DNS HOWTO

Nicolai Langfeldt janl@math.uio.no v1.1, 30 June 1996

HOWTO become a totally small time DNS admin.

Contents

Keywords: DNS, bind, named, dialup, ppp, slip, Internet, domain, name, hosts, resolving

Legal stuff:

 The author wishes to thank Arnt Gulbrandsen who read the drafts to this work countless times and provided many useful suggestions.

Other stuff:

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 money, comments and/or questions to janl@ifi.uio.no.

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

1 Introduction.

What this is and isn't.

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. For more complex setups you will need to *read* the Real Documentation. I'll get back to what this Real Documentation consists of in 7.

Before you start on this you should configure your machine so that you can telnet in and out of it, and make 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. You also need a good `/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 networking/NET-2 HOWTO explains how to set it up. Read it.

If you're using SLIP or PPP you need that working. Read the PPP HOWTO if it's not.

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 4.

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.vix.com:/pub/bind in either the release or testing subdirectory, whatever fits your lifestyle best.

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 overloaded 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.

2 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.

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

```
;  Boot file for nicolais caching name server
;
directory /var/named
;
; type          domain                          source file or host
cache           .                               root.cache
primary         0.0.127.in-addr.arpa            pz/127.0.0
```

The 'directory' line tells named where to look for files. All files named subsequently will be relative to this. /var/named is the right directory according to the *Linux Filesystem Standard*. Personaly, I use /local/named, but I'm quirky.

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

```
 .       518400  NS      D.ROOT-SERVERS.NET.
 .       518400  NS      E.ROOT-SERVERS.NET.
 .       518400  NS      I.ROOT-SERVERS.NET.
 .       518400  NS      F.ROOT-SERVERS.NET.
 .       518400  NS      G.ROOT-SERVERS.NET.
 .       518400  NS      A.ROOT-SERVERS.NET.
 .       518400  NS      H.ROOT-SERVERS.NET.
 .       518400  NS      B.ROOT-SERVERS.NET.
 .       518400  NS      C.ROOT-SERVERS.NET.
;
D.ROOT-SERVERS.NET.     3600000 A       128.8.10.90
E.ROOT-SERVERS.NET.     3600000 A       192.203.230.10
I.ROOT-SERVERS.NET.     3600000 A       192.36.148.17
F.ROOT-SERVERS.NET.     3600000 A       192.5.5.241
G.ROOT-SERVERS.NET.     3600000 A       192.112.36.4
A.ROOT-SERVERS.NET.     3600000 A       198.41.0.4
H.ROOT-SERVERS.NET.     3600000 A       128.63.2.53
B.ROOT-SERVERS.NET.     3600000 A       128.9.0.107
C.ROOT-SERVERS.NET.     3600000 A       192.33.4.12
```

The file describes the root name servers in the world. This changes over time and *must* be maintained. See the 5 for how to keep it up to date. This file is described in the named man page, but it is, IMHO, best suited for people that already understand named.

The next line in named.boot is the primary line. 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     linux.bogus. hostmaster.linux.bogus. (
                                1       ; Serial
                                28800   ; Refresh
                                7200    ; Retry
                                604800  ; Expire
                                86400)  ; 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 hostnames you want to connect to. The '`nameserver`' line specifies what address your machine can reach a nameserver at, in this case your own machine since that is where your named runs. (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`, `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, 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.

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.

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. 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 starting with `order` and it should look like this:

```
order hosts,bind
```

If there is no '`order`' line you should stick one in. 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) These two latest files are documented in the resolv(8) manpage (do '`man 8 resolv`') in most Linux distributions. That man page is IMHO readable, and everyone, especially DNS admins, should read it. Do it now, if you say to yourself "I'll do it later" you'll never get around to it.

