UnixWorld Online: ‘‘PC-Unix Connection’’ Column: No. 005

DNS Database Files

By Tom Yager

With the foundation of the previous column beneath you, you should be ready to start building your own DNS databases. There’s a lot of ground to cover, so I’ll dispense with the preliminaries, save one: the examples given here are actual DNS databases, pulled from my network. The host names, IP addresses and other data are real, so don’t try to use them on your site. If you use these files as templates for your own DNS databases, please remember to replace all the supplied data.

The listings will display without the need to scroll horizontally if you resize your browser window until you can display this entire line in the window:

```
|----------------------------------------------------------------------|
```

The file `/etc/named.boot` tells the `named` software on your system where to find its configuration tables. On Maxx, I chose to place this data under `/usr/local/bind` to distinguish it from system data that would be replaced in an OS update or re-install. The `named.boot` file on my system looks like this:

```
1  directory      /usr/local/bind
2  primary        maxx.net                    maxx.net.hosts
3  primary        0.0.127.in-addr.arpa        named.local
4  primary        17.251.204.in-addr.arpa     maxx.net.revhosts
5  cache          .                           named.ca
```

The line numbers are shown for reference; don’t enter them. Line 1 tells `named` where to find the files, and also saves you having to type the fully qualified path names. You’re free to place your database files anywhere, but I recommend you use a file system on your primary (boot) drive. If a secondary drive fails or gets unmounted, you can keep your name server running.

Line 2 starts with the key word *primary*, indicating that the database file contains primary server data. A primary server maintains all its databases locally. A secondary server loads DNS data from another server (primary or secondary). To keep these examples simple, we’ll stick with definitions for a primary server.

The line 2 entry, maxx.net tells `named` which domain the file (here, `maxx.net.hosts`) defines. You may define multiple domains with one invocation of `named`. Just create a database file for each domain you wish to define. Here are the contents of the primary host database for the domain maxx.net:

```
;
;   Addresses for the local domain

maxx.net.   IN      SOA     maxx.maxx.net. tyager.maxx.maxx.net. (
                            9602171         ; Serial
                            36000           ; Refresh every 10 hours
                            3600            ; Retry after 1 hour
```

```
                            360000          ; Expire after 100 hours
                            36000           ; Minimum TTL is 10 hours
                            )

;   Define name servers
;
maxx.net.   IN      NS      maxx.maxx.net.
maxx.net.   IN      A       204.251.17.241

;   Define localhost
;
localhost   IN  A           127.0.0.1

;   Set up hosts
;
maxx        IN  A           204.251.17.241
            IN  MX      5   maxx.maxx.net.

maxx.net.   IN  MX      5   maxx.maxx.net.
;
;   All mail for net delivered to maxx
;
;*          IN  MX      10  maxx.maxx.net.
www         IN  CNAME       maxx.maxx.net.
ftp         IN  CNAME       maxx.maxx.net.
news        IN  CNAME       maxx.maxx.net.
mail        IN  CNAME       maxx.maxx.net.
ns          IN  CNAME       maxx.maxx.net.
loghost     IN  CNAME       maxx.maxx.net.
lucy        IN  A           204.251.17.242
linux       IN  CNAME       lucy.maxx.net.
lucy        IN  MX      10  lucy.maxx.net.
messdos     IN  A           204.251.17.243
messdos     IN  MX      10  messdos.maxx.net.
pentium     IN  CNAME       messdos.maxx.net.
solaris     IN  A           204.251.17.244
solaris     IN  MX      10  solaris.maxx.net.
maxx4       IN  CNAME       solaris.maxx.net.
maxx5       IN  A           204.251.17.245
maxx5       IN  MX      10  maxx5.maxx.net.
maxx6       IN  A           204.251.17.246
maxx6       IN  MX      10  maxx6.maxx.net.
```

Most database file entries are known as DNS resource records. Generally, the resource records are shown ordered: SOA, NS, followed by the other types, but this ordering isn't required. The data in each entry may be entered in upper-, lower-, or mixed case. All entries in the database file must start at the beginning of the line. Blank lines as well as any text following a semicolon is ignored.

SOA stands for Start of Authority. This self-impressed acronym clues `named` that operational parameters follow. By far the most important is the Serial field. Every time you make a change to a database file, you *must* increment its serial number. Only by doing this will secondary servers know they need to reach into your system and pull out new name server data, a procedure known as a ''zone transfer.'' Many DNS administrators use a date-time stamp for this field, like 9602171 for the first version on February 17, 1996.

