DNSサーバの設定
～基礎から学ぶ正しいDNS～

Internet Week 2004 チュートリアル
株式会社 IRIコミュニケーションズ
伊藤 高一
kohi@iri.co.jp

そもそもDNSとは

～introduction～

そもそもDNSとは

- http://internetweek.jp/
  - 人間に優しい
- http://210.199.223.86/
  - 計算機に優しい
  - 人間には優しくない
- http://[3ffe:504:100:2ff:1234:5678:fedc:ba98]/
  - もっと人間に優しくない
- せっかく計算機を使っているんだから、計算機から人間に歩み寄って欲しいよね。

そもそもDNSとは(続き)

- 計算機に歩み寄ってもらうには?
- ホスト名<->IPアドレスなどを変換する仕掛けがあればいい。
- DNSはその解の1つ。
  - 小規模なイントラネットなどの閉じた環境ではNISなど別の解もある。
  - WorldWideが相手だと、事実上、唯一の解。

DNSの特徴

- WWW、Emailなどさまざまなネットワークアプリケーションの動作基盤。
  - OSIの人はわざわざアプリケーション層と分けてプレゼンテーション層という名前をつけちゃうぐらい重要。
- 設定はあんまり簡単とは言えない。
  - 対/etc/hosts比
- slaveや上位ゾーンのサーバなどと連携が必要。
  - 機械同士だけではなく管理者同士も。
- 設定が多少おかしくても、なんとなくそれっぽく動いてしまう。
  - でも何かの拍子にボロが出る!

このチュートリアル

- このチュートリアルが目指すこと
  - the InternetにおけるDNSの質の向上。
  - 初級クラスの運用者の中級へのステップアップ。
- そのために何をするか?
  - 実用上、必要な範囲に限定してDNSの機構、BINDの設定を復習/再確認する。
- 対象受講者
  - DNSサーバを設定/運用しているが、自分の設定に今ひとつ自信がない方。
  - DNSサーバの設定/運用を手がける予定の方。
  - 初学者/中級以上の運用者は対象としない。

agenda

◆ そもそもDNSとは〜introduction〜(done)
[We're Here!]
◆ DNSの機構の復習
◆ BINDを使ったネームサーバの基本的な設定
◆ BIND9を中心に説明します。
◆ BIND8については相違点を簡単に補足するに留めます。
◆ DNSの運用
◆ Advanced topics

今回、お話「しない」こと

◆ DNSSEC、TSIG、IDN、Dynamic updateなどの高度な機能
◆ BIND以外のサーバ、UNIX以外のプラットフォームの設定
◆ Internet Registryへの手続き
◆ ドメイン名に関するpolitics

DNSの機構の復習

top downに見たDNS

- Domain Name System
- 何のためのもの?
  - ホスト名<->IPアドレスなどの変換。
    - アクセスの利便性
    - リナンバーなどの隠蔽
      - Layer6がLayer3アドレス(の変更)を隠蔽
- キーワードでDNSを語ってみると…
  - 階層型ドメインに基づく分散データベース。

top downに見たDNS(続き)

- 階層型ドメイン
  - '.'で区切られた名前。
  - *ホスト名.サブドメイン名.ドメイン名*
    という親子構造。
- 分散データベース
  - ドメインを基にしたゾーンという単位に名前空間を分割してデータを管理。
    - 個々のゾーンは一元管理。
    - ゾーンの切れ目では親から子にauthorityを委任。
    - 全体としては1つの名前空間を形成。

ゾーンとは

- ゾーン
  - 純技術的視点ではauthorityを委任する単位。
  - 運用の視点では管理の単位。
    - ゾーンは自律的な管理が及ぶ範囲で区切られるべき。
  - サブドメインはゾーンの境界になりうる。
  - が、すべてのサブドメインが独立したゾーンに(なる|しなければならない)わけではない。

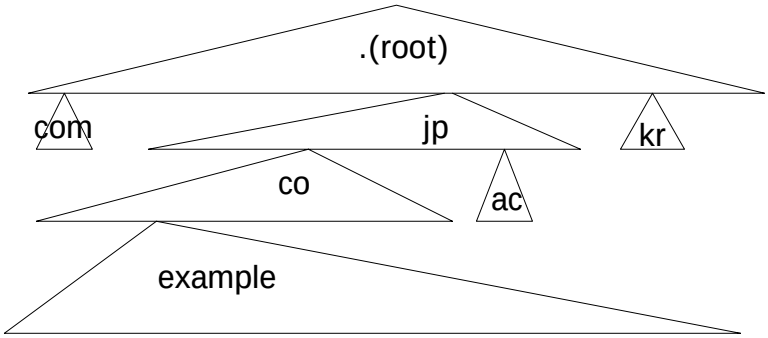
ゾーンとは(続き)

