DNS HOWTO

Nicolai Langfeldt (dns-howto(at)langfeldt.net), Jamie Norrish 他 Version 9.0, 2001-12-20

中野武雄 nakano(at)apm.seikei.ac.jp v9.0j1, 2002-02-03

短時間で DNS 管理者になる方法。

Contents

1 前書き

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

この文書は Linux Documentation Project の一部です。(訳注: 翻訳版は Japanese FAQ Project の一部です)

1.1 法的なこと

1.2 謝辞とヘルプ募集

本 HOWTO の査読をお願いしたすべての人々 (それぞれの方がご存じのはず)、提案や情報を電子メールで送ってくださったすべての読者に感謝します。

この文書はまだ完成したものではありません。この文書をより良い物にするために、問題点や成功例などについて筆者にメールを送って下さい。コメント・質問、現金などは janl(at)langfeldt.net まで。あるいは私の DNS 本を買ってください (題名は "The Concise Guide to DNS and BIND です。ISBN は参考文献リストにあります)。メールを送り、返信を希望する場合には、返信先のアドレスが正しいか、またちゃんと機能しているかどうかを確認して下さるようにお願いします。またメールする前には必ず 10 (Q & A) のセクションを読んでください。なお、私が読めるのはノルウェー語と英語に限られます。

これは HOWTO 文書です。私は 1995 年から、この文書を LDP の一部として管理してきました。2000 年に、私はこのトピックに関する書籍を書きました。お断りしておきたいのですが、この HOWTO はいろいろな点でその本と似ていますけれども、本の売り上げを伸ばすためにこの HOWTO で手抜きをしたようなことはありません。この HOWTO の読者は、DNS の理解がいかに難しいものであるかを私に教えてくださいました。それによってこの本は良いものになりましたし、また一方本を書くことで、この HOWTO に何が必要なのかを考えさせられることにもなりました。この HOWTO がその本を産み、またその本がこの HOWTO の第三版を産むことになりました。このチャンスを私に下さったことに対して、出版社の Que に感謝します :-)

訳注: この文書の v1.0 は、横田邦彦さんと藤原輝嘉さんとが翻訳されました。中野が v2.1.1 にあわせて更新し、以降の管理を行っています。更新の際には、ご意見をいただいた藤原さん・遠藤さん・花高さん・水原さん、校正をしてくださった長谷川さん・武井さんをはじめ、JF-ML の皆さんにお世話になりました。

翻訳に関するコメントは nakano(at)apm.seikei.ac.jp までお願いします。DNS に関する日本語での質問先としては

linux-users メーリングリスト `http://www.linux.or.jp/community/ml/linux-users/`

や *fj.os.linux.networking* , *fj.net.ip.dns*

などが適当でしょう。

1.3 献辞

この HOWTO 文書を Anne Line Norheim Langfeldt に捧げる。といっても彼女がこの文書を読むことは無いだろうけど。そういった類の女の子じゃないからなあ。

1.4 最新版

この HOWTO の更新版は、

`http://langfeldt.net/DNS-HOWTO/` または

`http://www.linuxdoc.org/` で見つかるはずです。この文書が 9 か月以上前の日付だったら、こちらに行ってください。

2 はじめに

この文書は何であって何ではないか。

DNS とは Domain Name System のことです。 DNS はマシンの名前を IP 番号 (ネットワーク上のマシンには必ずこの番号が付いています) に変換します。 DNS は名前からアドレスへの、またアドレスから名前への翻訳 (あるいは仲間うちの言葉でいえば「マップ」) などを行います。この HOWTO 文書では、 Unix システムを用いてこのようなマップを定義する方法について記述します。なお Linux に特有なことがらもいくつか含まれています。

「マップ」とは、単に二つのものを結びつけることです。ここでは `ftp.linux.org` といったようなマシンの名前と、そのマシンの IP 番号 (IP アドレス) である `199.249.150.4` のような値を結びつけることになります。 DNS には逆向きのマップも含まれます。すなわち、 IP 番号からマシンの名前への変換です。これは「逆引き」と呼ばれています。

初心者 (あなた ;-) にとって DNS は、ネットワーク管理のなかでもわかりにくい部分の一つです。幸い DNS は実際にはそれほど難しくはありません。この HOWTO では、いくつかの事柄を多少なりともわかるようにしたいと思っています。簡単な DNS ネームサーバを設定する方法も説明します。まずキャッシュ専用のサーバからはじめて、あるドメインに対するプライマリ DNS サーバを設定していきます。もっと複雑な設定を行なう場合には、この文書の 10 (Q & A) の章を参照してください。そこにも書いていなかったら、もっとちゃんとした文献を読む必要があるでしょう。「ちゃんとした文献」については、 11 (より熟練した管理者になるために) の章で説明します。

DNS についての作業を始める前に、あなたのマシンを設定して、 telnet での出入りやネットへの各種接続ができるようにしておいてください。特に `telnet 127.0.0.1` で、現在のマシン自身にログインできるようにしてください (今すぐテスト !)。また `/etc/nsswitch.conf` (あるいは `/etc/host.conf`)、 `/etc/resolv.conf`、`/etc/hosts` などのファイルに対して、正しい設定をしておいてください。これらの機能についてはこの文書では説明しません。以上の準備ができていない場合は、 Networking-HOWTO や Networking-Overview-HOWTO に説明がありますから、ちゃんと読んで設定しておいてください。

この文書で「あなたのマシン」と書いてあった場合、それは DNS を動作させようとしているマシンを指すものとします。他にもネットワークにつながっているあなたのマシンはあるでしょうけど、それのことではありません。

あなたのマシンが所属しているネットワークには、名前引きをブロックするような防火壁 (ファイアウォール) は

存在しないものとします。ファイアウォール内部にいる場合には特別な設定が必要になります。 10 (Q & A) の章を見てください。

UNIX システムでの名前引きのサービスは `named` と呼ばれるプログラムによって実現されます。これは *Internet Software Consortium* の “BIND” パッケージに含まれるプログラムです。 `named` は、ほとんどの Linux ディストリビューションに含まれています。たいていは `BIND` という名前のパッケージに入っていて (大文字小文字はメンテナの気分次第でしょうが)、 `/usr/sbin/named` としてインストールされます。

もし named がすでにあれば、それを使えばいいでしょう。もし無い場合には Linux の ftp サイトからバイナリを入手するか、最新の (そして最高の) ソースを

`ftp://ftp.isc.org/isc/bind9/`

から入手しましょう。この HOWTO では BIND の version 9 を対象にしています。 BIND 4 や 8 を対象にした古いバージョンの HOWTO は

`http://www.math.uio.no/~janl/DNS/`

にありますので、 BIND 4 を使っている人はこちらを参照してください (ついでにこの HOWTO も一緒においてあります)。 named の man ページ (最後の方にある FILES セクション) に `named.conf` に関する記述があれば、あなたの使っているのは BIND 8 または 9 です。逆に `named.boot` に関する記述があれば BIND 4 です。セキュリティに気を使わなければならない人で、 4 を使っている場合は、最新の BIND 8 や 9 にアップグレードするべきでしょう。今すぐに、です。

(訳注) 最後はちょっと意見の分かれるところかも知れません。例えばソースレベルでのセキュリティチェックを行っていることで知られる OpenBSD では、まだ依然として BIND 4 が現役の named だったりします。

DNS はネットワーク全体に広がるデータベースです。データの登録は慎重に行ないましょう。変な内容を登録すると、あなたも他の人達も迷惑します。真面目にちゃんと運用すれば、 DNS は恩恵をもたらしてくれるはずです。 DNS の使い方、管理の仕方、デバッグのやりかたを学び、良い管理者になってください。設定ミスでネットを落としたりすることがないようにしましょうね。

注意: 私が変更するように指示したファイルがすでに存在していたら、これらのバックアップを取っておきましょう。作業の結果がうまくいかなかった場合に、元の動いている状態に戻すことができるようにするためです。

