
# "localhost"   allows the IP addresses of all interfaces of the system
# "localnets"  allows any host on a network of the local interfaces
#
# defining an additional ACL:
#acl can_download { 192.168.0.17; 192.168.0.18; };
```

Ein anderer Mechanismus, um die Daten aus dem internen Netz vor Zugriffen von außen zu schützen, ist die Verwendung sogenannter "Access Control Lists" oder ACLs. DNS-Abfragen und Zonentransfers können so bestimmten vordefinierten oder selbstdefinierten Gruppen von Hosts erlaubt werden. "any" erlaubt jedem - egal ob intern oder extern - Zugriff. "none" sperrt alle aus. "localhost" erlaubt nur Anfragen von allen Interfaces der lokalen Maschine und "localnets" erlaubt von allen Hosts Anfragen, die sich in einem Netz befinden, zu dem der Rechner Netzzugang hat. Hier würden dann auch alle Rechner im Netz des Providers, zu dem `nick` über seine zweite Netzwerkkarte Kontakt hat, Zugriff auf unsere internen Konfigurationsdaten bekommen. Also bleibt letztendlich entweder die Definition einer eigenen ACL mit den Adressen aller Rechner im internen Netz nach dem Muster wie:

```
acl query { 192.168.17.1; 192.168.17.2; 192.168.17.3; 192.168.17.4; }
; ,
```

die Verwendung der `acl "localnets"` in Kombination mit `listen-on` auf den nach innen gerichteten Interfaces, oder die bloße Absicherung über `listen-on`. Im Beispiel wurde der

redundante Ansatz mit `acl` und `listen-on` gewählt.

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

zone "localhost" IN {
    type master;
    file "localhost.zone";
    check-names fail;
    allow-update { none; };
};

zone "0.0.127.in-addr.arpa" IN {
    type master;
    file "127.0.0.zone";
    check-names fail;
    allow-update { none; };
};

zone "home.nil" IN {
    type master;
    file "home.nil.zone";
    allow-query { localnets; };
    allow-transfer { none; };
    notify no;
};

zone "17.168.192.in-addr.arpa" IN {
    type master;
    file "192.168.17.zone";
    allow-query { localnets; };
    allow-transfer { none; };
    notify no;
};
```

Dies ist das absolute Herzstück der `named.conf`. Hier werden die einzelnen zu durchsuchenden Zonen zugeordnet. `zone "."` weist in diesem Fall auf eine Datei `root.hint`, die die Namen und IP-Adressen aller weltweiten Rootnameserver enthält.

Findet der Nameserver während einer Suche in seinem Cache keinen wirklich nützlichen Eintrag, wird der DNS-Baum von oben nach unten durchsucht - und in `root.hint` stehen quasi die Adressen, bei denen die Suche angefangen werden soll ;-).

`zone "localhost"` und `zone "0.0.127.in-addr.arpa"` weisen beide auf die Zoneninformationen des lokalen Rechners. Mehr zu den Inhalten dieser Dateien im nächsten Abschnitt.

`zone "home.nil"` und `zone "17.168.192.in-addr.arpa"` verwalten die Zoneninformationen über das lokale Netz. Im Prinzip enthalten diese Dateien nichts, was nicht auch in den `/etc/hosts` Dateien der einzelnen lokalen Maschinen schon gespeichert wurde. Deren Inhalt ist jetzt bis auf die Zeile

```
127.0.0.1 localhost
```

überflüssig und könnte gelöscht werden. Der zentralistische Ansatz über DNS hat den Vorteil, daß bei Erweiterung des lokalen Netzes jetzt nicht mehr auf jeder Maschine einzeln die `/etc/hosts` Dateien geändert werden müssen, sondern nur einmal zentral auf der Maschine, die den Nameserver darstellt. Je weniger Dateien geändert werden müssen, desto geringer die Wahrscheinlichkeit eines dummen, kleinen Tippfehlers in einer der Konfigurationsdateien. Letztendlich bedeutet das dann wiederum ein stressfreieres Leben für den Administrator ;-). Damit die Daten über die lokale Netzwerkkonfiguration nicht von externen Servern abgefragt werden können, müssen noch einige Vorsichtsmaßnahmen

ergriffen werden. Dazu unten mehr ...