- authority
  - 親のゾーンと子のゾーンが連携するための仕組み。
    - 親ゾーンのネームサーバから子ゾーンのネームサーバに子ゾーンのauthorityが委任される。
    - 「○○ゾーンのauthorityは××だ。」
  - そのゾーンについてWorldWideからの問い合わせが振り向けられる。
    - データを一元的に自律管理できる。
    - ちゃんと面倒を見なければいけない。

ゾーンとは(続き)

- 例えば
  - jp.は1つのゾーン。.(root)からJPRSにauthorityを委任されている。
  - ad.jp.も1つのゾーン。jp.ゾーンからauthorityを委任されている。
  - nic.ad.jp.も1つのゾーン。ad.jp.ゾーンからauthorityを委任されている。
  - internetweek.jp.も1つのゾーン。jp.ゾーンからauthorityを委任されている。

ゾーンとは(続き)



thomas edward henry gordon james percy

ゾーンとは(続き)

- ゾーンの中身
  - リソースレコード(RR)
    - ホスト名->IPアドレス(A,AAAA)
    - IPアドレス->ホスト名(PTR)
    - メールサーバ(MX)
    - データの鮮度や賞味期限など(SOA)
    - authorityの所在(NS)
    - alias(CNAME)

  など。

ゾーンとは(続き)

◆ゾーンのデータの例

```
example.co.jp. IN SOA thomas.example.co.jp. root.example.jp.(
                      2004120201
                      3600
                      1200
                      3600000
                      900)
            IN  NS    thomas.example.co.jp.
thomas      IN  A     172.16.7.153
```


query

- query
  - リソースレコードの値を問い合わせること。
- recursive query
  - 目的のリソースレコードの値を要求する。
- non-recursive query
  - 目的のリソースレコードに到達するためのauthorityの委任先を要求する。
  - 木構造の枝分かれを1段1段たどる。
    - 最終的には目的のリソースレコードに行き当たる。
    - あるいは存在しないことが明らかになる。

検索の流れ

- resolverルーチン
  - アプリケーションプログラムがネームサーバにqueryするためのライブラリルーチン。
  - 一般には特定のネームサーバに対して目的の名前をrecursive queryする。
    - UNIXでは/etc/resolv.confで指定。
    - MacOSやWindowsにも相当する設定あり。
    - 知らないうちにDHCPやIPCPで設定されているかも。
  - RFC1035の用語ではstub resolver。

検索の流れ(続き)

- BINDをインストールしてもresolverルーチンはOSに付属の物を使っていることが多い。
  - BIND9では意識的にインストールしないとコンパイルすらされない。
  - 後からインストールしたライブラリを意識的にlink。
  - ダイナミックリンクのOSなら共有ライブラリの中のモジュールを差し替え。
  - CA-2002-19(Buffer Overflows in Multiple DNS Resover Libraries)はresolverルーチンのセキュリティホール。
    - ネームサーバではなく全クライアントで対策が必要。

検索の流れ(続き)

- アプリケーション(resolver)からqueryを受けたネームサーバ
  - rootサーバに対し、目的のRRに近づくためのauthorityの委任先をnon-recursiveに要求。
    - rootサーバだけは決め打ち。
  - 得られたネームサーバに対して同様に要求。
    :
  - 目的のRRを得る。
    - あるいは「そのRRは存在しない」という情報。
  - アプリケーションに応答。

検索の流れ(続き)