2.1 他のネームサーバの実装

この節は Joost van Baal が書きました。

あなたのマシンを DNS サーバにするパッケージは何種類か存在しています。まず BIND パッケージ (`http://www.isc.org/products/BIND/`)、この HOWTO が対象としている実装です。もっとも広く使われているネームサーバで、 1980 年代から登場、普及してきました。現在インターネットでネームサービスを提供しているマシンの大部分が BIND を使っています。 BIND は BSD ライセンスで配布されています。もっとも広く使われているパッケージですから、 BIND に関する文書や知識もたくさん存在します。しかし、 BIND にはセキュリティ上

の問題が生じたこともありました。

それから djbdns (http://djbdns.org/) というのもあります。比較的新しい DNS パッケージで、 Daniel J. Bernstein (qmail の作者でもあります) が書きました。 djbdns は非常にモジュール化されています。いくつもの小さなプログラムが、ネームサーバの扱うべき仕事のそれぞれの部分を扱うのです。 djbdns はセキュリティを念頭において設計されています。ゾーンファイルのフォーマットはより単純で、また大抵の場合は設定も簡単です。しかしあまり有名ではないために、あなたの近くのグルによる助けは、このプログラムに関しては得られないかもしれません。残念ながら、このソフトウェアはオープンソースではありません。作者による宣伝は http://cr.yp.to/djbdns/ad.html

にあります。

DJB のソフトウェアが、古い他のソフトウェアに比べ、本当に進歩したものであるのかどうかは、活発な議論の対象になっています。 BIND vs djbdns に関する討論 (あるいはフレームウォー?) は、

http://www.isc.org/ml-archives/bind-users/2000/08/msg01075.html

にあります。

3 名前解決とキャッシュを行うネームサーバ

DNS 設定の最初の一歩。ダイアルアップ・ケーブルモデム・ ADSL などのユーザにはとても便利です。

Red Hat や、 Red Hat に関連したディストリビューションでは、 bind パッケージ・ bind-utils パッケージ・ caching-nameserver パッケージをインストールするだけで、この HOWTO の最初のセクションの結果と同じものが得られます。 Debian を使っているなら bind と bind-doc をインストールするだけです (あるいは前者に対しては bind9。この文書の執筆時では、 Debian の安定版 (potato) は BIND 9 をサポートしていません)。もちろんこれらのパッケージをインストールするだけでは、この HOWTO を読むことによって得られる知識は手に入りません。ですので、まずパッケージをインストールし、そこでインストールされたファイルを調べながら、読み進んでいくのが良いでしょう。

キャッシュ専用のネームサーバとは、名前引きの結果を記憶しておき、次回の問い合わせの時にその記憶を使って答えるものです。次回からの問い合わせに対する応答は (特に遅い回線を使っている場合には) とても速くなります。

まず最初に /etc/named.conf というファイルが必要です (Debian では /etc/bind/named.conf)。 named は起動するとまずこのファイルを読み込みます。現在のところは、次のような簡単なものでよいでしょう。

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

Linux ディストリビューションのパッケージでは、ここで紹介するそれぞれのファイルに、別の名前をつけているかもしれません。でも内容は同じはずです。

directory の行は、 named が参照するファイルの置き場所を指定するものです。これ以降のすべてのファイル名はここからの相対パスとなります。すなわちディレクトリ pz は /var/named 以下にあり、フルパスで表記すれば

/var/named/pz ということになります。 /var/named は *Linux Filesystem Standard*に準拠した正しいディレクトリ名です。

/var/named/root.hints というファイルの名前はここで付けられています。このファイルの中身は次のようになります。

```
;
; 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.
;
; すでにこのファイルがあった場合は、ここに開始コメントがあるかも
; しれません。なくても問題はありません。
;
; 行頭に空白文字があった場合は、削除してください！ 各行は ;、.
; または文字で始まります。空白で始まることはありません。
;
.                       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
```

このファイルには世界中のルートネームサーバを記述します。これは時間とともに変化していくので、ときどき更新する必要があります。更新の方法は 8 (メンテナンス) の章を見てください。

named.conf の末尾の方には zone セクションがあります。この利用法については後の章で述べるつもりですので、今のところは以下のような内容のファイルを pz サブディレクトリに 127.0.0 という名前で作っておいてください。(ここでもカットアンドペーストするときには先頭のスペースを取り除くようにしてください)

```
$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.
```

key や control といった名前がついたセクションは、この二つでもって、この named がリモートから制御できることを指定しています (rndc というプログラムが用いられます)。ここではローカルホストからの接続でなければならず、エンコードされた秘密鍵での認証が必要になります。この鍵はパスワードのようなものです。 rndc が機能するには、この鍵にマッチする /etc/rndc.conf が必要になります。

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

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

見てわかるように、 secret の指定は同一です。 rndc を他のマシンから使う場合は、それらの時計は 5 分以内に会っていなければなりません。この目的には ntp (xntpd や ntpdate) ソフトウェアを用いることをおすすめします。

次に、以下のような内容の /etc/resolv.conf が必要です。 (同じく空白を取り除くこと *!*)

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

'search' で始まっている行は、問い合わせされたホストを探すドメインの指定です。'nameserver' で始まる行は、ネームサーバのアドレス指定です。今は自分のマシンでネームサーバを動かすので、ローカルホストを指定します。 (注: named はこのファイルを参照しません。参照するのはレゾルバです。注 2: resolv.conf ファイル i には "domain" と書かれた行があるかもしれません。あっても問題ありませんが、 "search" と "domain" の両方を同時には用いないようにしてください。どちらかしか効力を持ちません。)

このファイルの意味を説明しましょう。クライアントが foo の名前引きを行うと、まず最初に foo.subdomain.your-domain.edu を調べ、次に foo.your-domain.edu を試し、最後に foo を調べます。 search 行にあまり多くのドメインを書くと、すべてを調べるのに時間がかかるようになるので、ほどほどにしておくのが良いでしょう。

この例ではあなたのマシンが subdomain.your-domain.edu にあるとしていますので、あなたのマシンの名前はおそらく your-machine.subdomain.your-domain.edu となっているでしょう。なお search 行にはあなたの TLD (Top Level Domain, この場合は 'edu') を含めるべきではありません。頻繁に接続するような特定のドメインがあれば、以下のように search 行にそのドメインを加えてもいいでしょう。 (先頭にスペースがあったら取り去るのを忘れないように。)

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

もちろん実際には本当のドメイン名を書く必要があります。ドメイン名の最後にはピリオドを書かないことに注意してください。これは重要なポイントです。ドメイン名の最後にはピリオドを書かないことに注意してください。

3.1 named を起動する

これらの準備がすんだら named を立ち上げましょう。ダイアルアップ接続をしている人は、まず先に接続してください。では named を起動します。ブートスクリプトから起動する場合は /etc/init.d/named start、named を直接起動する場合は /usr/sbin/named とします。以前の版の BIND で似たようなことを行ったときは、おそらく ndc を使ったことと思います。 BIND 9 では、これは rndc に変わりました。 rndc は named をリモートから制御できますが、 named を起動することはできません。 named を動かしている最中に syslog のメッセージファイル (普通は /var/adm/messages ですが、 Debian では /var/log/daemin ですし、ディレクトリが

/var/log だったり、ファイル名が別だったりするかもしれません) を見ると (tail -f /var/adm/messages とします)、以下のような出力が表示されるはずです:

(行末が \ の行は次の行に続きます)

```
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
```

エラーメッセージがあった場合は、何か間違えているのでしょう。 named は読んでいるそのファイルを名指ししてくれるはずです。戻ってファイルをチェックしてください。修正が終わったら再度 named を起動してください。

さて、ここまで行ってきた設定を試してみましょう。これまでは nslookup がテストのためのプログラムでした。最近では dig が推奨されています。

```
$ 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
```

