DNS HOWTO Nicolai Langfeldt (

Table of Contents

DNS HOWTO **1**
Nicolai Langfeldt (dns−howto(at)langfeldt.net), Jamie Norrish and others 1
1. Preamble 1
2. Introduction 1
3. A resolving, caching name server 1
4. Forwarding 1
5. A simple domain 1
6. Basic security options 1
7. A real domain example 2
8. Maintenance 2
9. Migrating to BIND 9 2
10. Questions and Answers 2
11. 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
1.4 Updated versions 3
2. Introduction 3
2.1 Other nameserver implementations 4
3. A resolving, caching name server 4
3.1 Starting named 8
3.2 Resolvers 10
3.3 Congratulations 10
4. Forwarding 10
5. A simple domain 11
5.1 But first some dry theory 11
5.2 Our own domain 13
5.3 The reverse zone 19
5.4 Words of caution 20
5.5 Why reverse lookups don't work 20
The reverse zone isn't delegated 21
You've got a classless subnet 21
5.6 Slave servers 21
6. Basic security options 22
6.1 Restricting zone transfers 22
6.2 Protecting against spoofing 23
6.3 Running named as non−root 23
7. A real domain example 24
7.1 /etc/named.conf (or /var/named/named.conf) 24
7.2 /var/named/root.hints 25
7.3 /var/named/zone/127.0.0 26
7.4 /var/named/zone/land−5.com 26
7.5 /var/named/zone/206.6.177 27
8. Maintenance 29
9. Migrating to BIND 9 30
10. Questions and Answers 31
11. How to become a bigger time DNS admin 34

DNS HOWTO

Nicolai Langfeldt (`dns-howto(at)langfeldt.net`), Jamie Norrish and others

v9.0, 2001−12−20

HOWTO become a totally small time DNS admin.

1. Preamble

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

2. Introduction.

- 2.1 Other nameserver implementations.

3. A resolving, caching name server.

- 3.1 Starting named
- 3.2 Resolvers
- 3.3 Congratulations

4. Forwarding

5. A *simple* domain.

- 5.1 But first some dry theory
- 5.2 Our own domain
- 5.3 The reverse zone
- 5.4 Words of caution
- 5.5 Why reverse lookups don't work.
- 5.6 Slave servers

6. Basic security options.

- 6.1 Restricting zone transfers
- 6.2 Protecting against spoofing
- 6.3 Running named as non−root

7. A real domain example

- 7.1 /etc/named.conf (or /var/named/named.conf)
- 7.2 /var/named/root.hints
- 7.3 /var/named/zone/127.0.0
- 7.4 /var/named/zone/land−5.com
- 7.5 /var/named/zone/206.6.177

8. Maintenance

9. Migrating to BIND 9

10. Questions and Answers

11. How to become a bigger time DNS admin.

---

1. Preamble

Keywords: DNS, BIND, BIND 4, BIND 8, BIND 9, named, dialup, PPP, slip, ISDN, Internet, domain, name, resolution, hosts, caching.

This document is part of the Linux Documentation Project.

1.1 Legal stuff

(C)opyright 1995−2001 Nicolai Langfeldt, Jamie Norrish & Co. Do not modify without amending copyright, distribute freely but retain copyright message.

1.2 Credits and request for help.

I want to thank all the people that I have bothered with reading this HOWTO (you know who you are) and all the readers that have e−mailed suggestions and notes.

This will never be a finished document; please send me mail about your problems and successes. You can help make this a better HOWTO. So please send comments and/or questions or money to janl(at)langfeldt.net. Or buy my DNS book (it's titled "The Concise Guide to DNS and BIND, the bibliography has ISBNs). 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 qanda section before mailing me. Another thing, I can only understand Norwegian and English.

This is a HOWTO. I have maintained it as part of the LDP since 1995. I have, during 2000, written a book on the same subject. I want to say that, though this HOWTO is in many ways much like the book it is *not* a watered down version concocted to market the book. The readers of this HOWTO have helped me understand what is difficult to understand about DNS. This has helped the book, but the book has also helped me to think more about what this HOWTO needs. The HOWTO begot the book. The book begot version 3 of this

HOWTO. My thanks to the book publisher, Que, that took a chance on me :−)

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.

1.4 Updated versions

You should be able to find updated versions of this HOWTO both at http://langfeldt.net/DNS−HOWTO/ and on http://www.linuxdoc.org/. Go there if this document is dated more than 9 months ago.

---

2. Introduction.

What this is and isn't.

DNS is the Domain Name System. DNS converts machine names to the IP addresses that all machines on the net have. It translates (or "maps" as the jargon would have it) from name to address and from address to name, and some other things. This HOWTO documents how to define such mappings using Unix system, with a few things specific to Linux.

A mapping is simply an association between two things, in this case a machine name, like `ftp.linux.org`, and the machine's IP number (or address) `199.249.150.4`. DNS also contains mappings the other way, from the IP number to the machine name; this is called a "reverse mapping".

DNS is, to the uninitiated (you ;−), one of the more opaque areas of network administration. Fortunately DNS isn't really that hard. 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 qanda 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 good `/etc/nsswitch.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 Networking−HOWTO and/or the Networking−Overview−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 qanda.

Name serving on Unix is done by a program called `named`. This is a part of the ``BIND" package which is coordinated by *The Internet Software Consortium*. `Named` is included in most Linux distributions and is usually installed as `/usr/sbin/named`, usually from a package called `BIND`, in upper or lower case

depending on the whim of the packager.

