DNS HOWTO

Table of Contents

DNS HOWTO **1**
Nicolai Langfeldt janl@math.uio.no 1
1.Preamble 1
2.Introduction 1
3.A caching only name server 1
4.A simple domain 1
5.A real domain example 2
6.Maintenance 2
7.Converting from version 4 to version 8 2
8.Questions and Answers 2
9.How to become a bigger time DNS admin 2
1.Preamble 2
1.1 Legal stuff 2
1.2 Credits and request for help 2
1.3 Dedication 3
2.Introduction 3
3.A caching only name server 4
3.1 Starting named 7
3.2 Making it even better 9
3.3 Congratulations 9
4.A simple domain 9
4.1 But first some dry theory 9
4.2 Our own domain 13
4.3 The reverse zone 20
4.4 Words of caution 22
4.5 Why reverse lookups don't work 22
The reverse zone isn't delegated 22
You've got a classless subnet 23
5.A real domain example 23
5.1 /etc/named.conf (or /var/named/named.conf) 24
5.2 /var/named/root.hints 24
5.3 /var/named/zone/127.0.0 25
5.4 /var/named/zone/land-5.com 26
5.5 /var/named/zone/206.6.177 27
6.Maintenance 28
7.Converting from version 4 to version 8 30
8.Questions and Answers 31
9.How to become a bigger time DNS admin 34

DNS HOWTO

Nicolai Langfeldt `janl@math.uio.no`

v2.2, 11 February 1999

HOWTO become a totally small time DNS admin.

1.Preamble

- 1.1 Legal stuff
- 1.2 Credits and request for help.
- 1.3 Dedication

2.Introduction.

3.A caching only name server.

- 3.1 Starting named
- 3.2 Making it even better
- 3.3 Congratulations

4.A *simple* domain.

- 4.1 But first some dry theory
- 4.2 Our own domain
- 4.3 The reverse zone
- 4.4 Words of caution
- 4.5 Why reverse lookups don't work.

5.A real domain example

- 5.1 /etc/named.conf (or /var/named/named.conf)
- 5.2 /var/named/root.hints
- 5.3 /var/named/zone/127.0.0
- 5.4 /var/named/zone/land−5.com
- 5.5 /var/named/zone/206.6.177

6.Maintenance

7.Converting from version 4 to version 8

8.Questions and Answers

9.How to become a bigger time DNS admin.

1.Preamble

Keywords: DNS, bind, bind−4, bind−8, named, dialup, ppp, slip, isdn, Internet, domain, name, hosts, resolving, caching.

This document is part of the Linux Documentation Project.

1.1 Legal stuff

(C)opyright 1995−1999 Nicolai Langfeldt. Do not modify without amending copyright, distribute freely but retain copyright message.

1.2 Credits and request for help.

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.

This will never be a finished document, please send me mail about your problems and successes, it can make this a better HOWTO. So please send comments and/or questions or money to janl@math.uio.no. If you send e−mail and want an answer please show the simple courtesy of *making sure* that the return address is correct and working. Also, **please** read the QnA section before mailing me. Another thing, I can only understand Norwegian and English.

If you want to translate this HOWTO please notify me so I can keep track of what languages it has been published in, and also I can notify you when the HOWTO has been updated.

1.3 Dedication

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.Introduction.

What this is and isn't.

DNS is is the Domain Name System. DNS converts machine names to the IP addresses that all machines on the net have. It maps from name to address and from address to name, and some other things. This HOWTO documents how to define such mappings using a Linux system. A mapping is simply a association between two things, in this case a machine name, like `ftp.linux.org`, and the machines IP number (or address) `199.249.150.4`.

DNS is, to the uninitiated (you ;−), one of the more opaque areas of network administration. This HOWTO will try to make a few things clearer. It describes how to set up a *simple* DNS name server. Starting with a caching only server and going on to setting up a primary DNS server for a domain. For more complex setups you can check the QnA section of this document. If it's not described there you will need to *read* the Real Documentation. I'll get back to what this Real Documentation consists of in the last chapter.

Before you start on this you should configure your machine so that you can telnet in and out of it, and successfully make all kinds of connections to the net, and you should especially be able to do `telnet 127.0.0.1` and get your own machine (test it now!). You also need a good `/etc/nsswitch.conf` (or `/etc/host.conf`), `/etc/resolv.conf` and `/etc/hosts` files as a starting point, since I will not explain their function here. If you don't already have all this set up and working the NET−3−HOWTO and/or the PPP−HOWTO explains how to set it up. Read them.

When I say `your machine' I mean the machine you are trying to set up DNS on. Not any other machine you might have that's involved in your networking effort.

I assume you're not behind any kind of firewall that blocks name queries. If you are you will need a special configuration, see the section on QnA.

Name serving on Unix is done by a program called `named`. This is a part of the ``bind'' package which is coordinated by Paul Vixie for The Internet Software Consortium. `Named` is included in most Linux distributions and is usually installed as `/usr/sbin/named`. If you have a named you can probably use it; if you don't have one you can get a binary off a Linux ftp site, or get the latest and greatest source from ftp.isc.org:/isc/bind/src/cur/bind−8/. This HOWTO is about bind version 8. The old version of the HOWTO, about bind 4 is still available at http://www.math.uio.no/~janl/DNS/ in case you use bind 4. If the named man page talks about (at the very end, the FILES section) `named.conf` you have bind 8, if it talks about `named.boot` you have bind 4. If you have 4 and are security conscious you really ought to upgrade to a recent 8.