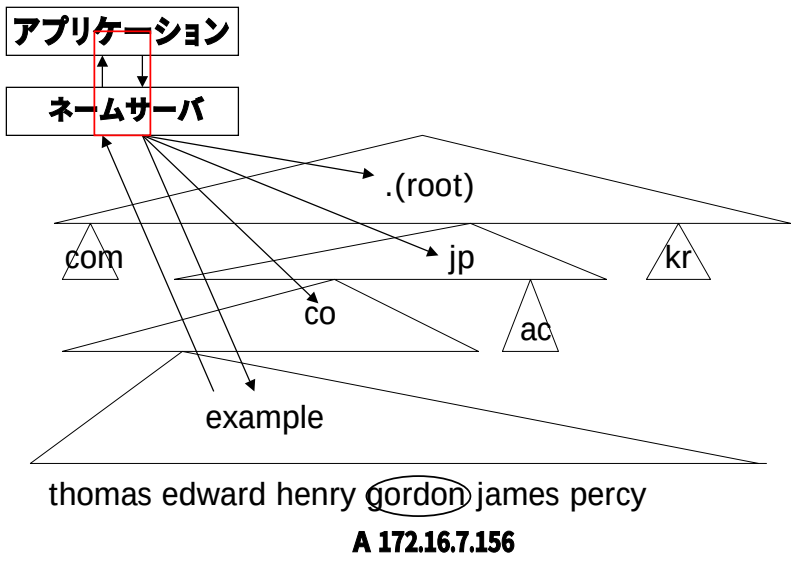
- rootサーバだけは決め打ち
  - namedの起動時に、ハードコーディングの内容(BIND9のみ)か設定ファイル(named.root)を参照して「rootサーバ」を1台選ぶ。
  - 選んだ「rootサーバ」に対してrootサーバの一覧を要求する。
  - 以降の動作にはハードコーディングや設定ファイルの内容は使わず、起動時に動的に得た一覧を使用する。

2種類の「ネームサーバ」

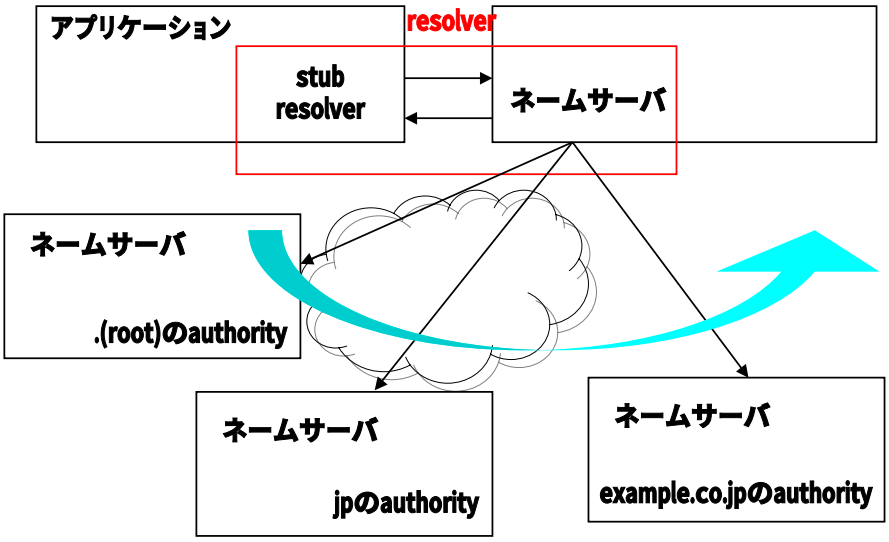
- アプリケーションからrecursive queryを受けるネームサーバ
  - アプリケーションに対し、WorldWideに関するネームサービスを提供。
  - resolv.confに書かれるネームサーバ
  - RFC1035ではrecursiveサーバと呼んでいる。
  - cacheサーバと呼ばれることもある。

  v.s.
- あるゾーンをサービスするネームサーバ
  - WorldWideに対し、そのゾーンのauthorityとしてネームサービスを提供。
  - NS RRに書かれるネームサーバ
  - authorityサーバ、contentsサーバなどと呼ばれる。
- 同じ「ネームサーバ」だが役割に違い。

検索の流れ(続き)



resolver: RFC1035での用語



逆索き

- DNSとは
  - ホスト名<->IPアドレスなどの変換

  をするもの、だったはず。
- さっきから->の話ばっかり。
- <-はどうなっている?