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://ftp.isc.org/isc/bind9/. This HOWTO is about BIND version 9. The old versions of the HOWTO, about BIND 4 and 8, is still available at http://langfeldt.net/DNS−HOWTO/ in case you use BIND 4 or 8 (incidentally, you will find this HOWTO there too). If the named man page talks about (at the very end, in 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 the latest version of BIND 8. Now.

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 its knees by mismanagement.

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

2.1 Other nameserver implementations.

This section was written by Joost van Baal.

Various packages exist for getting a DNS server on your box. There is the BIND package (http://www.isc.org/products/BIND/); the implementation this HOWTO is about. It's the most popular nameserver around and it's used on the vast majority of name serving machines on the Internet, around and being deployed since the 1980's. It's available under a BSD license. Since it's the most popular package, loads of documentation and knowledge about BIND is around. However, there have been security problems with BIND.

Then there is djbdns (http://djbdns.org/), a relatively new DNS package written by Daniel J. Bernstein, who also wrote qmail. It's a very modular suite: various small programs take care of the different jobs a nameserver is supposed to handle. It's designed with security in mind. It uses a simpler zone−file format, and is generally easier to configure. However, since it's less well known, your local guru might not be able to help you with this. Unfortunately, this software is not Open Source. The author's advertisement is on http://cr.yp.to/djbdns/ad.html.

Whether DJBs software is really an improvement over the older alternatives is a subject of much debate. A discussion (or is it a flame−war?) of BIND vs djbdns, joined by ISC people, is on http://www.isc.org/ml−archives/bind−users/2000/08/msg01075.html

---

3. A resolving, caching name server.

A first stab at DNS config, very useful for dialup, cable−modem, ADSL and similar users.

On Red Hat and Red Hat related distributions you can achieve the same practical result as this HOWTO's first section by installing the packages `bind`, `bind−utils` and `caching−nameserver`. If you use Debian simply install `bind` (or `bind9`, as of this writing, BIND 9 is not supported by Debian Stable (potato)) and `bind−doc`. Of course just installing those packages won't teach you as much as reading this

HOWTO. So install the packages, and then read along verifying the files they installed.

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` (Debian: `/etc/bind/named.conf`). This is read when named starts. For now it should simply contain:

```
// Config file for caching only name server
//
// The version of the HOWTO you read may contain leading spaces
// (spaces in front of the characters on these lines ) in this and
// other files.  You must remove them for things to work.
//
// Note that the filenames and directory names may differ, the
// ultimate contents of should be quite similar though.

options {
        directory "/var/named";

        // Uncommenting this might help if you have to go through a
        // firewall and things are not working out.  But you probably
        // need to talk to your firewall admin.

        // query-source port 53;
};

controls {
        inet 127.0.0.1 allow { localhost; } keys { rndc_key; };
};

key "rndc_key" {
        algorithm hmac-md5;
        secret "c3Ryb25nIGVub3VnaCBmb3IgYSBtYW4gYnV0IG1hZGUgZm9yIGEgd29tYW4K";
};

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

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

The Linux distribution packages may use different file names for each kind of file mentioned here; they will still contain about the same things.

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.
;
; About any leading spaces in front of the lines here: remove them!
; Lines should start in a ;, . or character, not blanks.
;
.                       6D  IN      NS      A.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      F.ROOT-SERVERS.NET.
.                       6D  IN      NS      G.ROOT-SERVERS.NET.
.                       6D  IN      NS      H.ROOT-SERVERS.NET.
.                       6D  IN      NS      I.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.
A.ROOT-SERVERS.NET.     6D  IN      A       198.41.0.4
B.ROOT-SERVERS.NET.     6D  IN      A       128.9.0.107
C.ROOT-SERVERS.NET.     6D  IN      A       192.33.4.12
D.ROOT-SERVERS.NET.     6D  IN      A       128.8.10.90
E.ROOT-SERVERS.NET.     6D  IN      A       192.203.230.10
F.ROOT-SERVERS.NET.     6D  IN      A       192.5.5.241
G.ROOT-SERVERS.NET.     6D  IN      A       192.112.36.4
H.ROOT-SERVERS.NET.     6D  IN      A       128.63.2.53
I.ROOT-SERVERS.NET.     6D  IN      A       192.36.148.17
J.ROOT-SERVERS.NET.     6D  IN      A       198.41.0.10
K.ROOT-SERVERS.NET.     6D  IN      A       193.0.14.129
L.ROOT-SERVERS.NET.     6D  IN      A       198.32.64.12
M.ROOT-SERVERS.NET.     6D  IN      A       202.12.27.33
```

The file describes the root name servers in the world. The servers change over time and must be maintained now and then. 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`: (*Again, please remove leading spaces if you cut and paste this*)

```
$TTL 3D
@               IN      SOA     ns.linux.bogus. hostmaster.linux.bogus. (
                                1       ; Serial
                                8H      ; Refresh
                                2H      ; Retry
                                4W      ; Expire
                                1D)     ; Minimum TTL
                        NS      ns.linux.bogus.
1                       PTR     localhost.
```

The sections called `key` and `controls` together specify that your named can be remotely controlled by a program called `rndc` if it connects from the local host, and identifis itself with the encoded secret key. This key is like a password. For rndc to work you need `/etc/rndc.conf` to match this:

```
key rndc_key {
    algorithm "hmac-md5";
    secret "c3Ryb25nIGVub3VnaCBmb3IgYSBtYW4gYnV0IG1hZGUgZm9yIGEgd29tYW4K";
};

options {
    default-server localhost;
    default-key    rndc_key;
};
```

As you see the secret is identical. If you want to use `rndc` from other machines their times need to be within 5 minutes of eachother. I recommend using the ntp (`xntpd` and `ntpdate`) software to do this.