DNS is a net−wide database. Take care about what you put into it. If you put junk into it, you, and others will get junk out of it. Keep your DNS tidy and consistent and you will get good service from it. Learn to use it, admin it, debug it and you will be another good admin keeping the net from falling to it's knees by mismanagement.

In this document I state flatly a couple of things that are not completely true (they are at least half truths though). All in the interest of simplification. Things will (probably ;−) work if you believe what I say.

Tip: Make backup copies of all the files I instruct you to change if you already have them, so if after going through this nothing works you can get it back to your old, working state.

3.A caching only name server.

A first stab at DNS config, very useful for dialup users.

A caching only name server will find the answer to name queries and remember the answer the next time you need it. This will shorten the waiting time the next time significantly, especially if you're on a slow connection.

First you need a file called `/etc/named.conf`. This is read when named starts. For now it should simply contain:

```
// Config file for caching only name server

options {
        directory "/var/named";

        // Uncommenting this might help if you have to go through a
        // firewall and things are not working out:
```

```
        // query-source port 53;
};

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

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

The `directory' line tells named where to look for files. All files named subsequently will be relative to this. Thus pz is a directory under /var/named, i.e., /var/named/pz. /var/named is the right directory according to the *Linux File system Standard*.

The file named /var/named/root.hints is named in this. /var/named/root.hints should contain this:

```
;
; There might be opening comments here if you already have this file.
; If not don't worry.
;
.                       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
```

The file describes the root name servers in the world. This changes over time and must be maintained. See the

maintenance section for how to keep it up to date.

The next section in `named.conf` is the last `zone`. I will explain its use in a later chapter, for now just make this a file named `127.0.0` in the subdirectory `pz`:

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

Next, you need a `/etc/resolv.conf` looking something like this:

```
search subdomain.your-domain.edu your-domain.edu
nameserver 127.0.0.1
```

The `search' line specifies what domains should be searched for any host names you want to connect to. The `nameserver' line specifies the address of your nameserver, in this case your own machine since that is where your named runs (127.0.0.1 is right, no matter if your machine has an other address too). If you want to list several name servers put in one `nameserver' line for each. (Note: Named never reads this file, the resolver that uses named does.)

To illustrate what this file does: If a client tries to look up `foo`, then `foo.subdomain.your-domain.edu` is tried first, then `foo.your-fomain.edu`, finally `foo`. If a client tries to look up `sunsite.unc.edu`, `sunsite.unc.edu.subdomain.your-domain.edu` is tried first (yes, it's silly, but that's the way it works), then `sunsite.unc.edu.your-domain.edu`, and finally `sunsite.unc.edu`. You may not want to put in too many domains in the search line, it takes time to search them all.

The example assumes you belong in the domain `subdomain.your-domain.edu`, your machine then, is probably called `your-machine.subdomain.your-domain.edu`. The search line should not contain your TLD (Top Level Domain, `edu' in this case). If you frequently need to connect to hosts in another domain you can add that domain to the search line like this:

```
search subdomain.your-domain.edu your-domain.edu other-domain.com
```

and so on. Obviously you need to put real domain names in instead. Please note the lack of periods at the end of the domain names. This is important, please note the lack of periods at the end of the domain names.

Next, depending on your libc version you either need to fix `/etc/nsswitch.conf` or

`/etc/host.conf`. If you already have `nsswitch.conf` that's what we'll fix, if not, we'll fix `host.conf`.

/etc/nsswitch.conf

This is a long file specifying where to get different kinds of data types, from what file or database. It usually contains helpful comments at the top, which you should consider reading. After that find the line starting with `` `hosts:' ``, it should read

```
hosts:      files dns
```

If there is no line starting with `` `hosts:' `` then put in the one above. It says that programs should first look in the `/etc/hosts` file, then check DNS according to `resolv.conf`.

/etc/host.conf

It probably contains several lines, one should start with `order` and it should look like this:

```
order hosts,bind
```

If there is no `` `order' `` line you should add one. It tells the host name resolving routines to first look in `/etc/hosts`, then ask the name server (which you in `resolv.conf` said is at 127.0.0.1).

3.1 Starting named

After all this it's time to start named. If you're using a dialup connection connect first. Type `` `ndc start' ``, and press return, no options. If that does not work try `` `/usr/sbin/ndc start' `` instead. If that back–fires see the QnA section. If you view your syslog message file (usually called `/var/adm/messages`, but another directory to look in is `/var/log` and another file to look in is `syslog`) while starting named (do `tail -f /var/log/messages`) you should see something like:

(the lines ending in \ continue on the next line)

```
Feb 15 01:26:17 roke named[6091]: starting.  named 8.1.1 Sat Feb 14 \
  00:18:20 MET 1998 ^Ijanl@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.
```

If there are any messages about errors then there is a mistake. Named will name the file it is in (one of named.conf and root.hints I hope :–) Kill named and go back and check the file.

Now you can test your setup. Start nslookup to examine your work.

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

>
```

If that's what you get it's working. We hope. Anything else, go back and check everything. Each time you change the `named.conf` file you need to restart named using the `ndc restart` command.