と表示されれば、うまく動いているはずです。こうなるといいですね。非常に異なった表示が出たら、やり直し、全部再チェックです。named.conf を変更したら、そのたびに rndc reload コマンドを実行する必要があります。

では問い合わせをしてみましょう。あなたの近くにあるマシンの名前を引いてみましょう。私の近く (Oslo 大学) には pat.uio.no というマシンがあります。

```
$ 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
```

今度は、dig はあなたのマシンで動いている named に pat.uio.no を探すよう依頼します。すると named は root.hints ファイルに書かれているネームサーバの一つに接続して、問い合わせをします。/etc/resolv.conf に書かれているドメインすべてについて調べる必要があるかもしれないので、結果が得られるまでに少々時間がかかることがあります。

ここでもう一度同じ問い合わせを行うと、次のような結果になるでしょう。

```
$ 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
```

こんどはずっと速かったことがはっきりわかるでしょう。前は 0.5 秒以上かかっていましたが、今回は 4ms ですみました。サーバからの回答がキャッシュされたのです。キャッシュされた回答は、古くなって現状と異なってしまう可能性もありますが、キャッシュされた回答を正しいと見なせる期間は、回答を返したサーバの側で制御できるので、得られた回答が正しいものである可能性は高いでしょう。

3.2 レゾルバ

標準的な C API を実装しているすべての OS には、 gethostbyname と gethostbyaddr というシステムコールが存在します。これらは何種類かの異なる情報源から情報を取得できます。どの情報源から取得するかは、 Linux なら `/etc/nsswitch.conf` というファイルで設定できます (これを用いている Unix は他にもあります)。これは長いファイルで、どのファイルから、あるいはどのデータベースから、いろいろな種類のデータを取得するかを指定

します。通常は先頭にコメント形式の解説がありますので、読んでおきましょう。読み終わったら ‘hosts:’ ではじまる行を探してください。以下のようになっているはずです。

```
hosts:      files dns
```

(先頭のスペースのことは覚えていますね？これ以上はもう言及しません。)

‘hosts:’ ではじまる行が無ければ、上記のような内容を書いておいてください。これは、プログラムはまず /etc/hosts ファイルを見に行き、次に DNS を resolv.conf にしたがってチェックせよ、と言っています。

3.3 おめでとう

さて、今やあなたはキャッシュ動作をする named の設定方法を知ったわけです。ビールでもミルクでも、お好きなもので乾杯しましょう。

4 フォワード (forwarding)

学術機関や ISP (Internet Service Provider) などの、上手に組織化された大きなネットワークでは、ネットワークのプロ達は DNS サーバに「フォワーダ (forwarder)」と呼ばれる階層を設けていることがあるかもしれません。こうすると、内部のネットワーク負荷や、外部にあるサーバの負荷を下げる効果があるのです。自分がそのようなネットワークの一部にいるのかどうかを知るのはそれほど簡単ではありません。しかしいずれにせよ、接続しているプロバイダの DNS サーバを「フォワーダ」として利用すれば、問い合わせの反応を速くでき、ネットワークへの負荷を下げることができます。これを用いると、あなたのネームサーバは、問い合わせを ISP のネームサーバに行います。問い合わせが起こるたび、ISP のネームサーバの巨大なキャッシュからデータをすくい取ることになります。よって問い合わせの速度は上がり、あなたのネームサーバは自分で全部の仕事をこなさなくても良くなります。モデムを使っている場合は、この効果はかなり大きいです。ここで例として、お使いのネットワークプロバイダには利用が推奨されているネームサーバが二つあるとします。それぞれの IP 番号を 10.0.0.1 と 10.1.0.1 としましょう。このような場合には、お手元の named.conf ファイルの最初のセクション、“options” という名前がついている部分に以下の行を挿入して下さい。

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

ダイアルアップマシン向けにも forwarders を使ったちょっと嬉しいトリックがあります。 10 (Q & A) の章に書いてあります。

ネームサーバを再起動して、 dig でテストしてください。うまくいっていると思います。

5 単純なドメイン

あなた自身のドメインの設定方法

5.1 でもまず最初に退屈な理論

まず最初に: ここまでの内容はちゃんと読みましたか？読んでなければ読むように。

このセクションを実際に始める前に、 DNS の動作に関する理論を少々と、実際の動作例を紹介しておきます。きっと役に立ちますから、ぜひ読みましょう。読みたくなくても、少なくとも流し読みくらいはしておいてください。 `named.conf` ファイルの設定に関する部分まできたら流し読みはストップです。

DNS は階層的なツリー構造のシステムです。その頂点は '.' と記述され、 (ツリー型データ構造での慣例に従い)「ルート (root)」と発音されます。 '.' の下にはたくさんの Top Level Domain (TLD) があります。 `ORG`, `COM`, `EDU`, `NET` などが有名ですが、他にもたくさんあります。実際の木と同じように、このツリー構造は根を持ち、枝分かれします。計算機科学の知識がある人には、 DNS は検索ツリーに見えるでしょう。またそこには節点 (node)、端点 (leaf node)、枝 (edge) があることも見て取れるでしょう。

マシンの検索を行うとき、問い合わせはルートから始まる階層に対して再帰的に行われます。いまホスト `prep.ai.mit.edu.` のアドレスを見つけたいとしましょう。するとネームサーバはどこかに問い合わせを行う必要があります。まずキャッシュにないかどうか探します。もし以前の問合わせがキャッシュに残っていて、答を知っていた場合には、直前の節で見たように、ただちに答を返します。キャッシュに答がなかった場合は、問い合わせのあった名前にどのくらい近い答えが返せるかを調べ、キャッシュされている情報をできるだけ使おうとします。最悪の場合は '.' (ルート) だけがマッチすることになり、よってルートサーバに尋ねる必要があります。ネームサーバは名前の左側の部分を消していき、自分が `ai.mit.edu.`, `mit.edu.`, `edu.` について知っているかチェックしていきます。これらを知らないと `.` に行くわけですが、この答は hints ファイルに書いてあるので、見つかります。ここであなたのネームサーバは `.` のサーバに `prep.ai.mit.edu` に関する問い合わせを行います。この `.` サーバは直接の答は知らないでしょうが、あなたのサーバに参照先を提示し、次にどこに聞けばいいかを教えてくれます。この参照先提示は同じように次々に行われ、あなたのネームサーバは答を知っているネームサーバにまで導かれます。これをいまからお見せしましょう。 `+norec` で dig に再帰的な問合わせをしないように命じ、再帰を我々自身で行うことにします。その他のオプションは、 dig に生成する情報を減らすように命じるもので、紙幅を節約します。

```
$ dig +norec +noques +nostats +nocmd prep.ai.mit.edu.
;; 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.
```

これは参照先の提示です。ここには ”Authority section” しかなく、 ”Answer section” がありません。私たちの立てたネームサーバは、私たちをこのネームサーバのどれかに指し向けます。どれかひとつをランダムに選んでみましょう。

```
$ 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
```

MIT.EDU のサーバ群がいっぺんに提示されました。ではまたどれかをランダムに選びましょう。

```
$ 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
```

今度は ”ANSWER SECTION” がありました。そして私たちの知りたかった答も見つかりました。 ”AUTHORITY SECTION” には、次回 ai.mit.edu に尋ねる際にはどのサーバにすべきか、という情報が含まれています。したがって次に ai.mit.edu の名前について知りたいときには、これらに直接聞けば良いわけです。 named は同時に mit.edu に関する情報も集めるので、次に例えば www.mit.edu が問い合わされたときには、答えにずっと近いところにいることになります。

というわけで、. からスタートし、参照先提示を辿ることで、ドメイン名の各レベルにおけるネームサーバを次々に見つけることができました。自前の DNS サーバがあれば、これらの他のネームサーバを使わなくても、あなたの named は、このように掘っていく段階で見つけた情報をすべてキャッシュし、しばらくは再び尋ねなくても良いようにしてくれます。