First, let's focus on the SOA section:

```
maxx.net.   IN      SOA     maxx.maxx.net. tyager.maxx.maxx.net. (
```

The ‘‘maxx.net.’’ field tells `named` the domain defined by this file. The name server will automatically append it to any host name that appears in the file. The trailing dot is not a typo: it keeps `named` from trying to tack on your domain name. Without it, the resolver would be confused by `named`’s expansion of my domain name to ‘‘maxx.net.maxx.net.’’

The IN stands for the ‘‘Internet’’ class of data. Even though other classes exist, they aren’t in common usage. The ‘‘maxx.maxx.net’’ field is the host on which these database files reside. Finally, ‘‘tyager.maxx.maxx.net’’ represents the e-mail address of the DNS administrator, where the first dot (between tyager and maxx) would be replaced by the at-sign (`@`) to create a valid address. (The at-sign can’t be used here because it has a reserved meaning in DNS database files.)

The open parenthesis at the end of the line lets you to split the SOA record across physical lines for readability:

```
                            9602171     ; Serial

                            36000       ; Refresh every 10 hours
                            3600        ; Retry after 1 hour
                            360000      ; Expire after 100 hours
                            36000       ; Minimum TTL is 10 hours
                            )
```

We discussed the ‘‘serial’’ field above. The remaining four fields specify various time intervals (all values in seconds) used by the secondary name server:

Refresh
: The time interval that must elapse between each poll of the primary by the secondary name server (here 36,000 seconds or 10 hours). If the ‘‘serial number’’ has been updated on the primary, the secondary assumes its data is stale and requests updated information as a ‘‘zone transfer’’.

Retry
: The time interval used between successive connection attempts by the secondary to reach the primary name server in case the first attempt failed (here 3,600 seconds or one hour). Generally, less than the ‘‘refresh’’ time.

Expire
: The time interval after which the secondary expires its data if it can’t reach the primary name server (here 360,000 seconds or 100 hours). The secondary will refuse to service requests after this interval.

Minimum
: The minimum time-to-live value, which specifies how long other servers should cache data from the name server (here 36,000 seconds or 10 hours).

There are several types of resource records, identified by the key word in field three of each record. You may present records in any order, but try to organize them for clarity (whatever that suggests to you). The NS (name server) record tells the hosts that query your server where the name servers for this domain can be found:

```
maxx.net.   IN      NS      maxx.maxx.net.
```

The address of the host maxx.maxx.net isn’t defined until later, but it doesn’t matter. It gets used in the

SOA record as well, so I relax my backward-reference ban in this case. You may list multiple name servers for your domain. In fact, your domain should have at least two name servers. As I said, your Internet service provider will probably allow you to use their name server as a secondary for your domain. Don't forget the trailing dots!

The first `A` record, which resolves a fully-qualified host name to an IP address, is a special one. It defines an IP address for unqualified queries, that is, queries for the host maxx.net. It's easy for my small network, which only has one host that offers any services to the outside world. Users can access it with a few saved keystrokes, and my users have the luxury of simplified e-mail addresses (for instance, `tyager@maxx.net`).

Other `A` records like this one:

```
lucy         IN  A       204.251.17.242
```

provide name-to-address mapping for a specific named host. The domain defined in this file (maxx.net) is appended to the host name you show in the first field.

The `CNAME` records create aliases for existing hosts. These examples illustrate a few common uses:

```
www          IN  CNAME   maxx.maxx.net.
ftp          IN  CNAME   maxx.maxx.net.
```

You may give a host any alias you like, and as many aliases as you like. The host needn't answer to that name, that is, the alias doesn't need to be the host's true name as reported by `hostname` or `uname`.

The other vital type of record is `MX`. This tells SMTP e-mail software where to send mail for each named host:

```
lucy         IN  MX      10 lucy.maxx.net.
```

When a remote host's mail delivery program sees an e-mail address in your domain, it will query your name server for its applicable MX record or records. This is wonderfully versatile. Every user on your LAN can receive e-mail, even if not every host is running its own e-mail software. The `MX` record for lucy, for instance, could easily redirect e-mail to another host on the LAN.

The number (10 in this case) in the fourth field represents a preference value. If you define multiple MX records for a host, delivery is attempted to lower-preference value hosts first. The actual value isn't important, only its relationship to other preference values.