Now you can enter a query. Try looking up some machine close to you. `pat.uio.no` is close to me, at the University of Oslo:

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

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

nslookup now asked your named to look for the machine `pat.uio.no`. It then contacted one of the name server machines named in your `root.hints` file, and asked its way from there. It might take tiny while before you get the result as it may need to search all the domains you named in `/etc/resolv.conf`.

If you ask the same again you get this:

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

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

Note the ``Non-authoritative answer:'' line we got this time around. That means that named did not go out on the network to ask this time, the information is in the cache now. But the cached information *might* be out of date (stale). So you are informed of this (very slight) possibility by it saying `Non-authorative answer:'. When `nslookup` says this the second time you ask for a host it's a sure sign that named caches the information and that it's working. You exit `nslookup` by giving the command `exit'.

3.2 Making it even better

In large, well organized, academic or ISP (Internet Service Provider) networks you will sometimes find that the network people has set up a forwarder hierarchy of DNS servers which helps lighten the internal network load and on the outside servers as well. It's not easy to know if you're inside such a network or not. It is however not important and by using the DNS server of your network provider as a ``forwarder'' you can make the responses to queries faster and less of a load on your network. If you use a modem this can be quite a win. For the sake of this example we assume that your network provider has two name servers they want you to use, with IP numbers `10.0.0.1` and `10.1.0.1`. Then, in your `named.conf` file, inside the opening section called ``options'' insert these lines:

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

Restart your nameserver and test it with nslookup. Should work fine.

3.3 Congratulations

Now you know how to set up a caching named. Take a beer, milk, or whatever you prefer to celebrate it.

4.A *simple* domain.

How to set up your own domain.

4.1 But first some dry theory

Before we *really* start this section I'm going to serve you some theory on and an example of how DNS works. And you're going to read it because it's good for you. If you don't want to you should at least skim it very quickly. Stop skimming when you get to what should go in your `named.conf` file.

DNS is a hierarchical, tree structured, system. The top is written `.' and pronounced `root'. Under . there are a number of Top Level Domains (TLDs), the best known ones are ORG, COM, EDU and NET, but there are many more. Just like a tree it has a root and it branches out. If you have any computer science background you will recognize DNS as a search tree, and you will be able to find nodes, leaf nodes and edges.

When looking for a machine the query proceeds recursively into the hierarchy starting at the top. If you want to find out the address of prep.ai.mit.edu your name server has to find a name server that serves edu. It asks a . server (it already knows the . servers, that's what the root.hints file is for), the . server gives a list of edu servers:

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

Start asking a root server:

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

Set the Query type to NS (name server records):

```
> set q=ns
```

Ask about edu:

```
> edu.
```

The trailing . here is significant, it tells nslookup we're asking that edu is right under . (and not under any of our search domains, it speeds the search).

```
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
```

This tells us that all `ROOT-SERVERS.NET` servers serves `EDU.`, so we can go on asking any of them. We'll continue asking `C`. Now we want to know who serves the next level of the domain name: `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
```

`steawb`, `w20ns` and `bitsy` all serves `mit.edu`, we select one and inquire about the name one more level up: `ai.mit.edu`:

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

Host names are not case sensitive, but I use my mouse to cut and paste so it gets copied as–is from the screen.

```
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
```

So `museli.ai.mit.edu` is a nameserver for `ai.mit.edu`:

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

Now I change query type, we've found the name server so now we're going to ask about everything `wheaties` knows about `prep.ai.mit.edu`.