ツリーとのアナロジーでいうと、名前の各 “.” は枝分かれのポイントに対応します。そして “.” に挟まれた部分はツリー中でのそれぞれの枝の名前になります。欲しい名前 (prep.ai.it.edu) の名前を得るには、このツリーを昇っていくことになります。 root (.) や、 root から prep.ai.mit.edu に至る途中のあらゆるサーバに情報を問い合わせ、それらをキャッシュします。キャッシュの制限に達すると、この再帰的なレゾルバはそのサーバへの問合わせをやめ、そこで参照提示された、名前の端のほうにある次のサーバへと進んでいきます。

いままでほとんど触れませんでしたが、同じくらい非常に重要なドメインとして in-addr.arpa があります。これは「普通の」ドメインのようにネストもします。 in-addr.arpa のおかげで、アドレスがわかっている場合にホスト名を得ることができるようになります。ここで重要なのは、IP 番号は in-addr.arpa ドメインでは逆順に記述されることです。あるマシンのアドレス 192.186.203.77 がわかっていた場合、 named は先程の prep.ai.mit.edu の例と同じように 77.203.168.198.in-addr.arpa を探そうとします。いま例えば、 ‘.’ 以外全くマッチしないような、キャッシュにないエントリを探すとしましょう。 root サーバに訪ね、 m.root-servers.net は他の root サーバへの参照を返します。 b.root-servers.net は直接 bitsy.mit.edu/ への参照を返してくれるので、そこから情報を取得することになります。

5.2 自分のドメインを作る

さて、私たちのドメインを定義しましょう。ドメイン linux.bogus を作り、そこに私たちのマシンを定義しましょう。ここでは完全に架空のドメイン名を使って、間違っても外部の人に迷惑がかからないようにしましょう。

始める前にもう一点。ホスト名に使える文字には制限があります。英語のアルファベット a-z、数字 0-9、および '-' (ダッシュ) 文字だけが使えます。守るようにしてください (この規則を破っても BIND 9 では大丈夫ですが、BIND 8 はダメです)。大文字小文字は DNS では区別されません。したがって pat.uio.no と Pat.UiO.No とはまったく同じように解釈されます。

実はこの章で最初に行うべき部分はすでに記述済みです。named.conf には以下のような行がありますよね。

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

このファイルではドメイン名の最後に '.' を付けていない点に注意してください。上記の内容から、これから私たちはゾーン 0.0.127.in-addr.arpa を定義すること、そしてこの named がそのゾーンのマスターサーバになること、またその内容がファイル pz/127.0.0 に保存されることなどがわかります。このファイルはすでに設定済みで、以下のような内容のはずです。

```
$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.
```

先程の named.conf の場合とは対照的に、こちらのファイルではすべてのドメイン名の最後に '.' があることに注意してください。ゾーンファイルの先頭に $ORIGIN 命令を置くことを好む人たちもいるようですが、これは不要です。ゾーンファイルの origin (このゾーンが属する DNS の階層) は named.conf のゾーンセクションで指定されます。この場合は 0.0.127.in-addr.arpa です。

この「ゾーンファイル」には三つの「リソースレコード (resource record: RR)」が含まれています。SOA RR, NS RR, PTR RR です。SOA は Start Of Authority の省略です。'@' は特別な記号で、origin を意味します。このファイルの 'domain' カラムは 0.0.127.in-addr.arpa ですから、最初の行の実際の意味は以下と同じになります。

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

NS は Name Server RR の略です。この行の先頭には '@' がありません。これは暗黙のうちにすでに指定されたことになっています。直前の行が '@' ではじまっていたからです。多少タイプの量が節約できますね。したがって NS の行は以下のようにも記述できることになります。

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

この行は DNS に、どのマシンがこのドメイン 0.0.127.in-addr.arpa のネームサーバであるかを教えます。ns.linux.bogus というわけですね。'ns' というのはネームサーバに良く用いられる名前ですが、これは web サーバに www.*something* という名前が付けられるのと似たようなものです。実際にはどんな名前を用いてもかまいません。

最後に PTR (Domain Name Pointer) レコードが、サブネット 0.0.127.in-addr.arpa のアドレス 1 のホスト、すなわち 127.0.0.1 が localhost という名前であることを示しています。

SOA レコードはどんなゾーンファイルでも先頭に置かれます。また各ゾーンファイルにつき一つ、先頭に (ただし $TTL 指定のあとに) 書きます。このレコードはゾーンの説明です。どこから得られるのか (ns.linux.bogus というマシン)、内容に関する責任者は誰か (hostmaster@linux.bogus: ここにはあなたの電子メールアドレスを入れましょう)、ゾーンファイルのバージョンはいくつか (シリアル番号: 1)、その他キャッシュやセカンダリ DNS サーバなどに関連した内容などを書きます。残りのフィールド (refresh, retry, expire, minimum) については、この HOWTO の値をそのまま使えば特に問題ないでしょう。 SOA の前には必須の行、 $TTL 3D と書かれた行があります。これはすべてのゾーンファイルに書いてください。

では、ここで named を再起動 (rndc stop; named) して、 dig コマンドで今までの設定の確認を行いましょう。 -x を使うと逆引きの問合わせを行います。

```
$ 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
```

なんとか 127.0.0.1 から localhost が得られました。いい感じですね。ではメインのお仕事である linux.bogus ドメインのために、named.conf に新しい 'zone' セクションを書きましょう。

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

ここでも named.conf ファイルに記述するドメイン名の最後には '.' が付いていないことに注目。

linux.bogus ゾーンファイルには、まったく架空のデータを置くことにしましょう。

```
;
; 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
```

SOA レコードについては二つの点に注意する必要があります。ns.linux.bogus は A レコードを持った実際のマシンでなければなりません。CNAME レコードは、SOA レコードのサーバマシンの部分には記述できません。名前は 'ns' でなくても、正しいホスト名であればかまいません。次に hostmaster.linux.bogus は hostmaster@linux.bogus と読み替えてください。これはメールエイリアスかメールボックスで、この DNS をメンテナンスしている人が頻繁にチェックしているところでなければなりません。このドメインに関するメールは、ここで記述されたアドレスに送ることになっています。名前は 'hostmaster' でなくあなたの e-mail アドレスでもかまいません。でも 'hostmaster' でももちろんちゃんと動くはずです。

このファイルには新しいタイプの RR があります。MX (Mail eXchanger) RR です。これはメールシステムに対して someone@linux.bogus 宛メールの送り先を伝えるもので、mail.linux.bogus または mail.friend.bogus がこれになります。マシンの名前の前に書かれた数値は MX RR の優先度を示します。最小の数値 (10) を持つホストに対して優先的にメールが送られます。この配送に失敗すると、メールはより大きな数値を持つホストに配送されます。すなわちここでは優先度 20 を持つ mail.friend.bogus です。

rndc reload を実行して、named に設定ファイルを再び読ませます。ここまでの設定を dig で確認しましょう。

```
$ 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
```

よく見ると、バグがあることがわかると思います。

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

というのは全くおかしいですね。これは、

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

でなければなりません。

読者の学習効果を慮って :-)、ここで私はわざと間違えました。ゾーンファイルを見ると、以下の行があるはずです。

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

ここにはピリオドがないですね。あるいは余計に 'linux.bogus' を書いてしまっている、とも言えます。ゾーンファイルに書かれたホスト名の最後にピリオドがない場合には、origin が最後に加えられます。つまり linux.bogus.linux.bogus と二重になってしまうのです。ですから、

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

または

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

とするべきです。私は後者が好きです。タイプ量が少ないですから。 BIND の専門家にはこの書式に反対する人もいます (賛成する人もいます)。ゾーンファイルでは、ドメインはすべて書き下して '.' で終えるか、全く書かないかどちらかにします。後者ではデフォルトで origin が付属します。