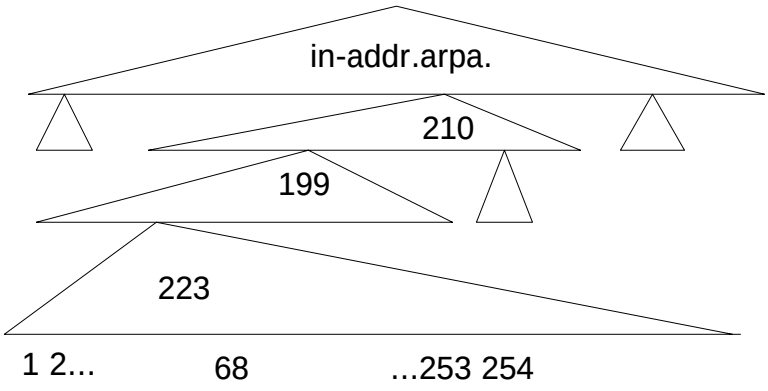
逆索き(続き)

- IP(v4)アドレス
  - 210.199.223.86
  - 階層があって'.'で区切られている。
  - なんだ、ドメイン名と同じだ!
- IP(v6)アドレス
  - フォーマットはちがうが方法は応用。

逆索き(続き)

- ホスト名
  - [左]小さい単位(子)->大きい単位(親)[右]
- IPアドレス
  - [左]大きい単位->小さい単位[右]
    - 大きい単位:ネットワーク部
    - 小さい単位:ホスト部
- 逆順で表記
  - 86.223.199.210.in-addr.arpa.
  - 8.9.a.b.(省略).f.f.2.0.0.0.1.0.4.0.5.0.e.f.f.3.ip6.arpa.
    - 以前はip6.int.だった。

逆索き(続き)



/24に満たないアドレス空間

- 例えば
  - 172.16.7.0/25: A社
  - 172.16.7.128/28: B学校
  - 172.16.7.144/29: C社
  - 172.16.7.152/29: D団体
  - 172.16.7.160/27: E社
  - 172.16.7.192/26: F社
- 「Class C」は死語。

/24に満たないアドレス空間(続き)

◆ ゾーンは自律的な管理が及ぶ範囲で区切られるべき。
  ◆ 最初の方のスライドより。
◆ 7.16.172.in-addr.arpa.は誰が管理する?
  ◆ 172.16.7.0/24に対応。

RFC2317

◆ RFC2317
  Classless IN-ADDR.ARPA delegation
  (Best Current Practice)
  ◆ 0/25.7.16.172.in-addr.arpa.
    ◆(A社: 172.16.7.0/25)
  ◆ 128/28.7.16.172.in-addr.arpa.
    ◆(B学校: 172.16.7.128/28)
    :
  ◆ 192/26.7.16.172.in-addr.arpa.
    ◆(F社: 172.16.7.192/26)
  を各組織が管理。

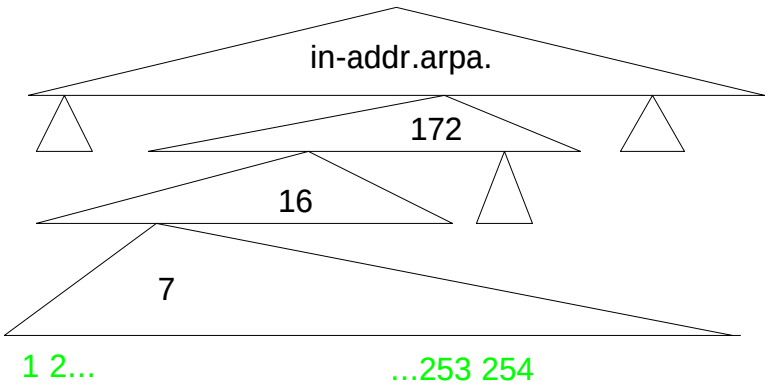
RFC2317(続き)