```
> 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
```

So starting at `.` we found the successive name servers for the each level in the domain name. If you had used your own DNS server instead of using all those other servers, your named would of–course cache all the information it found while digging this out for you, and it would not have to ask again for a while.

In the tree analogue each ``.'' in the name is a branching point. And each part between the ``.''s are the names of individual branches in the tree.

We climb the tree by taking the name we want (`prep.ai.mit.edu`) first finding the root (`.`) and then looking for the next branch to climb, in this case `edu`. Once we have found it we climb it by switching to the server that knows about that part of the name. Next we look for the `mit` branch over the `edu` branch (the combined name is `mit.edu`) and climb it by switching to a server that knows about `mit.edu`. Again we look for the next branch, it's `ai.mit.edu` and again we switch to the server that knows about it. Now we have arrived at the right server, at the right branching point. The last part is finding `prep.ai.mit.edu`, which is simple. In computer science we usually call `prep` a *leaf* on the tree.

A much less talked about, but just as important domain is `in-addr.arpa`. It too is nested like the `normal' domains. `in-addr.arpa` allows us to get the hosts name when we have its address. A important thing here is to note that ip addresses are written in reverse order in the `in-addr.arpa` domain. If you have the address of a machine: `192.128.52.43` named proceeds just like for the `prep.ai.mit.edu` example: find `arpa.` servers. Find `in-addr.arpa.` servers, find `192.in-addr.arpa.` servers, find `128.192.in-addr.arpa.` servers, `find 52.128.192.in-addr.arpa.` servers. Find needed records for `43.52.128.192.in-addr.arpa.` Clever huh? (Say `yes'.) The reversion of the numbers can be confusing for years though.

I have just told a lie. DNS does not work precisely the way I just told you. But it's close enough.

4.2 Our own domain

Now to define our own domain. We're going to make the domain `linux.bogus` and define machines in it. I use a totally bogus domain name to make sure we disturb no–one Out There.

One more thing before we start: Not all characters are allowed in host names. We're restricted to the

characters of the English alphabet: a–z, and numbers: 0–9 and the character '–' (dash). Keep to those characters. Upper and lower–case characters are the same for DNS, so `pat.uio.no` is identical to `Pat.UiO.No`.

We've already started this part with this line in `named.conf`:

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

Please note the lack of `.' at the end of the domain names in this file. This says that now we will define the zone `0.0.127.in-addr.arpa`, that we're the master server for it and that it is stored in a file called `pz/127.0.0`. We've already set up this file, it reads:

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

Please note the `.' at the end of all the full domain names in this file, in contrast to the `named.conf` file above. Some people like to start each zone file with a `$ORIGIN` directive, but this is superfluous. The origin (where in the DNS hierarchy it belongs) of a zone file is specified in the zone section of the `named.conf` file, in this case it's `0.0.127.in-addr.arpa`.

This `zone file' contains 3 `resource records' (RRs): A SOA RR. A NS RR and a PTR RR. SOA is short for Start Of Authority. The `@' is a special notation meaning the origin, and since the `domain' column for this file says 0.0.127.in–addr.arpa the first line really means

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

NS is the Name Server RR. There is no '@' at the start of this line, it is implicit since the last line started with a '@'. Saves some typing that. So the NS line could also be written

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

It tells DNS what machine is the name server of the domain `0.0.127.in-addr.arpa`, it is `ns.linux.bogus`. 'ns' is a customary name for name-servers, but as with web servers who are customarily named `www.`*something* the name may be anything.

And finally the PTR record says that the host at address 1 in the subnet `0.0.127.in-addr.arpa`, i.e., 127.0.0.1 is named `localhost`.

The SOA record is the preamble to *all* zone files, and there should be exactly one in each zone file. It describes the zone, where it comes from (a machine called `ns.linux.bogus`), who is responsible for its contents (`hostmaster@linux.bogus`, you should insert your e-mail address here), what version of the zone file this is (serial: 1), and other things having to do with caching and secondary DNS servers. For the rest of the fields (refresh, retry, expire and minimum) use the numbers used in this HOWTO and you should be safe.

Now restart your named (the command is `ndc restart`) and use nslookup to examine what you've done:

```
$ 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
```

so it manages to get `localhost` from 127.0.0.1, good. Now for our main task, the `linux.bogus` domain, insert a new 'zone' section in `named.conf`:

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

Note again the lack of ending `.' on the domain name in the `named.conf` file.

In the `linux.bogus` zone file we'll put some totally bogus data:

```
;
; Zone file for linux.bogus
;
; The full zone file
```

```
;
@       IN      SOA     ns.linux.bogus. hostmaster.linux.bogus. (
                        199802151       ; serial, todays date + todays serial #
                        8H              ; refresh, seconds
                        2H              ; retry, seconds
                        1W              ; expire, seconds
                        1D )            ; minimum, seconds
;
                NS      ns              ; Inet Address of name server
                MX      10 mail.linux.bogus     ; Primary Mail Exchanger
                MX      20 mail.friend.bogus.   ; Secondary Mail Exchanger
;
localhost       A       127.0.0.1
ns              A       192.168.196.2
mail            A       192.168.196.4
```

Two things must be noted about the SOA record. `ns.linux.bogus`*must* be a actual machine with a A record. It is not legal to have a CNAME record for he machine mentioned in the SOA record. It's name need not be `ns', it could be any legal host name. Next, hostmaster.linux.bogus should be read as hostmaster@linux.bogus, this should be a mail alias, or a mailbox, where the person(s) maintaining DNS should read mail frequently. Any mail regarding the domain will be sent to the address listed here. The name need not be `hostmaster', it can be your normal e−mail address, but the e−mail address `hostmaster' is often expected to work as well.

There is one new RR type in this file, the MX, or Mail eXchanger RR. It tells mail systems where to send mail that is addressed to `someone@linux.bogus`, namely too `mail.linux.bogus` or `mail.friend.bogus`. The number before each machine name is that MX RRs priority. The RR with the lowest number (10) is the one mail should be sent to if possible. If that fails the mail can be sent to one with a higher number, a secondary mail handler, i.e., `mail.friend.bogus` which has priority 20 here.

Restart named by running `ndc restart`. Examine the results with nslookup:

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

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

Upon careful examination you will discover a bug. The line

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

is all wrong. It should be

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

I deliberately made a mistake so you could learn from it :–) Looking in the zone file we find that the line

```
                MX      10 mail.linux.bogus     ; Primary Mail Exchanger
```

is missing a period. Or has a 'linux.bogus' too many. If a machine name does not end in a period in a zone file the origin is added to its end causing the double `linux.bogus.linux.bogus`. So either

```
                MX      10 mail.linux.bogus.    ; Primary Mail Exchanger
```

or

```
                MX      10 mail                 ; Primary Mail Exchanger
```

is correct. I prefer the latter form, it's less to type. There are some bind experts that disagree, and some that agree with this. In a zone file the domain should either be written out and ended with a `.' or it should not be included at all, in which case it defaults to the origin.

I must stress that in the named.conf file there should *not* be `.'s after the domain names. You have no idea how many times a `.' too many or few have fouled up things and confused the h*ll out of people.

So having made my point here is the new zone file, with some extra information in it as well:

```
;
; Zone file for linux.bogus
;
; The full zone file
;
@       IN      SOA     ns.linux.bogus. hostmaster.linux.bogus. (
                        199802151       ; serial, todays date + todays serial #
                        8H              ; refresh, seconds
```

```
                        2H              ; retry, seconds
                        1W              ; expire, seconds
                        1D )            ; minimum, seconds
;
                TXT     "Linux.Bogus, your DNS consultants"
                NS      ns              ; Inet Address of name server
                NS      ns.friend.bogus.
                MX      10 mail         ; Primary Mail Exchanger
                MX      20 mail.friend.bogus. ; Secondary Mail Exchanger

localhost       A       127.0.0.1

gw              A       192.168.196.1
                HINFO   "Cisco" "IOS"
                TXT     "The router"

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

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

mail            A       192.168.196.4
                MX      10 mail
                MX      20 mail.friend.bogus.
                HINFO   "386sx" "Linux 1.2"

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

There are a number of new RRs here: HINFO (Host INFOrmation) has two parts, it's a good habit to quote each. The first part is the hardware or CPU on the machine, and the second part the software or OS on the machine. The machine called 'ns' has a Pentium CPU and runs Linux 2.0. CNAME (Canonical NAME) is a way to give each machine several names. So www is an alias for ns.

CNAME record usage is a bit controversial. But it's safe to follow the rule that a MX, CNAME or SOA record should *never* refer to a CNAME record, they should only refer to something with a A record, so it is inadvisable to have

```
foobar          CNAME   www                     ; NO!
```

but correct to have

```
foobar          CNAME   ns                      ; Yes!
```

It's also safe to assume that a CNAME is not a legal host name for a e−mail address: `webmaster@www.linux.bogus` is an illegal e−mail address given the setup above. You can expect quite a few mail admins Out There to enforce this rule even if it works for you. The way to avoid this is to use A records (and perhaps some others too, like a MX record) instead:

```
www             A       192.168.196.2
```

A number of the arch−bind−wizards, recommend *not* using CNAME at all. But the discussion of why or why not is beyond this HOWTO.

But as you see, this HOWTO and many sites does not follow this rule.

Load the new database by running `ndc reload`, this causes named to read its files again.

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

> ls −d linux.bogus
```

This means that all records should be listed. It results in this:

```
[localhost]
$ORIGIN linux.bogus.
@                       1D IN SOA       ns hostmaster (
                                        199802151       ; serial
                                        8H              ; refresh
                                        2H              ; retry
                                        1W              ; expiry
                                        1D )            ; minimum

                        1D IN NS        ns
                        1D IN NS        ns.friend.bogus.
                        1D IN TXT       "Linux.Bogus, your DNS consultants"
                        1D IN MX        10 mail
                        1D IN MX        20 mail.friend.bogus.
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.bogus.
                        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.bogus.
                        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.bogus.
                        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.bogus.
                        1D IN HINFO     "Pentium" "Linux 1.2"
```

That's good. As you see it looks a lot like the zone file itself. Let's check what it says for `www` alone:

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

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

In other words, the real name of `www.linux.bogus` is `ns.linux.bogus`, and it gives you some of the information it has about ns as well, enough to connect to it if you were a program.

Now we're halfway.

4.3 The reverse zone

Now programs can convert the names in linux.bogus to addresses which they can connect to. But also required is a reverse zone, one making DNS able to convert from an address to a name. This name is used buy a lot of servers of different kinds (FTP, IRC, WWW and others) to decide if they want to talk to you or not, and if so, maybe even how much priority you should be given. For full access to all services on the Internet a reverse zone is required.

Put this in `named.conf`:

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

```
};
```

This is exactly as with the `0.0.127.in-addr.arpa`, and the contents are similar:

```
@       IN      SOA     ns.linux.bogus. hostmaster.linux.bogus. (
                        199802151 ; Serial, todays date + todays serial
                        8H      ; Refresh
                        2H      ; Retry
                        1W      ; Expire
                        1D)     ; Minimum TTL
                NS      ns.linux.bogus.