ひとつ強く注意しておきたいのですが、 named.conf ファイルでは、ドメイン名の後に '.' を付けてはいけません。'.' が多すぎたり少なすぎたりしたおかげで、どれだけ多くの物事がだめになり、人々が混乱させられたか、きっとあなたには想像もつかないでしょう。

では、この点を押さえて新たなゾーンファイルを書きましょう。少々新しい情報も加わっていますが、以下のようになります。

```
;
; 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) は、各マシンを複数の名前で呼ぶ方法です。よって www は ns の別名になります。 CNAME レコードの利用については、多少議論の余地があります。でも以下のルールを守っておけば大丈夫でしょう。 MX, CNAME, SOA の各レコードでは CNAME レコードを参照してはいけません。これらは A レコードだけを参照すべきなのです。したがって

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

という指定はすべきではなく、

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

という指定が正しいものとなります。

rndc reload を実行して新しいデータベースをロードしましょう。すると named がファイルを読み込み直します。

```
$ 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. 199802151 28800 72
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. 199802151 28800 72
;; 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
```

うまくいっていますね。ご覧の通り、ゾーンファイルそのものとちょっと似ています。 www だけについても調べてみましょう。

```
$ 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
```

つまり www.linux.bogus の本当の名前は ns.linux.bogus なわけです。そして named が ns について持っている情報も示してくれています。あなたがプログラムなら、この情報で接続できるはずです。

さて、ここまでが半分。

5.3 逆引きゾーン

今やプログラムは、 linux.bogus にある名前を、実際に接続すべきアドレスに変換できるようになったわけです。でも逆引きのゾーンも必要です。これは DNS でアドレスを名前に変換できるようにするためのものです。この名前はさまざまな種類のたくさんのサーバ (FTP, IRC, WWW などなど) において、あなたとの通信を認めるか、また認めた場合、どの程度の優先性を付与するかなどの判断に用いられます。インターネットにあるサービスすべてにアクセスするためには、逆引きのゾーンが必要になります。

以下を named.conf に記述してください。

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

これは 0.0.127.in-addr.arpa とまったく同じです。ファイルの中身も同じようになります。

```
$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.
```

では rndc reload を実行し、 named に設定ファイルを再び読ませ、再び dig でこれまでの設定を確認しましょう。

```
$ 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
```

うん、良さそうですね。全体もダンプして調べてみましょう。

```
$ 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
```

よさそうですね！このような出力にならなかった場合は、 syslog にエラーメッセージが出ていないか見てみましょう。やり方は 3.1 (named を起動する) 直下の最初のセクションで説明しましたね。

5.4 気をつけてほしいこと

ここでいくつか付け加えておくことがあります。上記で用いた IP 番号は 'private net' のうちの一つのブロックから取ってきたものです。つまりこれらの IP 番号はインターネットでパブリックに用いることはできません。ですからこの HOWTO で例として表示しても安全なわけです。次の点は `notify no;` の行です。これは named に対して、「ゾーンファイルのどれかが更新されても、それをセカンダリ (スレーブ) サーバに伝えない」という指示をすることになります。 BIND 8 以降の named は、ゾーンファイルの NS レコードにリストされている他のサーバに、ゾーンの更新を知らせることができます。これは通常は便利な機能ですが、プライベートな実験ではこの機能は off にしておきましょう。この実験によってインターネットに迷惑をかけたくはないでしょう？

そしてもちろん、このドメインは架空のいいかげんなもので、使われているアドレスも同じく架空のものです。現実の世界で用いられている本物の例は、次の章を見て下さい。

5.5 なぜ逆引きが動作しないのか

名前引きのシステムには、ちょっとした「できの悪い部分」がいくつかあります。通常これらが表に出てくることはありませんが、逆引きゾーンの設定では良くお目にかかることがあります。ここから以降を読み進める前には、あなたのマシンが「あなたのネームサーバ」から逆引きできることを確認してください。できない場合は戻ってやり直してからにしてください。

ここでは、逆引きを外部ネットワークから見た場合に生じやすい二つの問題点について議論します。

5.5.1 逆引きゾーンが代理されない

サービスプロバイダからネットワークアドレス空間とドメインネームをもらうときには、通常そのドメインネームは代理 (delegation) されます。代理とは橋渡しの役目をする NS レコードのことで、あるネームサーバから別のネームサーバを取得するときに用います。先に 5.1 (退屈な理論) の節で説明しました。読んでます、よね？逆引きゾーンが動作していない場合は、今すぐ戻って読んでください。

逆引きゾーンにも代理が必要です。例えば `192.168.196` のネットワークを `linux.bogus` ドメインと一緒にプロバイダからもらったとしたら、プロバイダには `NS` レコードを正引きゾーンだけでなく逆引きゾーンにも加えてもらう必要があります。 `in-addr.arpa` からあなたのネットワークまでの繋がりを辿っていくと、おそらくどこ

かで鎖の輪が切れていることでしょう。多分接続しているサービスプロバイダで。「切れている輪」が見付かったら、サービスプロバイダに連絡してエラーを修正してもらいましょう。

5.5.2 クラスレス (classless) のサブネットをもらった場合

これはやや高度な話題になります。しかしクラスレスのサブネットは最近非常に良く使われるようになってきたので、小さな会社に所属している人なら、おそらく身近にあるでしょう。

最近のインターネットをなんとか維持できているのは、実はクラスレスサブネットのおかげなのです。数年前に IP 番号の枯渇についてちょっとした騒ぎになったことがありました。その時 IETF (Internet Engineering Task Force: インターネットがちゃんと動いているのは彼らのおかげなのです) の賢人たちは、彼らの叡智を集めてこの問題を解決したのでした。ただし相応の対価をもって。その対価の一部は、“C” 未満のサブネットを使わなければならないこと、それによって動作しなくなるものが出てくること、です。このあたりに関する説明と、その扱い方に関しては、

Ask Mr. DNS `http://www.acmebw.com/askmrdns/00007.htm`

にある優れた解説を見てください。

読みました？ここでは説明しませんから、ちゃんと読んでくださいね。

この問題の半分は、接続先の ISP が Mr. DNS に書いてあったテクニックを理解していなければならない、というところにあります。小さな ISP では、これを知らずに動かしているところもあるでしょう。その場合は、あなたが彼らにがまん強く教えてあげなければいけません。それに、まずあなたが理解しないといけませんね ;-) 理解してくれたら、きっとちゃんとした逆引きゾーンを設定してくれるでしょう。 dig を使って正しいかどうか確かめましょう。

問題の残り半分は、あなたがこのテクニックを理解しなければならない、というところです。自信がなければ、もう一度読みにいきましょう。そして Mr. DNS の説明にしたがって、自分のクラスレス逆引きゾーンを設定しましょう。

実はここにはもう一つトラップが待ち構えています。(非常に) 古いレゾルバは、名前解決のチェーンの中に置かれたこの `CNAME` トリックの部分をたどることができず、あなたのマシンの逆引きに失敗してしまうことがあります。この結果、そのレゾルバは正しくないアクセスクラスを返したり、アクセスを拒否したり、とにかくそんなようなことになります。この問題に引っかかってしまったら、(私の知るかぎりでは) 接続先の ISP に頼むしかありません。トリックを使ったクラスレスゾーンファイルに、 CNAME の代わりにあなたの PTR レコードを直接書き込んでもらうことになります。

ISP によっては別の解法を提供していることもあります。たとえば Web ベースの form によって逆引きのマップを入力できるようになっているとか、あるいは似たような全自動型登録システムとか。

5.6 スレーブサーバ

マスターサーバでゾーンが正しく設定できたら、少なくとも 1 台のスレーブサーバが必要になります。スレーブサーバはシステムを堅牢にするために必要なものです。マスターが落ちても、ネットにいる外部の人が、スレーブからあなたのドメインに関する情報を取得できるようになるのです。スレーブは、あなたのいるところからできるだけ離れたところに置きます。マスターとスレーブは、電力供給源・ LAN・ ISP・町・国、などを、できる限り共有していないことが望ましいのです。これらがすべてマスターと異なっているスレーブが見つかったら、それは非常に良いスレーブだと言えます。

スレーブは、単にマスターからゾーンファイルをコピーするネームサーバです。以下のように設定します。

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

データのコピーにはゾーン転送という仕組みを用います。ゾーン転送は SOA レコードで制御します。