Next, you need a `/etc/resolv.conf` looking something like this: (*Again: Remove spaces!*)

```
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 another 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. Note 2: In some resolv.conf files you find a line saying "domain". That's fine, but don't use both "search" and "domain", only one of them will work).

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-domain.edu`, and finally `foo`. You may not want to put in too many domains in the search line, as 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: (*Remember to remove the leading spaces, if any*)

```
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.

3.1 Starting named

After all this it's time to start named. If you're using a dialup connection connect first. Now run named, either by running the boot script: `/etc/init.d/named start` or named directly: `/usr/sbin/named`. If you have tried previous versions of BIND you're probably used to `ndc`. I BIND 9 it has been replaced with `rndc`, which can controll your named remotely, but it can't start named anymore. If you view your syslog message file (usually called `/var/log/messages`, Debian calls it `/var/log/daemon`, another directory to look is the other files `/var/log`) while starting named (do `tail -f /var/log/messages`) you should see something like:

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

```
Dec 23 02:21:12 lookfar named[11031]: starting BIND 9.1.3
Dec 23 02:21:12 lookfar named[11031]: using 1 CPU
Dec 23 02:21:12 lookfar named[11034]: loading configuration from \
    '/etc/named.conf'
Dec 23 02:21:12 lookfar named[11034]: the default for the \
    'auth-nxdomain' option is now 'no'
Dec 23 02:21:12 lookfar named[11034]: no IPv6 interfaces found
Dec 23 02:21:12 lookfar named[11034]: listening on IPv4 interface lo, \
    127.0.0.1#53
Dec 23 02:21:12 lookfar named[11034]: listening on IPv4 interface eth0, \
    10.0.0.129#53
Dec 23 02:21:12 lookfar named[11034]: command channel listening on \
    127.0.0.1#953
Dec 23 02:21:13 lookfar named[11034]: running
```

If there are any messages about errors then there is a mistake. Named will name the file it is reading. Go back and check the file. Start named over when it is fixed.

Now you can test your setup. Traditionally a program called `nslookup` is used for this. These days `dig` is recommended:

```
$ dig -x 127.0.0.1
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 26669
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 1, ADDITIONAL: 0

;; QUESTION SECTION:
;1.0.0.127.in-addr.arpa.                IN      PTR

;; ANSWER SECTION:
1.0.0.127.in-addr.arpa. 259200  IN      PTR     localhost.

;; AUTHORITY SECTION:
0.0.127.in-addr.arpa.   259200  IN      NS      ns.linux.bogus.

;; Query time: 3 msec
;; SERVER: 127.0.0.1#53(127.0.0.1)
;; WHEN: Sun Dec 23 02:26:17 2001
;; MSG SIZE  rcvd: 91
```

If that's what you get it's working. We hope. Anything very different, go back and check everything. Each time you change a file you need to run `rndc reload`.

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:

```
$ dig pat.uio.no
; <<>> DiG 9.1.3 <<>> pat.uio.no
;; global options:  printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 15574
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 3, ADDITIONAL: 0

;; QUESTION SECTION:
;pat.uio.no.                    IN      A

;; ANSWER SECTION:
pat.uio.no.             86400   IN      A       129.240.130.16

;; AUTHORITY SECTION:
uio.no.                 86400   IN      NS      nissen.uio.no.
uio.no.                 86400   IN      NS      nn.uninett.no.
uio.no.                 86400   IN      NS      ifi.uio.no.

;; Query time: 651 msec
;; SERVER: 127.0.0.1#53(127.0.0.1)
;; WHEN: Sun Dec 23 02:28:35 2001
;; MSG SIZE  rcvd: 108
```

This time `dig` 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:

```
$ dig pat.uio.no

; <<>> DiG 8.2 <<>> pat.uio.no
;; res options: init recurs defnam dnsrch
;; got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 4
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 3, ADDITIONAL: 3
;; QUERY SECTION:
;;      pat.uio.no, type = A, class = IN

;; ANSWER SECTION:
pat.uio.no.             23h59m58s IN A  129.240.130.16

;; AUTHORITY SECTION:
UIO.NO.                 23h59m58s IN NS  nissen.UIO.NO.
UIO.NO.                 23h59m58s IN NS  ifi.UIO.NO.
UIO.NO.                 23h59m58s IN NS  nn.uninett.NO.

;; ADDITIONAL SECTION:
nissen.UIO.NO.          23h59m58s IN A  129.240.2.3
ifi.UIO.NO.             1d23h59m58s IN A  129.240.64.2
nn.uninett.NO.          1d23h59m58s IN A  158.38.0.181

;; Total query time: 4 msec
;; FROM: lookfar to SERVER: default -- 127.0.0.1
;; WHEN: Sat Dec 16 00:23:09 2000
;; MSG SIZE  sent: 28  rcvd: 162
```

As you can plainly see this time it was much faster, 4ms versus more than half a second earlier. The answer was cached. With cached answers there is the possibility that the answer is out of date, but the origin servers can control the time cached answers should be considered valid, so there is a high probability that the answer you get *is* valid.

3.2 Resolvers

All OSes implementing the standard C API has the calls gethostbyname and gethostbyaddr. These can get information from several different sources. Which sources it gets it from is configured in `/etc/nsswitch.conf` on Linux (and some other Unixes). This is a long file specifying from which file or database to get different kinds of data types. 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
```

(*You remembered about the leading spaces, right? I won't mention them again.*)

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`.

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. Forwarding

In large, well organized, academic or ISP (Internet Service Provider) networks you will sometimes find that the network people have set up a forwarder hierarchy of DNS servers which helps lighten the internal network load and the load on the outside servers as well. It's not easy to know if you're inside such a network or not. But 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. This works by your nameserver forwarding queries to your ISP nameserver. Each time this happens you will dip into the big cache of your ISPs nameserver, thus speeding your queries up, your nameserver does not have to do all the work itself. 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;
 };
```

There is also a nice trick for dialup machines using forwarders, it is described in the qanda section.