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

Now you restart your named (`ndc restart`) and examine your work with nslookup again:

```
> 192.168.196.4
Server:  localhost
Address:  127.0.0.1

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

so, it looks OK, dump the whole thing to examine that too:

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

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

Looks good! If your output didn't look like that look for error−messages in your syslog, I explained how to do that at the very beginning of this chapter.

4.4 Words of caution

There are some things I should add here. The IP numbers used in the examples above are taken from one of the blocks of 'private nets', i.e., they are not allowed to be used publicly on the internet. So they are safe to use in an example in a HOWTO. The second thing is the `notify no;` line. It tells named not to notify its secondary (slave) servers when it has gotten a update to one of its zone files. In bind−8 the named can notify the other servers listed in NS records in the zone file when a zone is updated. This is handy for ordinary use, but for private experiments with zones this feature should be off, we don't want the experiment to pollute the Internet do we?

And, of course, this domain is highly bogus, and so are all the addresses in it. For a real example of a real−life domain see the next main−section.

4.5 Why reverse lookups don't work.

There are a couple of ``gotchas'' that normally are avoided with name lookups that are often seen when setting up reverse zones. Before you go on you need reverse lookups of your machines working on your own nameserver. If it isn't go back and fix it before continuing.

I will discuss two failures of reverse lookups as seen from outside your network:

The reverse zone isn't delegated.

When you ask a service provider for a network−address range and a domain name the domain name is normally delegated as a matter of course. A delegation is the glue NS record that helps you get from one nameserver to another as explained in the dry theory section above. You read that, right? If your reverse zone dosn't work go back and read it. Now.

The reverse zone also needs to be delegated. If you got the `192.168.196` net with the `linux.bogus` domain from your provider they need to put `NS` records in for your reverse zone as well as for your forward zone. If you follow the chain from `in−addr.arpa` and up to your net you will probably find a break in the chain. Most probably at your service provider. Having found the break in the chain contact your service−provider and ask them to correct the error.

You've got a classless subnet

This is a somewhat advanced topic, but classless subnets are very common these days and you probably have one unless you're a medium sized company.

A classless subnet is what keeps the Internet going these days. Some years ago there was much ado about the shortage of ip numbers. The smart people in IETF (the Internet Engineering Task Force, they keep the Internet working) stuck their heads together and solved the problem. At a price. The price is that you'll get less than a ``C'' subnet and some things may break. Please see Ask Mr. DNS at http://www.acmebw.com/askmrdns/00007.htm for an good explanation of this and how to handle it.

Did you read it? I'm not going to explain it so please read it.

The first part of the problem is that your ISP must understand the technique described by Mr. DNS. Not all small ISPs have a working understanding of this. If so you might have to explain to them and be persistent. But be sure you understand it first ;–). They will then set up a nice reverse zone at their server which you can examine for correctness with nslookup.

The second and last part of the problem is that you must understand the technique. If you're unsure go back and read about it again. Then you can set up your own classless reverse zone as described by Mr. DNS.

There is another trap lurking here. Old resolvers will *not* be able to follow the `CNAME` trick in the resolving chain and will fail to reverse–resolve your machine. This can result in the service assigning it an incorrect access class, deny access or something along those lines. If you stumble into such a service the only solution (that I know of) is for your ISP to insert your PTR record directly into their trick classless zone file instead of the trick CNAME record.

Some ISPs will offer other ways to handle this, like Web based forms for you to input your reverse–mappings in or other automagical systems.

5.A real domain example

Where we list some *real* zone files

Users have suggested that I include a real example of a working domain as well as the tutorial example.

I use this example with permission from David Bullock of LAND–5. These files were current 24th of September 1996, and were then edited to fit bind 8 restrictions and use extensions by me. So, what you see here differs a bit from what you find if you query LAND–5's name servers now.

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

Here we find master zone sections for the two reverse zones needed: the 127.0.0 net, as well as LAND-5's `206.6.177` subnet. And a primary line for land-5's forward zone `land-5.com`. Also note that instead of stuffing the files in a directory called `pz`, as I do in this HOWTO, he puts them in a directory called `zone`.

```
// Boot file for LAND-5 name server

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";
};
```

If you put this in your named.conf file to play with **PLEASE** put ``notify no;'' in the zone sections for the two `land-5` zones so as to avoid accidents.

5.2 /var/named/root.hints

Keep in mind that this file is dynamic, and the one listed here is old. You're better off using one produced now, with dig, as explained earlier.

```
; <<>> 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

Just the basics, the obligatory SOA record, and a record that maps 127.0.0.1 to `localhost`. Both are required. No more should be in this file. It will probably never need to be updated, unless your nameserver or hostmaster address changes.

```
@               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

Here we see the mandatory SOA record, the needed NS records. We can see that he has a secondary name server at `ns2.psi.net`. This is as it should be, *always* have a off site secondary server as backup. We can also see that he has a master host called `land-5` which takes care of many of the different Internet services, and that he's done it with CNAMEs (a alternative is using A records).

As you see from the SOA record, the zone file originates at `land-5.com`, the contact person is `root@land-5.com`. `hostmaster` is another oft used address for the contact person. The serial number is in the customary yyyymmdd format with todays serial number appended; this is probably the sixth version of zone file on the 20th of September 1996. Remember that the serial number *must* increase monotonically, here there is only *one* digit for todays serial#, so after 9 edits he has to wait until tomorrow before he can edit the file again. Consider using two digits.

```
@       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
```

If you examine land–5s nameserver you will find that the host names are of the form `ws_`*number*. As of late bind 4 versions named started enforcing the restrictions on what characters may be used in host names. So that does not work with bind–8 at all, and I substituted '–' (dash) for '_' (underline) for use in this HOWTO.

Another thing to note is that the workstations don't have individual names, but rather a prefix followed by the two last parts of the IP numbers. Using such a convention can simplify maintenance significantly, but can be a bit impersonal, and, in fact, be a source of irritation among your customers.

We also see that `funn.land-5.com` is an alias for `land-5.com`, but using an A record, not a CNAME record. This is a good policy as noted earlier.

5.5 /var/named/zone/206.6.177

I'll comment on this file below