```
@       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
```

マスターのシリアル番号がスレーブよりも大きいときに限ってゾーンが転送されます。リフレッシュ (refresh) 時間に一回ずつ、スレーブはマスターが更新されていないかどうかチェックします。チェックできない (マスターに接続できない) と、スレーブはリトライ (retry) 時間に一回ずつ再接続を試みます。期限切れ (expire) 時間が経過しても失敗し続けた場合は、スレーブはそのゾーンをファイルシステムから削除し、それ以上はゾーン情報の提供を行わなくなります。

6 基本的なセキュリティオプション

By Jamie Norrish

問題を避けるためのオプション設定

いくつか簡単な作業を行えば、サーバをより安全にでき、またサーバの負荷を低減できます。ここで紹介する内容は出発点に過ぎません。セキュリティのことを考えるなら (考えるべきです)、ネット上にある他のリソースにあたってください (11 (最後の章) をご覧ください)。

以下の指定は named.conf に行います。これらの指定をこのファイルの options の内部に書くと、このファイルでリストされたすべてのゾーンに適用されます。特定の zone エントリの内部に書くと、そのゾーンだけに適用されます。 zone 内部に書かれたエントリは options に書かれたエントリよりも優先されます。

6.1 ゾーン転送の制限

スレーブサーバがドメインに対する問合わせに応えるには、プライマリサーバからゾーンの情報を転送してくる必要があります。しかしスレーブサーバ以外のホストには、この転送の必要はないはずです。ですからゾーン転送は allow-transfer オプションを使って制限しましょう。例えば ns.friend.bogus の IP アドレスである 192.168.1.4 と、それからデバッグ用の自分自身を追加するならば:

```
zone "linux.bogus" {
      allow-transfer { 192.168.1.4; localhost; };
};
```

ゾーン転送を制限すれば、外部の人々から見えるのは、彼らが直接尋ねたホストに関する内容だけに限られます。DNS 設定の詳細全体を問合わせることはできなくなるのです。

6.2 不正利用から守る

まず、内部ネットワークとローカルのマシンからのものをのぞき、あなたの管理するドメイン以外への問合わせは禁止しましょう。これは、悪意を持ってあなたの DNS サーバを利用しようとする試みを禁止するだけでなく、本来不必要な問合わせを減らします。

```
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; };
};
```

さらに内部 / ローカルからのものを除き、再帰的な問合わせも禁止します。これによりキャッシュ汚染攻撃 (cache poisoning attack: 間違ったデータをサーバに送りつけること) の危険性が減らせます。

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

6.3 named を root 以外で実行する

named を root 以外から実行するのは良い考えです。破られたときに、クラッカーに奪われる権限を減らすことが出来ますから。まず named を動作させるユーザを作り、次に named を起動している init スクリプトを修正します。新しく作ったユーザ名を、 named の -u フラグに指定します。

例えば Debian GNU/Linux 2.2 なら、 `/etc/init.d/bind` スクリプトを以下の行のように修正します (ユーザ `named` はあらかじめ作成しておきます):

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

Red Hat や他のディストリビューションでも同様にできるはずです。

Dave Lugo は、二つの chroot を用いたセキュアな設定を

`http://www.etherboy.com/dns/chrootdns.html`

で解説しています。きっと興味を持たれる読者が多いでしょう。これを用いれば named を動かしているホストをさらに安全にできます。

7 実際のドメインの例

実際に用いられているゾーンファイルの例

チュートリアルの例だけでなく実際に動作している例を載せて欲しい、という意見があったので、この章を設けました。

この例は LAND-5 の David Bullock の許可の下に用いています。これらのファイルは、 1996 年 9 月 24 日現在のものを、私が BIND 9 の制限と拡張にあわせて編集したものです。したがってここでの記述は、実際に LAND-5 のネームサーバに問い合わせを行った結果とは多少異なります。

7.1 /etc/named.conf (または /var/named/named.conf)

マスターゾーンセクションとして、必須の逆引きゾーンが二つ書かれています。 127.0.0 のネットと LAND-5 のサブネットである `206.6.177` です。 LAND-5 の正引きゾーンである `land-5.com` もプライマリとして指定されています。ゾーンファイルは本 HOWTO のこれまでの例で用いていた `pz` ではなく、 `zone` というディレクトリに収められていることにも注意してください。

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

このファイルをあなたの named.conf ファイルに用いるときには、**必ず** “`notify no;`” を `land-5` の二つの zone セクションに追加して、事故が起こらないようにしてください。

7.2 /var/named/root.hints

このファイルは動的に変化するものですから、このリストは古いです。以前に説明したようにして、新しく作ったものを使いましょう。

```
; <<>> 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

非常にシンプルなものです。まず絶対に必要な SOA レコード、そして 127.0.0.1 を localhost にマップするレコードです。これらは両方とも必須です。逆にこれ以上のものは置くべきではありません。このファイルは、使っているネームサーバか hostmaster のメールアドレスが変更されない限り、更新する必要はおそらくないでしょう。

```
$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.
```

適当にインストールされた BIND では、ここでの例のように $TTL の行がないかもしれません。この行は以前は用いられておらず、8.2 の BIND だけが起動時にこの行が無い旨の警告を出します。なお BIND 9 では $TTL は必須です。

7.4 /var/named/zone/land-5.com

まず必須である SOA レコードと、同じく必須の NS レコードがあります。セカンダリのネームサーバが ns2.psi.net に用意されていることもわかりますね。これは望ましい設定です。必ずサイトの外にバックアップ

のセカンダリネームサーバを置くべきです。マスターのホストは land-5 で、このホストは同時に各種のインターネットサービスを提供していることもわかります。これには (A レコードでなく) CNAME が用いられています。

SOA レコードからわかるように、このゾーンファイルは land-5.com を origin にしており、連絡担当者は root@land-5.com です。hostmaster も担当者のアドレスとして良く用いられます。シリアル番号は yyyymmdd 形式で、その日のうちのシリアル番号が追加されています。これはきっと 1996 年 9 月 20 日の第 6 版なのでしょう。シリアル番号は必ず増加しなければならないことを思い出してください。ここには当日中のシリアル番号として一桁しか使うことができません。したがって 9 回変更を行ったら、次の変更を行うには翌日まで待たなければなりません。二桁使う方が良いかもしれませんね。

```
$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
```

land-5 のネームサーバを試してみればわかりますが、本当のホスト名は `ws_`*number* となっています。 BIND 4 の後の方のバージョンから、ホスト名に用いることのできる文字が制限されるようになりました。したがってこの名前は BIND 8 では全く動作しませんから、この HOWTO に掲載する際には '_' (underline) を '-' (dash) で置き換えました。しかし、先に述べたように、 BIND 9 では再びこの制限はなくなりました。

もう一つ気がつきましたか？各ワークステーションには固別の名前は付いておらず、プレフィックスに IP 番号の最後の二つが付いた形式になっています。このような命名方法を用いればメンテナンスはとても楽になりますが、やや人間との相性は悪いので、顧客をイライラさせる結果になってしまうかもしれません。

`funn.land-5.com` も `land-5.com` のエイリアスになっていますが、これは CNAME レコードではなく A レコードを用いています。

7.5 /var/named/zone/206.6.177

このファイルについては後でコメントします。

```
$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.
```

逆引きのゾーンは、設定の中でも多くの悲劇を引き起こす部分と言えます。これはマシンの IP 番号がわかっている場合に、ホスト名を取得するために用いられます。例えば、あなたが立てている FTP サーバが FTP クライアントから接続されたとしましょう。あなたの FTP サーバはノルウェーにあるので、ノルウェーと他のスカ