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 back-fires try '`/usr/sbin/ndc start`' instead. Now you can test your setup. If you view your syslog message file (usually called `/var/adm/messages`) while starting named (do `tail -f /var/adm/messages`) you should see something like:

```
Jun 30 21:50:55 roke named[2258]: starting.  named 4.9.4-REL Sun Jun 30 21:29:0
3 MET DST 1996   janl@roke.slip.ifi.uio.no:/var/tmp/bind/named
Jun 30 21:50:55 roke named[2258]: cache zone "" loaded (serial 0)
Jun 30 21:50:55 roke named[2258]: primary zone "0.0.127.in-addr.arpa" loaded (s
erial 1)
```

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

```
$ 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.boot` 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:  129.240.2.50
```

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.cache` file, and asked its way from there. It might take tiny while before you get the result as it searches all the domains you named in `/etc/resolv.conf`.

If you try 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, it instead looked in it's cache and found it there. But the cached information *might* be out of date (stale). So you are informed of this (very slight) danger by it saying '`Non-authorative answer:`'. When `nslookup` says this the second time you ask for a host it's a sure sign it named caches the information and that it's working. You exit `nslookup` by giving the command '`exit`'.

If you're a dialup (ppp, slip) user please read the 6, there is some advice there for you.

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

3 A *simple* domain.

How to set up your own domain.

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

DNS is a hierarchical 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. When 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.cache` file is for), the `.` server gives a list of edu servers. I'll illustrate this here:

```
$ 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 the server we're asking that edu is right under ., this narrows the search somewhat.

```
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 `*.root-servers.net` serves `edu.`, so we can go on 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 = STRAWB.mit.edu
mit.edu nameserver = W20NS.mit.edu
mit.edu nameserver = BITSY.mit.edu

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

`steawb`, `w20ns` and `bitsy` serves `mit`, select one and inquire about `ai.mit.edu`:

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

Hostnames 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 = WHEATIES.AI.MIT.EDU
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

Authoritative answers can be found from:
AI.MIT.EDU      nameserver = WHEATIES.AI.MIT.EDU
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
WHEATIES.AI.MIT.EDU     internet address = 128.52.32.13
WHEATIES.AI.MIT.EDU     internet address = 128.52.35.13
ALPHA-BITS.AI.MIT.EDU   internet address = 128.52.32.5
ALPHA-BITS.AI.MIT.EDU   internet address = 128.52.37.5
GRAPE-NUTS.AI.MIT.EDU   internet address = 128.52.32.4
GRAPE-NUTS.AI.MIT.EDU   internet address = 128.52.36.4
TRIX.AI.MIT.EDU internet address = 128.52.32.6
TRIX.AI.MIT.EDU internet address = 128.52.38.6
MUESLI.AI.MIT.EDU       internet address = 128.52.32.7
MUESLI.AI.MIT.EDU       internet address = 128.52.39.7
```

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

```
> server WHEATIES.AI.MIT.EDU.
Default Server:  WHEATIES.AI.MIT.EDU
Addresses:  128.52.32.13, 128.52.35.13
```

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:  WHEATIES.AI.MIT.EDU
Addresses:  128.52.32.13, 128.52.35.13

prep.ai.mit.edu CPU = dec/decstation-5000.25    OS = unix
prep.ai.mit.edu
        inet address = 18.159.0.42, protocol = tcp
         #21 #23 #25 #79
prep.ai.mit.edu preference = 1, mail exchanger = life.ai.mit.edu
prep.ai.mit.edu internet address = 18.159.0.42
```

```
ai.mit.edu      nameserver = alpha-bits.ai.mit.edu
ai.mit.edu      nameserver = wheaties.ai.mit.edu
ai.mit.edu      nameserver = grape-nuts.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 = life.ai.mit.edu
ai.mit.edu      nameserver = mintaka.lcs.mit.edu
life.ai.mit.edu internet address = 128.52.32.80
alpha-bits.ai.mit.edu   internet address = 128.52.32.5
wheaties.ai.mit.edu     internet address = 128.52.35.13
wheaties.ai.mit.edu     internet address = 128.52.32.13
grape-nuts.ai.mit.edu   internet address = 128.52.36.4
grape-nuts.ai.mit.edu   internet address = 128.52.32.4
mini-wheats.ai.mit.edu  internet address = 128.52.32.11
mini-wheats.ai.mit.edu  internet address = 128.52.54.11
mintaka.lcs.mit.edu     internet address = 18.26.0.36
```

So starting at `.` we found the successive nameservers for the next 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.