```
@               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.
```

The reverse zone is the bit of the setup that seems to cause the most grief. It is used to find the host name if you have the IP number of a machine. Example: you are an IRC server and accept connections from IRC clients. However you are a Norwegian IRC server and so you only want to accept connections from clients in Norway and other Scandinavian countries. When you get a connection from a client the C library is able to tell you the IP number of the connecting machine because the IP number of the client is contained in all the packets that are passed over the network. Now you can call a function called gethostbyaddr that looks up the name of a host given the IP number. Gethostbyaddr will ask a DNS server, which will then traverse the DNS looking for the machine. Supposing the client connection is from ws-177200.land-5.com. The IP number the C library provides to the IRC server is 206.6.177.200. To find out the name of that machine we need to find `200.177.6.206.in-addr.arpa`. The DNS server will first find the `arpa.` servers, then find `in-addr.arpa.` servers, following the reverse trail through 206, then 6 and at last finding the server for the `177.6.206.in-addr.arpa` zone at LAND-5. From which it will finally get the answer that for `200.177.6.206.in-addr.arpa` we have a ``PTR ws-177200.land-5.com'' record, meaning that the name that goes with `206.6.177.200` is `ws-177200.land-5.com`. As with the explanation of how `prep.ai.mit.edu` is looked up, this is slightly fictitious.

Getting back to the IRC server example. The IRC server only accepts connections from the Scandinavian countries, i.e., `*.no`, `*.se`, `*.dk`, the name `ws-177200.land-5.com` clearly does not match any of those, and the server will deny the connection. If there was *no* reverse mapping of `206.2.177.200` through the `in-addr.arpa` zone the server would have been unable to find the name at all and would have to settle to comparing `206.2.177.200` with `*.no`, `*.se` and `*.dk`, none of which will match.

Some people will tell you that reverse lookup mappings are only important for servers, or not important at all. Not so: Many ftp, news, IRC and even some http (WWW) servers will *not* accept connections from machines of which they are not able to find the name. So reverse mappings for machines are in fact *mandatory*.

6.Maintenance

Keeping it working.

There is one maintenance task you have to do on nameds, other than keeping them running. That's keeping the `root.hints` file updated. The easiest way is using dig, first run dig with no arguments, you will get the `root.hints` according to your own server. Then ask one of the listed root servers with `dig @rootserver`. You will note that the output looks terribly like a `root.hints` file. Save it to a file (`dig @e.root-servers.net . ns >root.hints.new`) and replace the old `root.hints` with it.

Remember to reload named after replacing the cache file.

Al Longyear sent me this script, that can be run automatically to update `root.hints`, install a crontab entry to run it once a month and forget it. The script assumes you have mail working and that the mail−alias `hostmaster' is defined. You must hack it to suit your setup.

```
#!/bin/sh
#
# Update the nameserver cache information file once per month.
# This is run automatically by a cron entry.
#
# Original by Al Longyear
# Updated for bind 8 by Nicolai Langfeldt
# Miscelanious error-conditions reported by David A. Ranch
# Ping test suggested by Martin Foster
#
(
 echo "To: hostmaster <hostmaster>"
 echo "From: system <root>"
 echo "Subject: Automatic update of the root.hints file"
 echo

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

 # Are we online?  Ping a server at your ISP
 case `ping -qnc some.machine.net` in
   *'100% packet loss'*)
        echo "The network is DOWN. root.hints NOT updated"
        echo
        exit 0
        ;;
 esac

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

 case `cat root.hints.new` in
   *NOERROR*)
        # It worked
        :;;
   *)
        echo "The root.hints file update has FAILED."
        echo "This is the dig output reported:"
        echo
        cat root.hints.new
        exit 0
        ;;
 esac

 echo "The root.hints file has been updated to contain the following
information:"
 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 "The nameserver has been restarted to ensure that the update is complete."
 echo "The previous root.hints file is now called
/var/named/root.hints.old."
) 2>&1 | /usr/lib/sendmail -t
exit 0
```

Some of you might have picked up that the `root.hints` file is also available by ftp from Internic. Please don't use ftp to update `root.hints`, the above method is much more friendly to the net, and Internic.

7.Converting from version 4 to version 8

This was originally a section on using bind 8 written by David E. Smith (dave@bureau42.ml.org). I have edited it some to fit the new section name.

There's not much to it. Except for using named.conf instead of named.boot, everything is identical. And bind8 comes with a perl script that converts old-style files to new. Example named.boot (old style) for a cache-only name server:

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

On the command line, in the bind8/src/bin/named directory (*this assumes you got a source distribution. If you got a binary package the script is probably around, I'm not sure where it would be though. -ed.*), type:

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

Which creates named.conf:

```
// 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";
};
```

It works for everything that can go into a `named.boot` file, although it doesn't add all of the new enhancements and configuration options that bind8 allows. Here's a more complete `named.conf` that does the same things, but a little more efficiently.

```
// This is a configuration file for named (from BIND 8.1 or later).
// It would normally be installed as /etc/named.conf.
// The only change made from the `stock' named.conf (aside from this
// comment :) is that the directory line was uncommented, since I
// already had the zone files in /var/named.

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 the bind 8 distributions directory bind8/src/bin/named/test you find this, and copies of the zone files, that many people can just drop in and use instantly.

The formats for zone files and `root.hints` files are identical, as are the commands for updating them.

8.Questions and Answers

Please read this section before mailing me.