◆ 7.16.172.in-addr.arpa.ゾーンでは

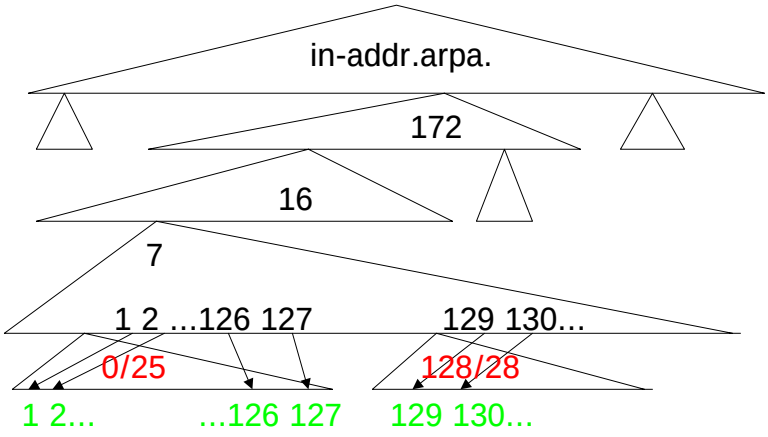
1.7.16.172.in-addr.arpa.
->1.0/25.7.16.172.in-addr.arpa.
2.7.16.172.in-addr.arpa.
->2.0/25.7.16.172.in-addr.arpa.
:
129.7.16.172.in-addr.arpa.
->129.128/28.7.16.172.in-addr.arpa.
:

のCNAMEを定義して辻褄を合わせる。

/24以上の場合



/24未満の場合



/24に満たないアドレス空間(続き)

- 7.16.172.in-addr.arpa.は誰が管理する?
  - ゾーン自体はISPが管理する。
  - でもPTRは実質的に顧客が管理する。
  - 顧客が融通の効かないGUIなサーバを使っている場合などはISPが直接PTRを書くこともある。
    - 更新は人間プロトコル、CGIなどDNSの枠外の方法。

RFC2317(続き)

- 具体的なゾーン名はISPと顧客の間で辻褄が合っていれば自由度あり。
  - 157.156/29.7.16.172.in-addr.arpa.
  - 157.156.7.16.172.in-addr.arpa.
  - 157.d-group.7.16.172.in-addr.arpa.
    :
- ISPの指示に従って下さい。

recursiveサーバからの不要な問い合わせの抑制

- localhost.の正索き、逆索き
  - rootサーバから検索するまでもなく、サイト内で適切に応答できる。
    - rootサーバはサービスしていない。
- 使っていないプライベートアドレスの逆索き
  - 同様にサイト内で適切に応答できる。
  - 「適切な応答」とはNXDOMAIN。SOAとNSだけ設定すればよい。
- 使っているプライベートアドレスの逆索き
  - サイトAでは192.168.0.1はpc001.site-a.co.jp
  - サイトBでは192.168.0.1はprinter.site-b.co.jp
  - サイト内で解決しないと正しい答えは得られない。
- recursiveサーバで、それぞれのゾーンのAuthorityを持つ。
- 大規模サイトではAS112相当の設定をするとよいかも。
  - http://www.as112.net/
  - BGPだと適用範囲が限られるが、IGPを使えば広く適用できる。

リソースレコード

- Resource Record: RRと略記
- A
  - ホスト名->IPv4アドレス

```
thomas IN A 172.16.7.153
```

- AAAA
  - ホスト名->IPv6アドレス

```
thomas IN AAAA 3ffe:504:fedc:ba98::1
```


リソースレコード(続き)

- PTR
  - IPアドレス->ホスト名

```
153.7.16.172.in-addr.arpa. IN PTR thomas.example.co.jp.
```

- MX
  - ドメイン名->メールの配送先ホスト名と優先度

```
example.co.jp.   IN   MX   10   gordon.example.co.jp.
                 IN   MX   20   henry.example.co.jp.
```

  - prefeneceは小さいほど優先。

リソースレコード(続き)

◆CNAME

◆alias->正規名

```
www.example.co.jp. IN  CNAME edward.example.co.jp.
```

◆CNAME RRを記述した名前には他のRRを記述できない。

```
www.example.co.jp. IN CNAME edward.example.co.jp.
                   IN MX 10 gordon.example.co.jp.
```

はダメ。

リソースレコード(続き)

◆1つのaliasに複数の正規名を記述してはいけない。

```
www  IN  CNAME  gordon.example.co.jp.
     IN  CNAME  james.example.co.jp.
```

はダメ。

◆BIND8ではmultiple-cnames yesと設定すれば許容されたがBIND9ではダメ。

◆NSやMXで指定するホスト名にはCNAMEで定義するaliasを書いてはいけない。

◆RFC974にはよくないという趣旨のことが書いてある。

◆RFC2181ではmust not be an aliasと書いてある。

リソースレコード(続き)