Klein, aber wichtig - noch mehr Konfigurationsdateien

root.hint

Die Datei /var/named/root.hint enthält die Adressen aller Rootserver weltweit. Sie sieht im Allgemeinen so aus:

```
;      This file holds the information on root name servers needed to
;      initialize cache of Internet domain name servers
;      (e.g. reference this file in the "cache . <file>"
;      configuration file of BIND domain name servers).
;
...

;
;
; formerly NS.INTERNIC.NET
;
.          3600000   IN   NS      A.ROOT-SERVERS.NET.
A.ROOT-SERVERS.NET. 3600000   A    198.41.0.4
;
; formerly NS1.ISI.EDU
;
.          3600000   NS    B.ROOT-SERVERS.NET.
B.ROOT-SERVERS.NET. 3600000   A    128.9.0.107
;
; formerly C.PSI.NET
;
...

.          3600000   NS    M.ROOT-SERVERS.NET.
M.ROOT-SERVERS.NET. 3600000   A    202.12.27.33
; End of File
```

Die Beispieldatei, die Teil eines aktuellen *bind* Paketes ist, muß nicht weiter abgewandelt werden. Allerdings ist diese Datei dynamisch in dem Sinn, daß ab und zu neue Rootserver eingeführt werden oder schon existierende zu einer neuen Adresse hin "umziehen". Eventuell schadet es nicht, im Laufe der Zeit ab und zu mal die Ausgaben der folgenden Befehle mit den Angaben in der eigenen *root.hint* Datei zu vergleichen und sie dann ggf. anzupassen. Dazu fragt man einen der Rootserver (zum Beispiel A) nach dem aktuellen Inhalt seiner *root.hint* Datei und leitet, wenn sich die Ausgabe von der eigenen *root.hint* unterscheiden sollte, die Ausgabe des folgenden Befehls nach *root.hint* um.

```
nick:~ # dig @A.ROOT-SERVERS.NET
```

Nachdem sie die alte *root.hint* zur Sicherheit vorher noch kopiert haben

```
nick:~ # cp root.hint root.hint.old
```

können Sie die Ausgabe umleiten:

```
nick:~ # dig @A.ROOT-SERVERS.NET > root.hint
```

Für denjenigen, der Arbeitsrationalisierung liebt, ist am Ende des [DNS-HOWTOs](#) ein kleines

Shellskript vorgestellt, das den Abgleich der `root.hint` automatisiert.

localhost.zone und 127.0.0.zone

Die Datei `/var/named/localhost.zone` enthält Daten über den lokalen Rechner, auf dem der Nameserver läuft. Der typische Aufbau einer solchen Datei im Detail:

```
$TTL 3D
@           1D IN SOA      @ root (
                                42           ; serial (d. adams)
                                3H           ; refresh
                                15M          ; retry
                                1W           ; expiry
                                1D )         ; minimum

          1D IN NS       @
          1D IN A        127.0.0.1
```

Der erste Eintrag in der Datei, "\$TTL . ." gibt die Time-to-Live Spanne (Lebensdauer) der enthaltenen Daten an. Die abfragenden Maschinen werden die Zonendaten bis zu drei Tagen im Cache behalten und danach von Neuem nach den Daten fragen. "@" weist auf den Ursprung (Origin), dem diese Datei in der DNS Hierarchie zugeordnet ist, hier ganz einfach `localhost`, wie in der `named.conf` schon angegeben. Die erste Zeile könnte also genausogut so lauten:

```
localhost           1D IN SOA ...
```

In Zusammenhang mit den Zonendateien wird noch ein Begriff wichtig, der oben bei den Definitionen noch gar nicht erwähnt wurde, die **Resource Records (RR)**. Nahezu alle Einträge in den `.zone`-Dateien setzen sich aus solchen RRs zusammen. Die wichtigen, die uns in unserer Einfachstkonfiguration begegnen, hier einmal aufgeschlüsselt:

SOA Record

SOA kürzt Start Of Authority ab. Der SOA gibt an, wem die Authority zu dieser Domain zugeordnet ist, d.h. welcher Nameserver die besten Informationen über unsere Domain hat. Hier ist dies `localhost`, also der Rechner, auf dem der Nameserver aufgesetzt wurde. Ausserdem wird hier noch die E-Mailadresse desjenigen angegeben, der für die Administration des Servers zuständig ist. Diese Information wird keinem Maschinenwesen etwas nützen, aber ein Mensch aus Fleisch und Blut bekommt mit dieser Adresse im Zonefile wenigstens einen Anhaltspunkt, wo er sich beschweren kann, falls die gelieferten Daten nicht einwandfrei sein sollten.