Restart your nameserver and test it with `dig`. Should still work fine.

5. A *simple* domain.

How to set up your own domain.

5.1 But first some dry theory

First of all: you read all the stuff before here right? You have to.

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', as is usual for tree data−structures. 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. The dots are nodes, the edges are on the names.

When looking for a machine the query proceeds recursively into the hierarchy starting at the root. If you want to find the address of `prep.ai.mit.edu.`, your nameserver has to start asking somewhere. It starts by looking it its cache. If it knows the answer, having cached it before, it will answer right away as we saw in the last section. If it does not know it will see how closely it can match the requested name and use whatever information it has cached. In the worst case there is no match but the `.' (root) of the name, and the root servers have to be consulted. It will remove the leftmost parts one at a time, checking if it knows anything about `ai.mit.edu.`, then `mit.edu.`, then `edu.`, and if not that it does know about `.` because that was in the hints file. It will then ask a `.` server about `prep.ai.mit.edu`. This `.` server will not know the answer, but it will help your server on its way by giving a referral, telling it where to look instead. These referrals will eventually lead your server to a nameserver that knows the answer. I will illustrate that now. `+norec` means that dig is asking non−recursive questions so that we get to do the recursion ourselves. The other options are to reduce the amount of dig produces so this won't go on for too many pages:

```
$ ;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 980
;; flags: qr ra; QUERY: 1, ANSWER: 0, AUTHORITY: 13, ADDITIONAL: 0

;; AUTHORITY SECTION:
.                       518400  IN      NS      J.ROOT-SERVERS.NET.
.                       518400  IN      NS      K.ROOT-SERVERS.NET.
.                       518400  IN      NS      L.ROOT-SERVERS.NET.
.                       518400  IN      NS      M.ROOT-SERVERS.NET.
.                       518400  IN      NS      A.ROOT-SERVERS.NET.
.                       518400  IN      NS      B.ROOT-SERVERS.NET.
.                       518400  IN      NS      C.ROOT-SERVERS.NET.
.                       518400  IN      NS      D.ROOT-SERVERS.NET.
.                       518400  IN      NS      E.ROOT-SERVERS.NET.
.                       518400  IN      NS      F.ROOT-SERVERS.NET.
.                       518400  IN      NS      G.ROOT-SERVERS.NET.
.                       518400  IN      NS      H.ROOT-SERVERS.NET.
.                       518400  IN      NS      I.ROOT-SERVERS.NET.
```

This is a referral. It is giving us an "Authority section" only, no "Answer section". Our own nameserver refers us to a nameserver. Pick one at random:

```
$ dig +norec +noques +nostats +nocmd prep.ai.mit.edu. @D.ROOT-SERVERS.NET.
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 58260
;; flags: qr; QUERY: 1, ANSWER: 0, AUTHORITY: 3, ADDITIONAL: 3

;; AUTHORITY SECTION:
mit.edu.                172800  IN      NS      BITSY.mit.edu.
mit.edu.                172800  IN      NS      STRAWB.mit.edu.
mit.edu.                172800  IN      NS      W20NS.mit.edu.

;; ADDITIONAL SECTION:
BITSY.mit.edu.          172800  IN      A       18.72.0.3
STRAWB.mit.edu.         172800  IN      A       18.71.0.151
W20NS.mit.edu.          172800  IN      A       18.70.0.160
```

It refers us to MIT.EDU servers at once. Again pick one at random:

```
$ dig +norec +noques +nostats +nocmd prep.ai.mit.edu. @BITSY.mit.edu.
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 29227
;; flags: qr ra; QUERY: 1, ANSWER: 1, AUTHORITY: 4, ADDITIONAL: 4

;; ANSWER SECTION:
prep.ai.mit.edu.        10562   IN      A       198.186.203.77

;; AUTHORITY SECTION:
ai.mit.edu.             21600   IN      NS      FEDEX.ai.mit.edu.
ai.mit.edu.             21600   IN      NS      LIFE.ai.mit.edu.
ai.mit.edu.             21600   IN      NS      ALPHA-BITS.ai.mit.edu.
ai.mit.edu.             21600   IN      NS      BEET-CHEX.ai.mit.edu.

;; ADDITIONAL SECTION:
FEDEX.ai.mit.edu.       21600   IN      A       192.148.252.43
LIFE.ai.mit.edu.        21600   IN      A       128.52.32.80
ALPHA-BITS.ai.mit.edu.  21600   IN      A       128.52.32.5
BEET-CHEX.ai.mit.edu.   21600   IN      A       128.52.32.22
```

This time we got a "ANSWER SECTION", and an answer for our question. The "AUTHORITY SECTION" contains information about which servers to ask about `ai.mit.edu` the next time. So you can ask them directly the next time you wonder about `ai.mit.edu` names. Named also gathered information about `mit.edu`, so of `www.mit.edu` is requested it is much closer to being able to answer the question.

So starting at `.` we found the successive name servers for each level in the domain name by referral. 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. One climbs the tree by taking the name we want (`prep.ai.mit.edu`) asking the root (`.`) or whatever servers father from the root toward `prep.ai.mit.edu` we have information about in the cache. Once the cache limits are reached the recursive resolver goes out asking servers, pursuing referrals (edges) further into the name.