◆CNAMEのCNAMEも避ける。
- ◆循環参照回避
- ◆RFCでは禁止していないが、××段まで動作すること、というような記述もない。
- ◆BIND8では8段、BIND9では16段を超えると正規化を打ち切る。
- ◆dnscache(djbdns)は4段らしい(伝聞)。
- ◆自サイトではなく相手サイトのネームサーバの実装に依存。
  - ◆「うちはBIND9だから16段まで大丈夫」ではなく「djbdnsに索かれたら5段でアウト」

リソースレコード(続き)

◆ ありがちな間違い
- ◆ user@example.co.jpというメールアドレスを使いたい。

```
@   IN   SOA    thomas.example.co.jp. root.example.co.jp. (
                2004120201
                1h
                15m
                30d
                15m)
        CNAME   gordon      ; MXを設定するのが正しい。
```

リソースレコード(続き)

- NS
  - ゾーン名->authorityのホスト名

```
example.co.jp.  IN  NS  thomas.example.co.jp.
```

  - 親ゾーン中に記述
    - 子ゾーンのauthorityの所在。
    - authorityの所在が変わるときは所定の手続き。
    - slave(後で説明)にも委任してもらう。
  - 子ゾーン中に記述
    - 自分がauthorityであることの宣言。
    - slaveもauthorityであることを宣言。

リソースレコード(続き)

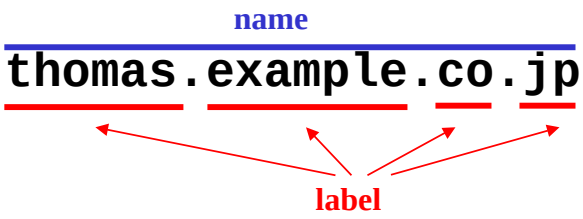
- SOA
  - ゾーン名->cacheやslaveを制御するパラメータ群

```
example.co.jp. IN SOA thomas.example.co.jp. root.example.co.jp. (
                  2004120201
                  1h
                  15m
                  30d
                  15m)
```

  - cacheやslaveが登場してから説明。
- 他にもいろいろなtypeのRRがあるが、普通はこれだけあれば十分。

ドメイン名の制約

- 文字数(RFC1035)
  - label: 63文字まで
  - name: 255文字まで



ドメイン名の制約(続き)

- 文字種
  - RFC1035に、labelは
    - アルファベットで始まり
    - アルファベット、数字、'-'(ハイフン)の繰り返し
    - アルファベットまたは数字で終わる

    のが無難だろう、という意味のことが書いてある。
  - RFC1123で1文字目が数字のドメイン名もよいことになった。
    - 3com.com、0123.co.jp、…
  - RFC2181では8bit cleanであるべし、となった。

ドメイン名の制約(続き)

- '_'はダメ。
  - BIND4.9.xのあるバージョンでチェックが厳しくなり、slave(secondary)をしていたゾーンがエラーになってあせった。
  - BIND8、9.3ではチェックの厳しさが指定できる。(RFC1035)
    - zone{check_names …;};
    - warn,fail,ignore
  - BIND9.0〜9.2はチェックしていない。(RFC2181)
- 大文字/小文字は区別されない。
  - internetweek.jp
  - InternetWeek.JP

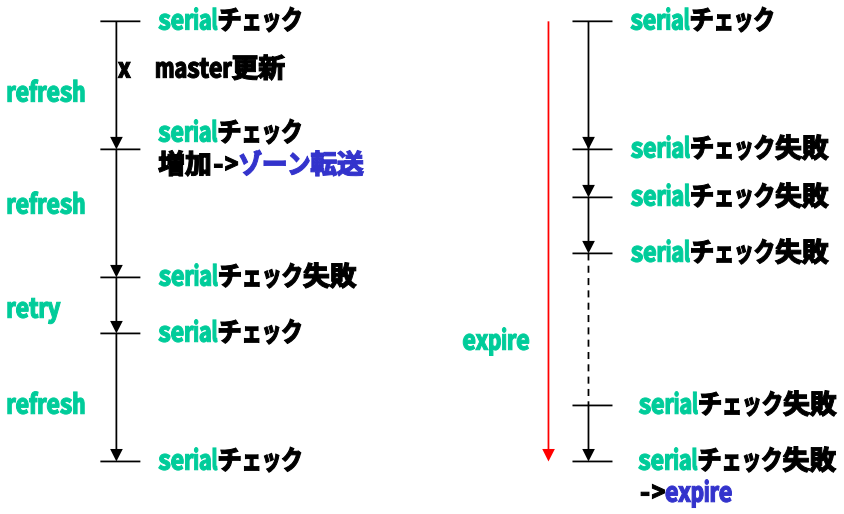
slave

- データは各ゾーン毎に一元管理。
  - DNSの長所
- 障害時は?
  - single point of failureではいけない。
  - 他のネームサーバにゾーンデータをコピー。
  - そっちのネームサーバにもauthorityを委任。
    - slave(v.s.master)
      - 昔はsecondaryと言った。(v.s.primary)
    - queryした側には対等に見える。
      - fallbackではない。

slave(続き)