On larger LANs it's a good idea to create backup e-mail servers. Smaller LANs, like mine, can simply rely on the fact that most SMTP mailers will retry deliveries to my site for three days before returning a message to its sender.

The line--shown commented out here--would arrange to redirect e-mail for all hosts in this domain to a single machine:

```
;
; All mail for net delivered to maxx
;
;*           IN  MX      10 maxx.maxx.net.
```

This is an exceedingly good idea for company LANs that benefit from a central e-mail repository.

Address-to-name mapping

Reverse-mapping files let resolvers post queries armed with only the IP address. This reverse mapping is used, for example, by Internet server software that prefers to log host names rather than less informative IP addresses.

Your host will require at least two reverse-mapping files. The first, defined on line two of the sample `named.boot` file, sets up the reverse mapping of the `localhost` entry:

```
;
; Addresses in the local domain
;
@           IN  SOA     maxx.maxx.net tyager.maxx.maxx.net. (
                        9602171 ; Serial
                        36000   ; Refresh every 100 hours
                        3600    ; Retry after 1 hour
                        3600000 ; Expire after 1000 hours
                        36000   ; Minimum TTL is 100 hours
                        )

maxx.net.   IN  NS      maxx.maxx.net.
1           IN  PTR     localhost.
```

There's only one host in this database. The at-sign in the first column of the SOA record is shorthand for ‘‘insert my domain here.’’ The domain, as defined by this database file's entry in the `named.boot` file, is 0.0.127.in- addr.arpa. As discussed in my previous column, this domain is recognized as special. Among its special qualities is that the network address identified in the domain (127.0.0 in this case) is reversed before it is is processed. Therefore, the PTR entry in the above example uses one (`1`) as the suffix to the IP address given in `named.boot`, creating the IP address 127.0.0.1.

The reverse-mapping database for the rest of the domain, set up in line four of the sample `named.boot`, looks like this:

```
;
;   Reverse addressing for the local domain
;
@           IN      SOA     maxx.maxx.net. tyager.maxx.maxx.net. (
                            9602171     ; Serial
                            36000       ; Refresh every 10 hours
                            3600        ; Retry after 1 hour
                            360000      ; Expire after 100 hours
                            36000       ; Minimum TTL is 10 hours
                            )
;
;   Define name server
;
            IN      NS      maxx.maxx.net.

;
;   Addresses point to canonical names:
;
```

```
241          IN       PTR      maxx.maxx.net.
242          IN       PTR      lucy.maxx.net.
243          IN       PTR      messdos.maxx.net.
244          IN       PTR      solaris.maxx.net.
245          IN       PTR      maxx5.maxx.net.
246          IN       PTR      maxx6.maxx.net.
```

If you leave field one blank (as in the NS record for this example) `named` will use previous record's field-one entry. In this case, the at-sign gets replaced by the domain name from line four of `named.boot`.

Only one line from `named.boot` remains to be discussed, the ''cache'' entry. This is a bit of a misnomer as it doesn't have anything to do with local caching. Instead, it defines the master root domain name servers for the Internet. You can retrieve this list from ftp://nic.ddn.mil/netinfo/root-servers.txt. You will need to check this site periodically to ensure you have the latest list.

This file lists the root domain servers in human-readable format. You'll need to reformat it for consumption by `named`. Here's what the cache file looks like, using the root servers defined when this article was written (February 1996).

```
;            Servers from the root domain
;            ftp://nic.ddn.mil/netinfo/root-servers.txt
;
.                    99999999    IN      NS  A.ROOT-SERVERS.NET
.                    99999999    IN      NS  B.ROOT-SERVERS.NET
.                    99999999    IN      NS  C.ROOT-SERVERS.NET
.                    99999999    IN      NS  D.ROOT-SERVERS.NET
.                    99999999    IN      NS  E.ROOT-SERVERS.NET
.                    99999999    IN      NS  F.ROOT-SERVERS.NET
.                    99999999    IN      NS  G.ROOT-SERVERS.NET
.                    99999999    IN      NS  H.ROOT-SERVERS.NET
.                    99999999    IN      NS  I.ROOT-SERVERS.NET

;            Root servers by address

A.ROOT-SERVERS.NET  99999999    IN      A   198.41.0.4
B.ROOT-SERVERS.NET  99999999    IN      A   128.9.0.107
C.ROOT-SERVERS.NET  99999999    IN      A   192.33.4.12
D.ROOT-SERVERS.NET  99999999    IN      A   128.8.10.90
E.ROOT-SERVERS.NET  99999999    IN      A   192.203.230.10
F.ROOT-SERVERS.NET  99999999    IN      A   192.5.5.241
G.ROOT-SERVERS.NET  99999999    IN      A   192.112.36.4
H.ROOT-SERVERS.NET  99999999    IN      A   128.63.2.53
I.ROOT-SERVERS.NET  99999999    IN      A   192.36.148.17
```

Here, the dot (`.`) refers to the root domain and the 99999999 means a *very long* time-to-live value. The TTL value is no longer used for caching because the data isn't discarded if it times out, but administrators generally keep it around because it does no harm.