NS Record

NS steht für Nameserver. NS Records geben Auskunft über den/die Nameserver einer Domain.

A Record

A Records schlüsseln die Zuordnung von Namen zu IP-Adressen auf. Im Beispiel ist der A Record der Adresse `localhost` schlicht und einfach `127.0.0.1`.

PTR Record

PTR Records sind die einfache Umkehrung von A Records, also die Zuordnung von IP-Adressen zu bestimmten Namen. Hier: Host 1 im Subnetz `0.0.127.in-addr.arpa` ist `localhost`.

Die restlichen Angaben in der Datei beziehen sich auf den Fall, daß neben dem einen Nameserver in unserem Netz noch andere sekundären Server aufgesetzt wurden, die dieselben Daten vorhalten und

sich ab und zu mit dem Haupt(Master-)Server abgleichen müssen.

`serial`

Wann immer an den Zonendaten Änderungen vorgenommen werden, sollte die diese Zahl um einen Zähler hochgesetzt werden. Hier ist es meist üblich, entweder von 1 hochzuzählen oder das Datum und die Anzahl der Änderungen pro Tag in diesem Format zu codieren: YYYYMMDDNN, also 2001050901 für die erste Änderung am 9. Mai 2001. So sind bis zu 100 Änderungen pro Tag möglich. Die Seriennummer ist nicht einfach eine nette Gedächtnisstütze für den Administrator, wann er die Datei das letzte Mal geändert hat, sondern dienen konkret dem sekundären Server als Anhaltspunkt, ob er seinen Datenbestand aktualisieren muß.

`refresh`

Diese Zeitangabe weist den/die sekundären Server an, wann nach "frischen" Informationen gefragt werden soll.

`retry`

Wenn eine Anfrage an den Master innerhalb der Refresh Periode fehlschlägt, wird nach der hier angegebenen Zeitspanne ein neuer Versuch gestartet.

`expiry`

Schlägt innerhalb der hier angegeben Zeitspanne jeder Versuch fehl, die Daten zu synchronisieren, gibt der Sekundärserver die Weitergabe der Zonendaten auf. Für ihn sind sie damit veraltet. Die Devise ist hier: "Besser keine Daten weitergeben, als veraltetes Material weiterreichen."

Zonенinformationen über das lokale Netz

Mit den Informationen, die in den Zonendateien stehen, die nun vorgestellt werden, übernimmt unser Nameserver auf `nick` die zentrale Verantwortung für das lokale Netz und ist somit die Instanz, die alle internen Rechner verwaltet.

Für die Zuordnung der Rechnernamen zu den zugehörigen IP-Adressen des lokalen Netzes wird eine Datei `/var/named/home.nil.zone` mit folgendem Inhalt angelegt:

```
$TTL 3D
@           1D IN SOA      nick.home.nil. root.home.nil (
                                42           ; serial (d. adams)
                                3H           ; refresh
                                15M          ; retry
                                1W           ; expiry
                                1D )         ; minimum

                                1D IN NS     nick.home.nil.
localhost  1D IN A       127.0.0.1
nick       1D IN A       192.168.17.1
wallace    1D IN A       192.168.17.2
gromit     1D IN A       192.168.17.3
sean       1D IN A       192.168.17.4
```

Nach einem Neustart des `named` Daemons müßte jetzt schon möglich sein, per `nslookup` `rechnername` einen lokalen Host aufzulösen. Damit die Zuordnung aber auch von IP-Adresse nach Namen funktioniert, muß noch eine `192.168.17.zone` Datei mit dem folgenden Inhalt angelegt werden:

```

$TTL 3D
@           1D IN SOA      nick.home.nil.  root.home.nil (
                                42                ; serial (d. adams)
                                3H                ; refresh
                                15M               ; retry
                                1W                ; expiry
                                1D )              ; minimum

                                1D IN NS          nick.home.nil.

1           1D IN PTR      nick.home.nil.
2           1D IN PTR      wallace.home.nil.
3           1D IN PTR      gromit.home.nil.
4           1D IN PTR      sean.home.nil.

```

Diese Datei ist eigentlich eine genaue Umkehrung der `home.nil.zone`-Datei, mit der kleinen Änderung, daß nicht mehr die volle IP-Adresse aufgeführt wird, sondern nur noch die letzte Ziffer, i.e. nur die Bits, die variabel und nicht schon mit der Netzmaske fest gesetzt sind.