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 it's address. A important thing here is to note that ip#s are written in reverse order in the in-addr.arpa domain. If you have the address of a machine: 192.128.52.43 named procedes 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'). Also, the reversion of the numbers is kinda confusing the first 2 years.

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

Our own doman.

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.

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

```
primary         0.0.127.in-addr.arpa            pz/127.0.0
```

Please note the lack of '`.`' at the end of the domain names in this file. The first line names the file `pz/127.0.0` as defining `0.0.127.in-addr.arpa`. We've already set up this file, it reads:

```
@               IN      SOA     linux.bogus. hostmaster.linux.bogus. (
                                1       ; Serial
                                28800   ; Refresh
```

```
                              7200    ; Retry
                              604800  ; Expire
                              86400)  ; 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.boot` 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 'domain' column of the `named.boot` 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.arps the first line really says

```
0.0.127.IN-ADDR.ARPA. IN      SOA ...
```

NS is the Name Server RR, it tells DNS what machine is the name server of the domain. And finally the PTR record says that 1 (equals 1.0.0.127.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, the very first record. It describes the zone, where it comes from (a machine called `linux.bogus`), who is responsible for its contents (`hostmaster@linux.bogus`), 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.

The NS record tells us who does DNS serving for `0.0.127.in-addr.arpa`, it is `ns.linux.bogus`. The PTR record tells us that `1.0.0.127.in-addr.arpa` (aka 127.0.0.1) is known as `localhost`.

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. Don't push it. Now for our main task, the `linux.bogus` domain, insert a new primary line in `named.boot`:

```
primary                 linux.bogus                     pz/linux.bogus
```

Note the continued lack of ending '.' on the domain name in the `named.boot` file.

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

```
;
; Zone file for linux.bogus
;
; Mandatory minimum for a working domain
;
@       IN      SOA     linux.bogus. hostmaster.linux.bogus. (
                        199511301       ; serial, todays date + todays serial #
                        28800           ; refresh, seconds
                        7200            ; retry, seconds
                        3600000         ; expire, seconds
                        86400 )         ; minimum, seconds
                NS      ns.linux.bogus.
                NS      ns.friend.bogus.
                MX      10 mail.linux.bogus   ; Primary Mail Exchanger
                MX      20 mail.friend.bogus. ; Secondary Mail Exchanger

localhost       A       127.0.0.1
ns              A       127.0.0.2
mail            A       127.0.0.4
```

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 primarily. If that fails it 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 = linux.bogus
        mail addr = hostmaster.linux.bogus
        serial = 199511301
        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     nameserver = ns.friend.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
linux.bogus     nameserver = ns.friend.bogus
ns.linux.bogus  internet address = 127.0.0.2
mail.linux.bogus        internet address = 127.0.0.4
```

Upon careful examination you will discover a bug. The line

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

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 it's end. 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. 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.boot 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
;
; Mandatory minimum for a working domain
;
@       IN      SOA     linux.bogus. hostmaster.linux.bogus. (
```

```
                        199511301       ; serial, todays date + todays serial #
                        28800           ; refresh, seconds
                        7200            ; retry, seconds
                        604800          ; expire, seconds
                        86400 )         ; minimum, seconds

                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
ns              A       127.0.0.2
mail            A       127.0.0.4
;
; Extras
;
@               TXT     "Linux.Bogus, your DNS consultants"

ns              MX      10 mail
                MX      20 mail.friend.bogus.
                HINFO   "Pentium" "Linux 1.2"
                TXT     "RMS"
richard         CNAME   ns
www             CNAME   ns

donald          A       127.0.0.3
                MX      10 mail
                MX      20 mail.friend.bogus.
                HINFO   "i486"  "Linux 1.2"
                TXT     "DEK"

mail            MX      10 mail
                MX      20 mail.friend.bogus.
                HINFO   "386sx" "Linux 1.0.9"

ftp             A       127.0.0.5
                MX      10 mail
                MX      20 mail.friend.bogus.
                HINFO   "P6" "Linux 1.3.59"
```

You might want to move the first three A records so that they're placed next to their respective other records, instead on top like that.

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. ns has a Pentium CPU and runs Linux 1.2. The TXT record is a free text record that you can use for anything you like. CNAME (Canonical NAME) is a way to give each machine several names. So richard and www is a alias for ns. It's important to note that A MX, CNAME and SOA record should *never* refer to a CNAME record, they should only refer to something with a A record, so it would wrong to have

```
foobar  CNAME   richard                 ; NO!
```

but correct to have

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

It's also important to note 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       127.0.0.2
```

Paul Vixie, the primary named wizard, recommends *not* using CNAME. So consider not using it *very* seriously.