ンジナビアの国々以外からの接続は多めに、他の国々からの接続は少なめに制限したいとします。クライアントから接続されると、C ライブラリによって接続してきたマシンの IP 番号を知ることができます。なぜならクライアントの IP 番号は、ネットワークを運ばれてきた IP パケットのそれぞれに書き込まれているからです。ここで gethostbyaddr という関数を呼べば、IP 番号からホストの名前を引くことができます。gethostbyaddr は DNS サーバに尋ね、DNS サーバは DNS からそのマシンを探します。接続してきたクライアントは ws-177200.land-5.com だったとしてみましょう。C ライブラリが IRC サーバに渡す IP 番号は 206.6.177.200 となります。したがって名前を引くためには `200.177.6.206.in-addr.arpa` を見つける必要があります。DNS サーバはまず `arpa.` のサーバに問い合わせをし、`in-addr.arpa.` のサーバを教えてもらいます。続いて 206, 6 を順次逆に辿って、最後に Land-5 のゾーンである `177.6.206.in-addr.arpa` ゾーンを発見します。最後にサーバは、そこから `200.177.6.206.in-addr.arpa` に対する答えを入手します。“`PTR ws-177200.land-5.com`” レコードから、`206.6.177.200` は `ws-177200.land-5.com` であることがわかります。

FTP サーバはスカンジナビアの国々、すなわち `*.no`, `*.se`, `*.dk` からの接続を優先しますが、`ws-177200.land-5.com` は明らかに以上のどれにもマッチしませんから、サーバはこの節続を、バンド幅がより小さく、最大接続数も少ないクラスに割り当てます。`206.2.177.200` に対する逆引きマップがそもそも `in-addr.arpa` ゾーンに存在しなければ、サーバは決して名前を見つけることができませんから、`206.2.177.200` そのものを `*.no`, `*.se`, `*.dk` と比較します。どれにもマッチするわけはなく、サーバはクラスの割り当てができないその節続を、拒否することもあり得ます。

逆引きマップが重要なのはサーバだけだ、という人や、そもそも逆引きマップなんて全然大事じゃないんだ、なんていう人がいるかもしれません。これは間違いです。多くの ftp, news, IRC サーバでは逆引きのできないマシンからの接続を拒否します (WWW サーバにさえ拒否するものもあります)。ですからマシン名の逆引きマップは実のところは必須なのです。

8 メンテナンス

動作を維持するために

named には、ただ走らせる以外にも一つ保守作業があります。`root.hints` ファイルを最新の状態に保つ作業です。一番簡単なのは `dig` を使うやり方です。まず引き数なしで `dig` を動かすと、現在サーバで使っている `root.hints` の内容が表示されます。次にリストされているルートサーバのいずれかに対して `dig @rootserver` のように問い合わせを行います。出力結果は `root.hints` の内容にとてもよく似ているはずです。この結果を `dig @e.root-servers.net . ns > root.cache.new` のように保存して、古い `root.hints` と置き換えます。

キャッシュファイルを入れ替えた後には named に再読み込みさせるのをお忘れなく。

Al Longyear がスクリプトを送ってくれました。自動的に `root.hints` を更新してくれるものです。これを月に一度起動する crontab のエントリをインストールすれば、後は全部おまかせです。スクリプトでは、メールがちゃんと動作していて、メールエイリアスとして ‘hostmaster’ が定義されていることを前提としています。あなたの設定にあわせてハックする必要があります。

```
#!/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
```

訳注: 訳者はまだ BIND 8 なのでこのスクリプトを試していないのですが、`rndc restart` というコマンドは `rndc stop; named` で置き換えないといけないような気がします。

`root.hints` は Internic から ftp でも入手できる、と言うことをすでにご存じの方もいるかもしれません。ですが `root.hints` の更新に ftp は使わないようにしてください。上記の方法のほうが、ずっと「ネット (と Internic) に優しい」のです。

9 BIND 9 に移行する

BIND 9 の配布アーカイブや、パッケージ化されたバージョンには、`migration` という文書が含まれており、そこに BIND 8 から BIND 9 に移行するための情報が記述されています。この文書は非常にわかりやすく書かれています。バイナリパッケージをインストールした場合は、`/usr/share/doc/bind*` や `/usr/doc/bind*` あたりに置かれていると思います。

BIND 4 を使っている人は、同じ場所にある `migration-4to9` を見てください。

10 Q & A

私にメールする前に、まずこの章を読んでください。

1. 私の named では named.boot ファイルが必要と言われます

   読んでいる HOWTO が間違っています。この HOWTO の古い版では bind 4 のことを扱っていますので、そちらを読んでください。

   `http://langfeldt.net/DNS-HOWTO/` にあります。

2. ファイアウォールの中で DNS を使うには？

   ヒント。`forward only;`。また

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

が named.conf ファイルの “options” の部分に必要になるでしょう。3 (キャッシュ専用のネームサーバ) の節にある例でちょっと触れましたね。

3. DNS によって、あるサービスに対するアドレスを順繰りにまわす (round-robin する) にはどうすれば良いですか？つまり例えば www.busy.site に対する負荷を分散させるようにするにはどうすれば良いでしょうか。

   www.busy.site に対する **A** レコードを複数用意して、4.9.3 以降の BIND を用いましょう。BIND は回答を round-robin してくれます。古い版の BIND では、これは動作しません。

4. (クローズな) イントラネットで DNS を使いたいのです。どうすれば良いですか？

   root.hints ファイルを使わないようにして、ゾーンファイルだけを使いましょう。root.hints ファイルをいちいち更新する必要もないわけです。

5. セカンダリ (スレーブ) のネームサーバを設定するには？

   プライマリ（マスタ）のサーバが、アドレス 127.0.0.1 だったとして、以下のような行をセカンダリの named.conf に記述します。

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

   zone 情報を取ってこれるマスタサーバが他にもある場合は、masters リストに ‘;’ (セミコロン) で区切って追加することもできます。

6. net から切断されているときにも BIND を動作させておきたいんですが。

   これに関連した記事を 4 つ紹介しましょう。

   - BIND 8/9 に特化したやり方を Adam L Rice が電子メールで教えてくれました。ダイアルアップのマシンで DNS を手間をかけずに動作させる方法です。

     > 私は、最近のバージョンの BIND では、これ [編注: ファイルを切り
     > 替える] がもう不必要であることに気がつきました。 "forwarders"
     > 指定の他に "forward" 指定が可能になっていて、後者で前者の使われ方を
     > 制御できるようになっていたんです。デフォルトの設定は
     > "forward first" で、 最初にそれぞれの forwarders に問い合わせを行い、

失敗した場合にはじめて自分自身で聞き込み調査を始めます。これが
ラインが切れている時に gethostbyname() にやたらと時間がかかって
しまう、おなじみの振る舞いです。しかし "forward only" を設定して
おくと、 BIND は forwarders から反応が帰ってこないとすぐに
あきらめます。したがって gethostbyname() も速やかに返ってくる
ことになります。ですから技巧を使って /etc のファイルを切り替え、
サーバを再起動する必要はないのです。

私の場合では、以下の行を named.conf ファイルの options { }
セクションに追加するだけでした。

```
forward only;
forwarders { 193.133.58.5; };
```

とってもうまく動作してます。この方法のただ一つの欠点は、非常に
洗練された DNS ソフトウェアを、キャッシュ動作だけしかしない
単機能なソフトにしてしまう、ということです。ただ DNS キャッシュ
だけをするソフトがあれば私は実はそっちを使いたいんですけど、
Linux ではそのようなソフトはないみたいですね。

- 以下の記事は Ian Clard <ic@deakin.edu.au> からもらったメールです。彼のやり方を説明してくれています。

IP マスカレードをさせている手元のマシンで named を走らせています。
root.hints ファイルを二つ用意します。一つは root.hints.real で、
本物の root サーバの名前が書かれています。もう一つは
root.hints.fake で、その内容は...