Damit diese Informationen jetzt auch perfekt in das Gesamtkunstwerk "Nameserver" passen - und vor allem, damit die hier gespeicherten Informationen nicht nach draußen dringen, müssen einige Einträge in der `/etc/named.conf` angepaßt werden, wie schon unter dem Abschnitt [named.conf](#) beschrieben wurde. Die Gründe für diese Aussperrungsmaßnahmen liegen darin, daß die internen Rechner keine voll gültige und im gesamten Internet einmalige Adresse besitzen und dort nur für Datenchaos sorgen würden und darin, daß die lokale Konfiguration schon aus Sicherheitsgründen niemandem außer den lokalen Administratoren bekannt sein sollte.

Später dann, wenn es um die Zugriffs- und Transferrechte der einzelnen Zonen geht, müssen hier die jeweils sinnvollen Optionen eingetragen werden. Ein Beispiel:

```

zone "localhost" IN {
    type master;
    file "localhost.zone";
    check-names fail;
    allow-update { none; };
};

```

Diese Einstellung "none" bei `allow-update` untersagt es allen Rechnern, egal ob sie aus dem lokalen oder dem Internet kommen, diese Informationen zu aktualisieren. Für die Zoneninformationen über den lokalen Rechner ist dies auch die einzig sinnvolle Einstellung.

Die Informationen über die lokale Konfiguration, wie sie in den Dateien `home.nil.zone` und `192.168.17.zone` stehen, sollen an wirklich niemanden ausserhalb des lokalen Netzwerkes weitergegeben werden. Deshalb sollen weder Zonentransfers (Weitergabe der kompletten Zonendaten, beispielweise an einen Secondary Nameserver) noch Abfragen ("queries") von außerhalb möglich sein. In diesem Beispiel macht es zum Beispiel keinen Sinn, mehr als die lokalen Hosts für Abfragen zuzulassen, Zonentransfers sollten schon gar keine möglich gemacht werden. Also werden `allow-transfer` und `allow-query` mit den entsprechenden Einstellungen gesetzt. Zum Beispiel so:

```

zone "home.nil" IN {
    type master;
    file "home.nil.zone";
    allow-query { localnets; };
    allow-transfer { none; };
    notify no;
};

```

In unserem Fall ist die `allow-transfer` auf `none` gesetzt, da kein sekundärer Nameserver eingerichtet wurde und auch sonst niemand zu Zonentransfers berechtigt sein soll.

Die Absicherung mit `allow-query` ist nicht zwingend nötig, wenn wie in unserem Fall schon in der

named.conf mit listen-on festgelegt wird, daß eh nur solche Anfragen angenommen werden, die aus dem internen Netz kommen. Wird ohne die listen-on-Angabe bei allow-query der Wert "localnets" angegeben, erweitert sich der Kreis derjenigen, die an unseren Nameserver Anfragen richten können. Dann können alle zum Netz des Providers gehörigen Rechner ebenfalls Anfragen stellen. Um seine lokalen Konfigurationsdaten ganz vor fremden Blicken zu schützen, sollte entweder auf allow-query ganz verzichtet werden, oder eine zusätzliche Sicherung mittels listen-on eingebaut werden. Wenn alternativ die selbstdefinierte acl query verwendet werden soll, ist die Zeile:

```
allow-query { localnets; }; durch allow-query { query; }; zu ersetzen.
```

Damit sollten für den "Hausgebrauch" unseres Netzwerks alle Arbeiten an der Konfiguration beendet sein. Jetzt noch zum guten Schluß einen Eintrag in der /etc/rc.config aktivieren und der Nameserver wird automatisch bei jedem Start des Rechners mitgestartet:


```
#  
# start the named (package bind)? You have to configure the named first,  
# before you can start it (man named).  
#  
START_NAMED="yes"
```

Voreingestellt ist hier natürlich "no". Separat wird *named* zum ersten Mal mit

```
rcnamed start
```

gestartet.

Literatur

- Das [DNS-HOWTO](#) von Nicolai Langfeldt, Jamie Norrish und anderen.
- [Linux. Installation, Konfiguration,...](#) von Michael Kofler. Für Anfänger ein richtig wertvolles Nachschlagewerk ;-)
- [DNS and BIND](#) von Paul Albitz und Cricket Liu. Die mittlerweile vierte Auflage behandelt jetzt auch BIND in Versionen größer als 8.2.3. 

Linux auf dem Server 15.05.2001