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.

```
[localhost]
 linux.bogus.                 SOA   linux.bogus hostmaster.linux.bogus. (1995
11301 28800 7200 604800 86400)
 linux.bogus.                 NS    ns.linux.bogus
 linux.bogus.                 NS    ns.friend.bogus
 linux.bogus.                 MX    10   mail.linux.bogus
 linux.bogus.                 MX    20   mail.friend.bogus
 linux.bogus.                 TXT   "Linux.Bogus, your DNS consultants"
 localhost                    A     127.0.0.1
 mail                         A     127.0.0.4
 mail                         MX    10   mail.linux.bogus
 mail                         MX    20   mail.friend.bogus
 mail                         HINFO 386sx       Linux 1.0.9
```

```
 donald                          A     127.0.0.3
 donald                          MX    10   mail.linux.bogus
 donald                          MX    20   mail.friend.bogus
 donald                          HINFO i486        Linux 1.2
 donald                          TXT   "DEK"
 www                             CNAME ns.linux.bogus
 richard                         CNAME ns.linux.bogus
 ftp                             A     127.0.0.5
 ftp                             MX    10   mail.linux.bogus
 ftp                             MX    20   mail.friend.bogus
 ftp                             HINFO P6          Linux 1.3.59
 ns                              A     127.0.0.2
 ns                              MX    10   mail.linux.bogus
 ns                              MX    20   mail.friend.bogus
 ns                              HINFO Pentium     Linux 1.2
 ns                              TXT   "RMS"
 linux.bogus.                    SOA   linux.bogus hostmaster.linux.bogus. (1995
11301 28800 7200 604800 86400)
```

That's good. 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
```

...In other words, the real name of `www.linux.bogus` is `ns.linux.bogus`

```
linux.bogus     nameserver = ns.linux.bogus
linux.bogus     nameserver = ns.friend.bogus
ns.linux.bogus  internet address = 127.0.0.2
```

and ns.linux.bogus has the address 127.0.0.2. Looks good too.

Winding down

Of course, this domain is highly bogus, and so are all the addresses in it. For a real domain insert the real domain names and addresses and all the other information. When that is done you need a reverse lookup zone file, it should be like the 127.0.0 file and contain exactly one PTR RR for each address in use, i.e.

```
127.0.0.2       PTR     ns.linux.bogus.
127.0.0.3       PTR     donald.linux.bogus.
127.0.0.4       PTR     mail.linux.bogus.
127.0.0.5       PTR     ftp.linux.bogus.
```

for our example domain (in addition to the SOA RR of-course). The domain name (origin) of the file should be reversed just as with 127.0.0 versus 0.0.127.in-addr.arpa.

Now it's time for you to play around with things and set up your domain. When you've finished playing with the `linux.bogus` domain remove it from your `named.boot` file.

4 Firewalls and other odd things.

Q: How do use DNS from inside a firewall? A: A couple of hints: 'forwarders', 'slave', and have a look in the literature list at the end of this HOWTO.

Q: 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? A: Make several A records for www.busy.site and bind 4.9.3 or later will round-robin the answers. It will *not* work with earlier versions of bind.

5 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.cache` file updated. The easiest way is using dig, first run dig with no arguments, you will get the `root.cache` 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.cache file except for a couple of extra numbers. Those numbers are harmless. Save it to a file (`dig . @e.root-servers.net >root.cache.new`) and replace the old `root.cache` with it.

Remember to restart named after replacing the cache file.

Al Longyear sent me this script that can be run automatically to update `named.cache`, install it a crontab entry for it and forget it. The script assumes you have mail working and that the mail-alias 'hostmaster' is defined. You should 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.
#
(
 echo "To: hostmaster <hostmaster>"
 echo "From: system <root>"
 echo "Subject: Automatic update of the named.boot file"
 echo

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