Testing your name server

The book, *DNS and BIND* by Paul Albitz and Cricket Liu, published by O'Reilly and Associates, includes instructions for detailed diagnostics. However, you can perform some simple checks on your name server's health with `nslookup`. This utility is standard with every TCP/IP-network aware version of Unix I've used. There are other similar tools--such as `dig`--in Internet archives; check *DNS and BIND* for details.

You can find the source code for `dig` at several anonymous FTP archive sites, including ftp://ftp.wonderland.org/NetBSD/NetBSD-current/src/usr.sbin/named/dig/ for the NetBSD release. Use Archie to find other sites.

The `nslookup` utility can be used interactively, much like other programs, such as `ftp`. That is, if you invoke this program without command-line arguments, it displays a prompt and waits for your command:

```
Default Server:  localhost
Address:  127.0.0.1

>
```

By default, `nslookup` performs queries based on host names you submit. Just enter a host name after the prompt:

```
> messdos
Server:  localhost
Address:  127.0.0.1

Name:    messdos.maxx.net
Address:  204.251.17.243
```

because `nslookup` will tack on your domain name. Check your reverse-mapping entries by entering an IP address:

```
> 204.251.17.245
Server:  localhost
Address:  127.0.0.1

Name:    maxx5.maxx.net
Address:  204.251.17.245
```

You can check your MX and NS records by using `nslookup`'s `set type` command:

```
> set type=mx
> maxx.net
Server:  localhost
Address:  127.0.0.1

maxx.net        preference = 5, mail exchanger = maxx.maxx.net
maxx.maxx.net   internet address = 204.251.17.241
> lucy.maxx.net
Server:  localhost
Address:  127.0.0.1

lucy.maxx.net   preference = 10, mail exchanger = lucy.maxx.net
lucy.maxx.net   internet address =3D 204.251.17.242
> set type=ns
> maxx.net
Server:  localhost
Address:  127.0.0.1

maxx.net        nameserver = maxx.maxx.net
maxx.maxx.net   internet address = 204.251.17.241
```

The Fat Lady Sings

I also recommend using `nslookup` to check host names outside your domain. This will help ensure you got your root domain cache database right and that your host's `named` can forward queries to other servers. You should also avail yourself of a system outside your LAN to make sure your name server can be seen and understood by others. This will also show you when the NIC has added your entries to the root servers. Until that's done, no one outside your LAN can see your name server without having its IP address. Once the NIC gets your domain set up, `nslookup` on another host will suddenly know where to look to resolve names and addresses in your domain. Make sure you're ready to roll before you start publishing any host names in your domain. Internet users have a notoriously low tolerance for sites which are advertised but unreachable.

From here, administration is simple as long as your domain doesn't grow by a huge leap. Just download the root servers list from the NIC periodically and keep your ears open for mail from users about sites on your LAN that can't be seen. As simple as all these files seem, it's ridiculously easy to make basic mistakes that are hard to find. I had the devil's own time getting my DNS databases configured; Becca Thomas (my UWOL editor) patiently struggled squeezing e-mail through my initially -busted DNS setup. A messed-up DNS configuration can make your system hard to reach from the outside, and the symptoms of that difficulty don't always make the problem clear. It helps, too, to have some experts around. It should be a criteria in choosing an Internet provider that good help is easily available. I owe my provider, FastLane Communications (http://www.fastlane.net), a link for being so helpful when I was getting maxx.net on line.

That's it for our series on DNS. It's no replacement for a more thorough text, but we hope it helps you get started. Next month we'll help you face what you may (or may not!) dread: connecting Windows 95 systems to your TCP/IP LAN. Until then, thanks for reading.

Edited by Becca Thomas / Online Editor / UnixWorld Online / editor@unixworld.com

Go To Content   Search

Last Modified: Sunday, 10-Mar-96 10:02:24 PST