- 定期的にmasterのデータが更新されていないかチェック。
  - SOA RRの各パラメータで制御される。
  - serialが増加していないか?
    - 増加していればゾーン転送でデータをコピー。
  - refresh(秒)間隔でチェック。
  - 失敗したらretry(秒)で再試行。
  - expire(秒)の間、失敗し続けたら、その時点で保持しているデータは無効化。

slave(続き)



slave(続き)

- NOTIFY
  - RFC1996
  - masterの更新があったときにslaveに対して能動的に通知。
  - slaveがrefresh(秒)間隔でチェックに来るのを待たない。
    - 変更が速く伝播する。
  - best effort
  - NOTIFYを聴かないslaveも居る。

slave(続き)

- 用語の整理
  - primary、secondary
    - ゾーンデータの出所に注目。
    - ローカルならprimary、ゾーン転送で得ていればsecondary。
  - master、slave
    - ゾーン転送の動作に注目。
    - 転送元がmaster、転送先がslave。
    - 孫secondaryが居れば、第2世代は場面によってmasterだったりslaveだったりする。
  - primary masterという用語もある。

cache

- 毎回、rootサーバから順にたどっていては
  - 処理量
  - トラフィック
  - 待ち時間

  が大変。
- recursiveサーバは一度索いたRRをキャッシュ。
- そのRRのTTLの間だけキャッシュ上に保持。
- authority側にはinvalidation手段はない。
  - 設定変更時は事前にTTLを短縮。

TTL

- TTLはどこで設定する?
  - さっきのRRの説明には出てこなかったけど…。
- 各RRに個別に指定。

```
thomas   1d   IN A 172.16.7.153
```

- $TTLディレクティブでそのゾーン中のRRのTTLのデフォルトを設定。

```
$TTL 1d
@ IN   SOA   thomas.example.co.jp…(…)
       :
```

TTL(続き)

- $TTL
  - RFC2308で導入された。
  - BINDでは8.2から対応。
  - ゾーンファイル中、$TTL以降のRRに作用。
  - ないとnamedが警告を出す。
    - BIND9.0.x、9.1.xではエラーになる。
- SOAのminimumフィールド
  - BIND8.2より前ではこの値が省略時のTTL。
  - BIND8.2以降ではnegative cache上での保持期間に意味が変わった。
  - BIND8.2以降でもRRに明示的指定がなく、$TTLもなければminimumの値が使われる。
    - 9.0.x、9.1.xを除く。

negative cache

- queryしたRRが存在しなかったときに、しばらくの間、同じRRのqueryを抑制するcache。
  - 処理量、トラフィックの削減という目的は同じ。
  - 具体的なRRの値ではなく、存在しなかったという事実をcache。
- RFC2308

negative cache(続き)

- エラーで索けなかった場合ではなく、明示的に「存在しない」という応答を得た場合。
  - 名前そのものがない。
    - NXDOMAIN
  - 名前はあったが、そのtypeのRRが定義されていない。
    - NODATA、NXRRSET
- BIND8.2から対応。
  - $TTLの導入
  - SOAのminimumの意味の変更。

negative cache(続き)

- minimumはよそのネームサーバに存在しないRRをquery されたときに、相手のネームサーバのnegative cacheに保持させる時間。
  - このネームサーバがnegative cacheに保持する時間ではない。
  - そういう値なら個々のゾーンではなくnamed.conf中に記述するはず。
    - max-ncache-ttl

SOA

MNAME

RNAME

```
example.co.jp. IN  SOA thomas.example.co.jp. root.example.co.jp. (
                        2004120201     serial
                        1h             refresh
                        15m            retry
                        30d            expire
                        15m)           minimum
```


SOA(続き)

- MNAME
  - そのゾーンのデータの大元があるホスト名。
  - primaryとsecondaryの見分け。
- RNAME
  - そのゾーンの管理者のメールアドレス
  - @は.に書き換える。
    - hostmaster@example.co.jp.->hostmaster.example.co.jp.
    - @はゾーンデータの中では特別な意味(origin)。
    - @の前に.を含むメールアドレスは¥.に書き換える。
      - Sir.Tophamhat@example.co.jp.->Sir¥.Tophamhat.example.co.jp.