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 host's name when we have its address. A important thing to

note here is that the IP addresses are written in reverse order in the `in-addr.arpa` domain. If you have the address of a machine: `198.186.203.77` named proceeds to find the named 77.203.168.198.in-addr.arpa/ just like it did for `prep.ai.mit.edu`. Example: Finding no cache entry for any match but `.', ask a root server, `m.root-servers.net` refers you to some other root servers. `b.root-servers.net` refers you directly to bitsy.mit.edu/. You should be able to take it from there.

5.2 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 (BIND 9 will not bug you if you break this rule, BIND 8 will). 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:

```
$TTL 3D
@               IN      SOA     ns.linux.bogus. hostmaster.linux.bogus. (
                                1       ; Serial
                                8H      ; Refresh
                                2H      ; Retry
                                4W      ; 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 previous 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 (Domain Name Pointer) 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, at the top (but after the `$TTL` directive). It describes the zone, where it comes from (a machine called `ns.linux.bogus`), who is responsible for its contents (`hostmaster@linux.bogus`; you should insert your e-mail address here), what version of the zone file this is (serial: 1), and other things having to do with caching and secondary DNS servers. For the rest of the fields (refresh, retry, expire and minimum) use the numbers used in this HOWTO and you should be safe. Before the SOA comes a mandatory line, the `$TTL 3D` line. Put it in all your zone files.

Now restart your named (`rndc stop; named`) and use `dig` to examine your handy work. `-x` asks for the inverse query:

```
$ dig -x 127.0.0.1
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 30944
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 1, ADDITIONAL: 0

;; QUESTION SECTION:
;1.0.0.127.in-addr.arpa.                IN      PTR

;; ANSWER SECTION:
1.0.0.127.in-addr.arpa. 259200  IN      PTR     localhost.

;; AUTHORITY SECTION:
0.0.127.in-addr.arpa.   259200  IN      NS      ns.linux.bogus.

;; Query time: 3 msec
;; SERVER: 127.0.0.1#53(127.0.0.1)
;; WHEN: Sun Dec 23 03:02:39 2001
;; MSG SIZE  rcvd: 91
```

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" {
        type master;
        notify no;
        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
;
$TTL 3D
@       IN      SOA     ns.linux.bogus. hostmaster.linux.bogus. (
                        199802151       ; serial, todays date + todays serial #
                        8H              ; refresh, seconds
                        2H              ; retry, seconds
                        4W              ; 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 the machine mentioned in the SOA record. Its 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 to `mail.linux.bogus` or `mail.friend.bogus`. The number before each machine name is that MX RR's 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.

Reload named by running `rndc reload`. Examine the results with `dig`:

```
$ dig any linux.bogus
; <<>> DiG 9.1.3 <<>> any linux.bogus
;; global options:  printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 55239
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 4, AUTHORITY: 1, ADDITIONAL: 1

;; QUESTION SECTION:
;linux.bogus.                   IN      ANY

;; ANSWER SECTION:
linux.bogus.            259200  IN      SOA     ns.linux.bogus. \
      hostmaster.linux.bogus. 199802151 28800 7200 2419200 86400
linux.bogus.            259200  IN      NS      ns.linux.bogus.
linux.bogus.            259200  IN      MX      20 mail.friend.bogus.
linux.bogus.            259200  IN      MX      10 mail.linux.bogus.linux.bogus.
```

```
;; AUTHORITY SECTION:
linux.bogus.        259200  IN      NS      ns.linux.bogus.

;; ADDITIONAL SECTION:
ns.linux.bogus.     259200  IN      A       192.168.196.2

;; Query time: 4 msec
;; SERVER: 127.0.0.1#53(127.0.0.1)
;; WHEN: Sun Dec 23 03:06:45 2001
;; MSG SIZE  rcvd: 184
```

Upon careful examination you will discover a bug. The line

```
linux.bogus.        259200  IN MX        10 mail.linux.bogus.linux.bogus.
```

is all wrong. It should be

```
linux.bogus.        259200  IN MX        10 mail.linux.bogus.
```

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

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

It 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
;
$TTL 3D
@       IN      SOA     ns.linux.bogus. hostmaster.linux.bogus. (
```

```
                        199802151       ; serial, todays date + todays serial #
                        8H              ; refresh, seconds
                        2H              ; retry, seconds
                        4W              ; 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
                TXT     "The router"

ns              A       192.168.196.2
                MX      10 mail
                MX      20 mail.friend.bogus.
www             CNAME   ns

donald          A       192.168.196.3
                MX      10 mail
                MX      20 mail.friend.bogus.
                TXT     "DEK"

mail            A       192.168.196.4
                MX      10 mail
                MX      20 mail.friend.bogus.

ftp             A       192.168.196.5
                MX      10 mail
                MX      20 mail.friend.bogus.
```

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 an A record, so it is inadvisable to have