1. My named wants a named.boot file

   You are reading the wrong HOWTO. Please see the old version of this HOWTO, which covers bind

4, at http://www.math.uio.no/~janl/DNS/

2. How do use DNS from inside a firewall?

   A hint: `forward only;`, You will probably also need

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

   inside the ``options'' part of the `named.conf` file as suggested in the example caching section.

3. How do I make DNS rotate through the available addresses for a service, say www.busy.site to obtain a load balancing effect, or similar?

   Make several **A** records for www.busy.site and use bind 4.9.3 or later. Then bind will round–robin the answers. It will *not* work with earlier versions of bind.

4. I want to set up DNS on a (closed) intranet. What do I do?

   You drop the root.hints file and just do zone files. That also means you don't have to get new hint files all the time.

5. How do I set up a secondary (slave) name server?

   If the primary/master server has address 127.0.0.1 you put a line like this in the named.conf file of your secondary:

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

   You may list several alternate master servers the zone can be copied from inside the `masters` list, separated by ';' (semicolon).

6. I want bind running when I'm disconnected from the net.

   There are three items regarding this:

   - I have received this mail from Ian Clark <ic@deakin.edu.au> where he explains his way of doing this:

```
I run named on my 'Masquerading' machine here. I have
two root.hints files, one called root.hints.real which contains
the real root server names and the other called root.hints.fake
which contains...
```

```
----
; root.hints.fake
; this file contains no information
----

When I go off line I copy the root.hints.fake file to root.hints and
restart named.

When I go online I copy root.hints.real to root.hints and restart
named.

This is done from ip-down & ip-up respectively.

The first time I do a query off line on a domain name named doesn't
have details for it puts an entry like this in messages..

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

which I can live with.

It certainly seems to work for me. I can use the nameserver for
local machines while off the 'net without the timeout delay for
external domain names and I while on the 'net queries for external
domains work normally
```

- I have also received information about how bind interacts with NFS and the portmapper on a mostly offline machine from Karl-Max Wanger:

```
I use to run my own named on all my machines which are only
occasionally connected to the Internet by modem. The nameserver only
acts as a cache, it has no area of authority and asks back for
everything at the name servers in the root.cache file. As is usual with
Slackware, it is started before nfsd and mountd.

With one of my machines (a Libretto 30 notebook) I had the problem
that sometimes I could mount it from another system connected to my
local LAN, but most of the time it didn't work.  I had the same effect
regardless of using PLIP, a PCMCIA ethernet card or PPP over a serial
interface.

After some time of guessing and experimenting I found out that
apparently named messed with the process of registration nfsd and
mountd have to carry out with the portmapper upon startup (I start
these daemons at boot time as usual). Starting named after nfsd and
mountd eliminated this problem completely.

As there are no disadvantages to expect from such a modified boot
sequence I'd advise everybody to do it that way to prevent potential
trouble.
```

- Finally, there is HOWTO information about this at Ask Mr. DNS at http://www.acmebw.com/askmrdns/#linux-ns. It is about bind 4 though, so you have to adapt what he says to bind 8.

7. Where does the caching name server store its cache? Is there any way I can control the size of the cache?

The cache is completely stored in memory, it is *not* written to disk at any time. Every time you kill named the cache is lost. The cache is *not* controllable in any way. named manages it according to some simple rules and that is it. You cannot control the cache or the cache size in any way for any reason. If you want to you can ``fix'' this by hacking named. This is however not recommended.

8. Does named save the cache between restarts? Can I make it save it?

   No, named does *not* save the cache when it dies. That means that the cache must be built anew each time you kill and restart named. There is *no* way to make named save the cache in a file. If you want you can ``fix'' this by hacking named. This is however not recommended.

9. How can I get a domian? I want to set up my own domain called (for example) `linux-rules.net`. How can I get the domain I want assigned to me?

   Please contact your network service provider. They will be able to help you with this. Please note that in most parts of the world you need to pay money to get a domain.

9.How to become a bigger time DNS admin.

Documentation and tools.

Real Documentation exists. Online and in print. The reading of several of these is required to make the step from small time DNS admin to a big time one. In print the standard book is *DNS and BIND* by C. Liu and P. Albitz from O'Reilly & Associates, Sebastopol, CA, ISBN 0−937175−82−X. I read this, it's excellent, though based on bind 4, this is not a real problem though. There is also a section in on DNS in *TCP/IP Network Administration*, by Craig Hunt from O'Reilly..., ISBN 0−937175−82−X. Another must for Good DNS administration (or good anything for that matter) is *Zen and the Art of Motorcycle Maintenance* by Robert M. Pirsig :−) Available as ISBN 0688052304 and others.

Online you will find stuff on http://www.dns.net/dnsrd/ (DNS Resources Directory), http://www.isc.org/bind.html; A FAQ, a reference manual (BOG; Bind Operations Guide) as well as papers and protocol definitions and DNS hacks (these, and most, if not all, of the RFCs mentioned below, are also contained in the bind distribution). I have not read most of these, but then I'm not a big−time DNS admin either. Arnt Gulbrandsen on the other hand has read BOG and he's ecstatic about it :−). The newsgroup comp.protocols.tcp−ip.domains is about DNS. In addition there are a number of RFCs about DNS, the most important are probably these:

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*, this is available at
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.