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

です。切断するときには root.hints.fake ファイルを root.hints に
コピーして named を再起動します。

接続するときには root.hints.real ファイルを root.hints にコピー
して named を再起動します。

これらは ip-down と ip-up でそれぞれ自動実行させています。

オフラインの時にドメイン名に対する問い合わせを行うと、named は
それらに付いて知りませんから、以下のようなエントリを messages に
出力します。

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

これは気にしなくてもかまいません。

私のところではこれで全く問題なく動作しています。ネットから切断
されているときは、ローカルマシンのネームサーバを外部のドメイン名に
対するタイムアウトの待ち時間なしで使えますし、接続されているとき
には外部のドメインに対する問い合わせを普通に行うことができています。

しかし、 Peter Denison は Ian のやり方がまだ充分でないと教えてくれました。彼のメッセージによると:

オンライン時）キャッシュされたエントリ（とローカルネットのエントリ）は
ただちに提供する。キャッシュされていないエントリについては、
自分の ISP のネームサーバにフォワードする。
オフライン時）ローカルネットワーク関連の問合わせはただちに提供する。
その他の問合わせについては ** ただちに ** 失敗する。

root キャッシュファイルの変更と、問合わせのフォワードとの組み合わせは
うまく動作しません。

そこで、私は二つの named を（地域 LUG で議論しながら）以下のように
設定しました。

named-online: ISP のネームサーバへフォワード
localnet ゾーンのマスター
localnet の逆引きゾーン（1.168.192.in-addr.arpa）のマスター
0.0.127.in-addr.arpa のマスター
ポート 60053 で待機

named-offline: フォワードを行わない
root キャッシュファイルは「にせもの」にする
3 つのローカルゾーンのスレーブ（マスターは 127.0.0.1:60053）
ポート 61053 で待機

そしてこれをポートフォワードと組み合わせ、ポート 53 をオフラインの時には

61053 に、オンラインの時には 60053 にフォワードします（私は 2.3.18 で
新しい netfilter パッケージを使いましたが、以前の（ipchains）の機構でも
動作するはずです。

ただしこれはマシンの外側からの問合わせには動作しません。 BIND 8.2 には
小さなバグがあって、スレーブをマスターと同じ IP アドレスでは（ポートが
異なっても）同時に動作できないからです（開発者には知らせました)。
明らかなパッチなので、おそらくすぐに直るでしょう。

- 切断されている時間の長いマシンにおいて、 BIND が NFS やポートマッパとどのように相互作用するのかに関する情報もいただきました。 Karl-Max Wanger からです。

  インターネットに対してモデム経由でたまにしか接続しないマシンには、
  私はすべて named を走らせていました。ネームサーバはキャッシュと
  してのみ動作し、 authority をもつ zone は保有せず、すべてを
  root.cache ファイルに書かれたネームサーバに問い合わせに行く設定に
  していました。 Slackware の流儀に従い、 named は nfsd や mountd の
  前に起動していました。

  マシンのうちの一つ（Libretto 30 notebook）で、問題が起こりました。
  私のローカルな LAN につながっている他のマシンから、そのマシンを
  mount できなくなってしまうのです（ごくたまにできる時もありますが)。
  これは接続形式に依存せず、 PLIP でも PCMCIA のイーサネットカードでも、
  シリアル経由の PPP でも同じように起こりました。

  しばらく実験と考察を行った後、以下のような結論に達しました。
  nfsd と mountd が起動時に portmapper に対して行った登録動作
  (私はこれらのデーモンを、通常通りブート時にスタートしていました)
  を、 named はめちゃめちゃにしてしまうのです。 named の起動を
  nfsd と mountd のあとに行うようにしたところ、この問題は完全に
  解決しました。

  ブートの順序をこのように変更することによる不利はまったくありません
  から、潜在的な問題を避けるために、このようにすることをすべての
  皆さんにお薦めしたいと思います。

7. キャッシュネームサーバはどこにキャッシュを保存しているの？キャッシュのサイズは制御できますか？

   キャッシュはすべてメモリに保管されています。ディスクに書き込まれることはまったくありません。named を kill すると、キャッシュも失われます。キャッシュを制御する方法はありません。named のキャッシュ管理は単純なルールに従っているからです。キャッシュそのものも、あるいはキャッシュのサイズも、どんな理由があれ制御できません。この点を「修正」したければ named をハックしても良いでしょ

う。おすすめはできませんが。

8. named は再起動されるときにキャッシュを保存してくれますか？保存するようにできますか？

   いいえ、 named は終了時にキャッシュを保存しません。つまり named を kill して再起動するたびに、キャッシュはゼロから再構成されます。キャッシュをファイルに保存するように named に指示する方法はないのです。この点を「修正」したければ named をハックしても良いでしょう。おすすめはできませんが。

9. ドメインを手に入れるにはどうすればいいですか？私は (例えば) `linux-rules.net` というドメインを立ち上げたいのですが、このドメインを割り当ててもらうにはどうすればいいのでしょうか。

   ネットワークサービスプロバイダに連絡してみれば、おそらく助けてもらえるでしょう。なお世界のほとんどの地域では、ドメインの入手にはお金が必要であるはずです。

10. DNS サーバを安全にするにはどうすればいいでしょう？ split DNS の設定のしかたは？

    両方とも高度な話題になります。いずれも `http://www.etherboy.com/dns/chrootdns.html`

    で取り上げられています。この話題は、これ以上ここでは扱いません。

11 より熟練した DNS 管理者になるために

文献とツール

しっかりした文献がちゃんと存在しています。オンラインのものと印刷されているものとがそれぞれあります。即席 DNS 管理者が熟練した DNS 管理者になるためのステップを踏むには、この中のいくつかを読むことが必要です。

私は *The Concise Guide to DNS and BIND* (by Nicolai Langfeldt, Que, ISBN 0-7897-2273-9) を書きました。この本はこの HOWTO と、とても似ていますが、多少詳細に、そしてずっと幅広い話題を扱っています。この本はポーランド語に翻訳され、 Helion から *DNS i BIND* として出版されています。(`http://helion.pl/ksiazki/dnsbin.htm` , ISBN 83-7197-446-9) C. Liu P. Albitz が書いた *DNS and BIND* は、今や第四版となりました (O'Reilly & Associates, ISBN 0-937175-82-X. バッタ本として知られています)。また *Linux DNS Server Administration* という本が Craig Hunt によって書かれ、 Sybex から出版されています (ISBN 0782127363)。私はこれはまだ読んでいません。良い DNS (その他なんでも) の管理者になるためには、 Robert M. Pirsig の *Zen and the Art of Motorcycle Maintenance* も必読でしょう。

訳注: Langfeldt さんの本の日本語訳は、オーム社から

『*DNS & BIND* 入門』 `http://www.ohmsha.co.jp/data/books/contents/4-274-06421-2.htm`

というタイトルで出版されています。オライリーの『DNS & BIND』の日本語版は、現在

第 *3* 版 `http://www.oreilly.co.jp/BOOK/dns3/`

が出版されており、第 4 版も近々に発刊予定とのことです。

オンラインでは、私の本 (やその他の大量の本) を電子的に購読するサービスが

`http://safari.informit.com/` にあります。

`http://www.dns.net/dnsrd` (DNS Resources Directory) や `http://www.isc.org/bind.html` でもいろいろ見つかります。 FAQ、リファレンスマニュアル、論文やプロトコル定義や DNS のハックもあります (これらや、以下に示す RFC の (全部ではないにせよ) ほとんどは、 BIND の配布アーカイブに含まれています)。私はこのあたりのほとんどは読んでいません。ニュースグループ *comp.protocols.tcp-ip.domains* では DNS の議論をしています。また DNS に関する RFC もたくさん存在しています。中でも重要なものを以下に挙げておきます。 BCP (Best Current Practice) の番号が付いているものは必読です。

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.