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

but correct to have

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

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

```
$ dig linux.bogus axfr

; <<>> DiG 9.1.3 <<>> linux.bogus axfr
;; global options:  printcmd
```

```
linux.bogus.            259200  IN      SOA     ns.linux.bogus. hostmaster.linux.bogus. 199802
linux.bogus.            259200  IN      NS      ns.linux.bogus.
linux.bogus.            259200  IN      MX      10 mail.linux.bogus.
linux.bogus.            259200  IN      MX      20 mail.friend.bogus.
donald.linux.bogus.     259200  IN      A       192.168.196.3
donald.linux.bogus.     259200  IN      MX      10 mail.linux.bogus.
donald.linux.bogus.     259200  IN      MX      20 mail.friend.bogus.
donald.linux.bogus.     259200  IN      TXT     "DEK"
ftp.linux.bogus.        259200  IN      A       192.168.196.5
ftp.linux.bogus.        259200  IN      MX      10 mail.linux.bogus.
ftp.linux.bogus.        259200  IN      MX      20 mail.friend.bogus.
gw.linux.bogus.         259200  IN      A       192.168.196.1
gw.linux.bogus.         259200  IN      TXT     "The router"
localhost.linux.bogus.  259200  IN      A       127.0.0.1
mail.linux.bogus.       259200  IN      A       192.168.196.4
mail.linux.bogus.       259200  IN      MX      10 mail.linux.bogus.
mail.linux.bogus.       259200  IN      MX      20 mail.friend.bogus.
ns.linux.bogus.         259200  IN      MX      10 mail.linux.bogus.
ns.linux.bogus.         259200  IN      MX      20 mail.friend.bogus.
ns.linux.bogus.         259200  IN      A       192.168.196.2
www.linux.bogus.        259200  IN      CNAME   ns.linux.bogus.
linux.bogus.            259200  IN      SOA     ns.linux.bogus. hostmaster.linux.bogus. 199802
;; Query time: 41 msec
;; SERVER: 127.0.0.1#53(127.0.0.1)
;; WHEN: Sun Dec 23 03:12:31 2001
;; XFR size: 23 records
```

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

```
$ dig www.linux.bogus

; <<>> DiG 9.1.3 <<>> www.linux.bogus
;; global options:  printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 16633
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 2, AUTHORITY: 1, ADDITIONAL: 0

;; QUESTION SECTION:
;www.linux.bogus.               IN      A

;; ANSWER SECTION:
www.linux.bogus.        259200  IN      CNAME   ns.linux.bogus.
ns.linux.bogus.         259200  IN      A       192.168.196.2

;; AUTHORITY SECTION:
linux.bogus.            259200  IN      NS      ns.linux.bogus.

;; Query time: 5 msec
;; SERVER: 127.0.0.1#53(127.0.0.1)
;; WHEN: Sun Dec 23 03:14:14 2001
;; MSG SIZE  rcvd: 80
```

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.

5.3 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 by 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" {
        type master;
        notify no;
        file "pz/192.168.196";
};
```

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

```
$TTL 3D
@       IN      SOA     ns.linux.bogus. hostmaster.linux.bogus. (
                        199802151 ; Serial, todays date + todays serial
                        8H      ; Refresh
                        2H      ; Retry
                        4W      ; 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 reload your named (`rndc reload`) and examine your work with `dig` again:

```
$ dig -x 192.168.196.4
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 58451
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 1, ADDITIONAL: 1

;; QUESTION SECTION:
;4.196.168.192.in-addr.arpa.    IN      PTR

;; ANSWER SECTION:
4.196.168.192.in-addr.arpa. 259200 IN   PTR     mail.linux.bogus.

;; AUTHORITY SECTION:
196.168.192.in-addr.arpa. 259200 IN     NS      ns.linux.bogus.

;; ADDITIONAL SECTION:
ns.linux.bogus.         259200  IN      A       192.168.196.2
```

```
;; Query time: 4 msec
;; SERVER: 127.0.0.1#53(127.0.0.1)
;; WHEN: Sun Dec 23 03:16:05 2001
;; MSG SIZE  rcvd: 107
```

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

```
$ dig 196.168.192.in-addr.arpa. AXFR

; <<>> DiG 9.1.3 <<>> 196.168.192.in-addr.arpa. AXFR
;; global options:  printcmd
196.168.192.in-addr.arpa. 259200 IN     SOA     ns.linux.bogus. \
        hostmaster.linux.bogus. 199802151 28800 7200 2419200 86400
196.168.192.in-addr.arpa. 259200 IN     NS      ns.linux.bogus.
1.196.168.192.in-addr.arpa. 259200 IN   PTR     gw.linux.bogus.
2.196.168.192.in-addr.arpa. 259200 IN   PTR     ns.linux.bogus.
3.196.168.192.in-addr.arpa. 259200 IN   PTR     donald.linux.bogus.
4.196.168.192.in-addr.arpa. 259200 IN   PTR     mail.linux.bogus.
5.196.168.192.in-addr.arpa. 259200 IN   PTR     ftp.linux.bogus.
196.168.192.in-addr.arpa. 259200 IN     SOA     ns.linux.bogus. \
        hostmaster.linux.bogus. 199802151 28800 7200 2419200 86400
;; Query time: 6 msec
;; SERVER: 127.0.0.1#53(127.0.0.1)
;; WHEN: Sun Dec 23 03:16:58 2001
;; XFR size: 9 records
```

Looks good! If your output didn't look like that look for error−messages in your syslog, I explained how to do that in the first section under the heading Starting named

5.4 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 and later 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.

5.5 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 doesn'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 if you're a small 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 in part that you'll get less than a ``C'' subnet and some things may break. Please see Ask Mr. DNS 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 dig.

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. (Very) 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.

5.6 Slave servers

Once you have set up your zones correctly on the master servers you need to set up at least one slave server. Slave servers are needed for robustness. If your master goes down the people out there on the net will still be able to get information about your domain from the slave. A slave should be as long away from you as possible. Your master and slave should share as few as possible of these: Power supply, LAN, ISP, city and country. If all of these things are different for your master and slave you've found a really good slave.