```
 dig . @rs.internic.net >named.cache.new

 echo "The named.boot file has been updated to contain the following
information:"
 echo
 cat named.boot.new

 chown root.root named.cache.new
 chmod 444 named.cache.new
 rm -f named.cache.old
 mv named.cache named.cache.old
 mv named.cache.new named.cache
 ndc restart
 echo
 echo "The nameserver has been restarted to ensure that the update is complete.
"
 echo "The previous named.cache file is now called
/var/named/named.cache.old."
) 2>&1 | /usr/lib/sendmail -t
exit 0
```

6 Automatic setup for dialup connections.

This section explains how I have set things up to automate everything. My way might not suit you at all, but you might get a idea from something I've done. Also, I use ppp for dialup, while many use slip or cslip, so almost everything in your setup can be different from mine. But slip's dip program should be able to do many of the things I do.

Normally, when I'm not connected to the net I have a `resolv.conf` file simply containing the line

```
domain uio.no
```

This ensures I don't have to wait for the hostname resolving library to try to connect to a nameserver that can't help me. But when I connect I want to start my named and have a `resolv.conf` looking like the one described above. I have solved this by keeping two `resolv.conf` 'template' files named `resolv.conf.local` and `resolv.conf.connected`. The latter looks like the `resolv.conf` described before in this document.

To automatically connect to the net I run a script called 'ppp-on':

```
#!/bin/sh
echo calling...
pppd
```

pppd has a file called `options` that tells it the particulars of how to get connected. Once my ppp connection is up the pppd starts a script called `ip-up` (this is described in the pppd man page). This is parts of the script:

```
#!/bin/sh
interface="$1"
device="$2"
speed="$3"
myip="$4"
upip="$5"
 ...
cp -v /etc/resolv.conf.connected /etc/resolv.conf
 ...
/usr/sbin/named
```

I.e. I start my named there. When ppp is disconnected pppd runs a script called `ip-down`:

```
#!/bin/sh
cp /etc/resolv.conf.local /etc/resolv.conf
read namedpid </var/run/named.pid
kill $namedpid
```

So this gets things configured and up when connecting and Dis-configured and down when disconnecting.

Some programs, irc and talk come to mind, make a few too many assumptions, and for irc the dcc features and talk to work right you have to fix your hosts file. I insert have this in my `ip-up` script:

```
cp /etc/hosts.ppp /etc/hosts
echo $myip      roke >>/etc/hosts
```

`hosts.ppp` simply contains

```
127.0.0.1       localhost
```

and the echo thing inserts the ip# i have received for my host name (roke). You should use the name your host knows itself by instead. This can be found with the hostname command.

It is probably not smart to run named when you are not connected to the net, this is because named will try to send queries to the net and it has a long timeout, and you have to wait for this timeout every time some program tries to resolve a name. If you're using dialup you should start named when connecting and kill it when disconnecting. I have received mail saying it isn't so, but I have not been able to make it work having to wait for long timeouts. Please mail *all* details if you have better information.

Some people like to use a forwarders directive on slow connections. If your internet provider has DNS servers at 1.2.3.4 and 1.2.3.5 you can insert the line

```
forwarders 1.2.3.4 1.2.3.5
```

in the `named.boot` file. Also leave the `named.cache` file empty. That will decrease the amount of IP traffic your host originates, any possibly speed things up. This especially important if you're paying pr. byte that goes over the wire. This has the added value of letting you off the one maintenance duty you have as a caching named maintainer: you don't have to update a empty `named.cache` file.

7 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. 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. Prisig :-) Available as ISBN 0688052304 and others.

Online you will find stuff on `<http://www.dns.net/dnsrd/>`, `<http://www.vix.com/isc/bind.html>`; A FAQ, a reference manual (BOG; Bind Operations Guide) as well as papers and protocol definitions and DNS hacks. 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 extatic 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 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 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.