A slave is simply a nameserver that copies zone files from a master. You set it up like this:

```
zone "linux.bogus" {
        type slave;
        file "sz/linux.bogus";
        masters { 192.168.196.2; };
};
```

A mechanism called zone−transfer is used to copy the data. The zone transfer is controlled by your SOA record:

```
@       IN      SOA     ns.linux.bogus. hostmaster.linux.bogus. (
                        199802151       ; serial, todays date + todays serial #
                        8H              ; refresh, seconds
                        2H              ; retry, seconds
                        4W              ; expire, seconds
                        1D )            ; minimum, seconds
```

A zone is only transferred if the serial number on the master is larger than on the slave. Every refresh interval the slave will check if the master has been updated. If the check fails (because the master is unavailable) it will retry the check every retry interval. If it continues to fail as long as the expire interval the slave will remove the zone from it's filesystem and no longer be a server for it.

6. Basic security options.

By Jamie Norrish

Setting configuration options to reduce the possibility of problems.

There are a few simple steps that you can take which will both make your server more secure and potentially reduce its load. The material presented here is nothing more than a starting point; if you are concerned about security (and you should be), please consult other resources on the net (see the last chapter).

The following configuration directives occur in `named.conf`. If a directive occurs in the `options` section of the file, it applies to all zones listed in that file. If it occurs within a `zone` entry, it applies only to that zone. A `zone` entry overrides an `options` entry.

6.1 Restricting zone transfers

In order for your slave server(s) to be able to answer queries about your domain, they must be able to transfer the zone information from your primary server. Very few others have a need to do so. Therefore restrict zone transfers using the `allow-transfer` option, assuming 192.168.1.4 is the IP address of ns.friend.bogus and adding yourself for debugging purposes:

```
zone "linux.bogus" {
```

```
      allow-transfer { 192.168.1.4; localhost; };
};
```

By restricting zone transfers you ensure that the only information available to people is that which they ask for directly – no one can just ask for all the details about your set–up.

6.2 Protecting against spoofing

Firstly, disable any queries for domains you don't own, except from your internal/local machines. This not only helps prevent malicious use of your DNS server, but also reduces unnecessary use of your server.

```
options {
      allow-query { 192.168.196.0/24; localhost; };
};

zone "linux.bogus" {
      allow-query { any; };
};

zone "196.168.192.in-addr.arpa" {
      allow-query { any; };
};
```

Further, disable recursive queries except from internal/local sources. This reduces the risk of cache poisoning attacks (where false data is fed to your server).

```
options {
        allow-recursion { 192.168.196.0/24; localhost; };
};
```

6.3 Running named as non–root

It is a good idea to run named as a user other than root, so that if it is compromised the privileges gained by the cracker are as limited as possible. You first have to create a user for named to run under, and then modify whatever init script you use that starts named. Pass the new user name and group to named using the –u and –g flags.

For example, in Debian GNU/Linux 2.2 you might modify your `/etc/init.d/bind` script to have the following line (where user `named` have been created):

```
start-stop-daemon --start --quiet --exec /usr/sbin/named -- -u named
```

The same can be done with Red Hat and the other distributions.

Dave Lugo has described a secure dual chroot setup http://www.etherboy.com/dns/chrootdns.html which you may find interesting to read, it makes the host your run your named on even more secure.

7. 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.

7.1 /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";
};

controls {
        inet 127.0.0.1 allow { localhost; } keys { rndc_key; };
};

key "rndc_key" {
        algorithm hmac-md5;
        secret "c3Ryb25nIGVub3VnaCBmb3IgYSBtYW4gYnV0IG1hZGUgZm9yIGEgd29tYW4K";
};

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.

7.2 /var/named/root.hints

Keep in mind that this file is dynamic, and the one listed here is old. You're better off using a new one 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
```

7.3 /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.

```
$TTL 3D
@               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.
```

If you look at a random BIND installation you will probably find that the `$TTL` line is missing as it is here. It was not used before, and only version 8.2 of BIND has started to warn about its absence. BIND 9 *requires* the `$TTL`.

7.4 /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.

```
$TTL 3D
@       IN      SOA     land-5.com. root.land-5.com. (
                        199609206       ; serial, todays date + todays serial #
                        8H              ; refresh, seconds
                        2H              ; retry, seconds
                        4W              ; 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. But, as mentioned earlier, BIND 9 no longer enforces this restriction.

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.

7.5 /var/named/zone/206.6.177

I'll comment on this file below

```
$TTL 3D
@               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 FTP server and accept connections from FTP clients. As you are a Norwegian FTP server you want to accept more connections from clients in Norway and other Scandinavian countries and less from the rest of the world. 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 FTP 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`.

The FTP server prioritizes 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 put the connection in a connection class with less bandwidth and fewer clients allowed. 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 at all, it may even deny the connection for lack of classification.

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*.

8. 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
# named up-test suggested by Erik Bryer.
#
(
 echo "To: hostmaster <hostmaster>"
 echo "From: system <root>"

 # Is named up? Check the status of named.
 case `rndc status 2>&1` in
    *refused*)
        echo "named is DOWN. root.hints was NOT updated"
        echo
        exit 0
        ;;
 esac

 PATH=/sbin:/usr/sbin:/bin:/usr/bin:
 export PATH
 # NOTE: /var/named must be writable only by trusted users or this script
 # will cause root compromise/denial of service opportunities.
 cd /var/named 2>/dev/null || {
    echo "Subject: Cannot cd to /var/named, error $?"
    echo
    echo "The subject says it all"
    exit 1
 }

 # Are we online?  Ping a server at your ISP
 case `ping -qnc 1 some.machine.net 2>&1` in
   *'100% packet loss'*)
```

```
        echo "Subject: root.hints NOT updated.  The network is DOWN."
        echo
        echo "The subject says it all"
        exit 1
        ;;
 esac

 dig @e.root-servers.net . ns >root.hints.new 2> errors

 case `cat root.hints.new` in
   *NOERROR*)
        # It worked
        :;;
   *)
        echo "Subject: The root.hints file update has FAILED."
        echo
        echo "The root.hints update has failed"
        echo "This is the dig output reported:"
        echo
        cat root.hints.new errors
        exit 1
        ;;
 esac

 echo "Subject: The root.hints file has been updated"
 echo
 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 errors
 mv root.hints root.hints.old
 mv root.hints.new root.hints
 rndc 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.

9. Migrating to BIND 9

The BIND 9 distribution, and the prepackaged versions too, contains a document called `migration` that contains notes about how to migrate from BIND 8 to BIND 9. The document is very straight forward. If you installed binary packages it's likely stored in `/usr/share/doc/bind*` or `/usr/doc/bind*` somewhere.

If you're running BIND 4, you may find a document called `migration-4to9` in the same place.

10. 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://langfeldt.net/DNS−HOWTO/

2. How do use DNS from inside a firewall?

   A hint: `forward only;`. You might 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 four items regarding this:

- Specific to BIND 8/9, Adam L Rice has sent me this e−mail, about how to run DNS painlessly on a dialup machine:

```
I have discovered with newer versions of BIND that this
[<em/shuffeling files, -ed/] is no longer necessary.  There is a
"forward" directive in addition to the "forwarders" directive that
controls how they are used.  The default setting is "forward first",
which first asks each of the forwarders, and then tries the normal
approach of doing the legwork itself if that fails.  This gives the
familiar behaviour of gethostbyname() taking an inordinately long time
when the link is not up.  But if "forward only" is set, then BIND
gives up when it doesn't get a response from the forwarders, and
gethostbyname() returns immediately.  Hence there is no need to
perform sleight-of-hand with files in /etc and restart the server.

In my case, I just added the lines

forward only;
forwarders { 193.133.58.5; };

to the options { } section of my named.conf file. It works very
nicely. The only disadvantage of this is that it reduces an incredibly
sophisticated piece of DNS software to the status of a dumb cache. To
some extent, I would just like to run a dumb cache for DNS instead,
but there doesn't seem to be such a piece of software available for
Linux.
```

- 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
```

Peter Denison thought that Ian does not go far enough though. He writes:

```
When connected) serve all cached (and local network) entries immediately
```

```
                for non-cached entries, forward to my ISPs nameserver
When off-line)  serve local network queries immediately
                fail all other queries **immediately**

The combination of changing the root cache file and forwarding queries
doesn't work.

So, I've set up (with some discussion of this on the local LUG) two nameds
as follows:

named-online:   forwards to ISPs nameserver
                master for localnet zone
                master for localnet reverse zone (1.168.192.in-addr.arpa)
                master for 0.0.127.in-addr.arpa
                listens on port 60053

named-offline:  no forwarding
                "fake" root cache file
                slave for 3 local zones (master is 127.0.0.1:60053)
                listens on port 61053

And combined this with port forwarding, to send port 53 to 61053 when
off-line, and to port 60053 when online. (I'm using the new netfilter
package under 2.3.18, but the old (ipchains) mechanism should work.)

Note that this won't quite work out-of-the-box, as there's a slight bug in
BIND 8.2, which I have logged wth the developers, preventing a slave
having a master on the same IP address (even if a different port). It's a
trivial patch, and should go in soon I hope.
```

- 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.
```

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 domain? 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.

10. How can I secure my DNS server? How do I set up split DNS?

    Both of these are advanced topics. They are both covered in http://www.etherboy.com/dns/chrootdns.html. I will not explain the topics further here.

---

11. 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.

I have written *The Concise Guide to DNS and BIND* (by Nicolai Langfeldt, me), published by Que (ISBN 0−7897−2273−9). The book is much like this HOWTO, just more details, and a lot more of everything. It has also been translated to Polish and published as *DNS i BIND* by Helion (http://helion.pl/ksiazki/dnsbin.htm, ISBN 83−7197−446−9). Now in 4th edition is *DNS and BIND* by Cricket Liu and P. Albitz from O'Reilly & Associates (ISBN 0−937175−82−X, affectionately known as the Cricket book). Another book is *Linux DNS Server Administration*, by Craig Hunt, published by Sybex (ISBN 0782127363), I have not read it yet. Another must for good DNS administration (or good anything for that matter) is *Zen and the Art of Motorcycle Maintenance* by Robert M. Pirsig.

Online you will find my book, along with tons of other books, available electronically as a subscription service at http://safari.informit.com/. There is stuff on http://www.dns.net/dnsrd/ (DNS Resources Directory), http://www.isc.org/bind.html; A FAQ, a reference manual (the ARM should be enclosed in the BIND distribution as well) 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. The newsgroup news:comp.protocols.tcp−ip.domains is about DNS. In addition there are a number of RFCs about DNS, the most important are probably the ones listed here. Those that have BCP (Best Current Practice) numbers are *highly recommended*.

RFC 2671

P. Vixie, *Extension Mechanisms for DNS (EDNS0)* August 1999.

RFC 2317

BCP 20, H. Eidnes et. al. *Classless IN−ADDR.ARPA delegation*, March 1998. This is about CIDR, or classless subnet reverse lookups.

RFC 2308

M. Andrews, *Negative Caching of DNS Queries*, March 1998. About negative caching and the $TTL zone file directive.

RFC 2219

BCP 17, M. Hamilton and R. Wright, *Use of DNS Aliases for Network Services*, October 1997. About CNAME usage.

RFC 2182

BCP 16, R. Elz et. al., *Selection and Operation of Secondary DNS Servers*, July 1997.

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*. Only 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.