SOA(続き)

- 以下の各数値は
  - 32bit unsigned int
  - 時間の単位は秒
- serial
  - ゾーンデータの鮮度。
  - 日付+通し番号を使うことが多いが、あくまで人間界の流儀。
  - 日付を付けずに1から順に増やす流儀もあり。
  - slaveがmasterの更新をチェックする際の手がかり。

SOA(続き)

- refresh
  - slaveにmasterの更新をチェックさせる間隔。
- retry
  - refreshのタイミングでチェックに失敗したときの再試行間隔。
- expire
  - retryを繰り返してもチェックに失敗し続けたときに、その時点で保持しているデータの有効期限。
- minimum
  - negative cache上でのデータの保持期間。

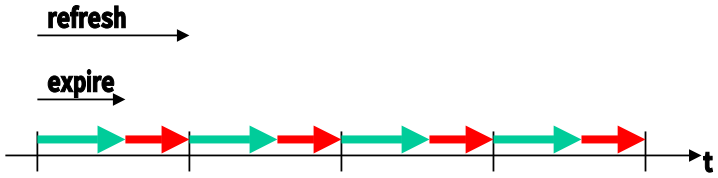
SOA(続き)

- RFC1033には
  - refresh: 3600(1時間)
  - retry: 600(10分)
  - expire: 3600000(約42日)
  - minimum: 86400(24時間)
    - ◆現代風には$TTLと読み替えること。

  がお勧め値と書いてある。
- RFC1912のお勧めは
  - expire: 2-4 weeks
  - minimum: 1-5 days

SOAに関する失敗例(その1)

- expire < refresh にしてしまった。
  - slaveはrefreshの間隔でmasterの更新をチェックするが、その間に必ずexpireしてしまう。
  - 時の運でslaveが正常に応答したりエラーになったり。

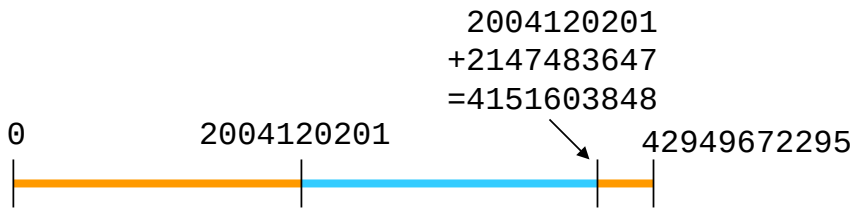


SOAに関する失敗例(その2)

◆ serialに大きすぎる値を設定してしまった。
- ◆ 手が滑って
  - ◆来月の日付にしてしまった。
  - ◆桁数が1桁増えちゃった。
  - :
- ◆ 昔ならゾーン転送されちゃう前に直せばセーフだった。
- ◆ 今はNOTIFYのおかげで即座にゾーン転送されてしまう。

SOAに関する失敗例(その2:続き)

◆ serial
- ◆32bit
- ◆0〜4,294,967,295=(2^32)-1
- ◆ある数nから次の2,147,483,647(=(2^31)-1)個の数はnより大きい。
- ◆nの前2,147,483,647個の数はnより小さい。
- ◆4,294,967,295の次は0。



SOAに関する失敗例(その2:続き)

◆4294967295より大きな値だとエラーになる。

Sep 4 14:43:18 thomas named[35341]: general: error: dns_rdata_fromtext: localhost:3: near '4294967297': out of range

Sep 4 14:43:18 thomas named[35341]: general: error: zone localhost/IN: loading master file localhost: out of range

◆2004120201に設定したかったのに3004120201とタイプしてしまい、reloadしてから気付いた ;_;

- ◆3004120201より「大きく」2004120201より「小さな」番号に設定。
- ◆slaveにゾーン転送。
- ◆2004120201に設定。
- ◆slaveにゾーン転送。

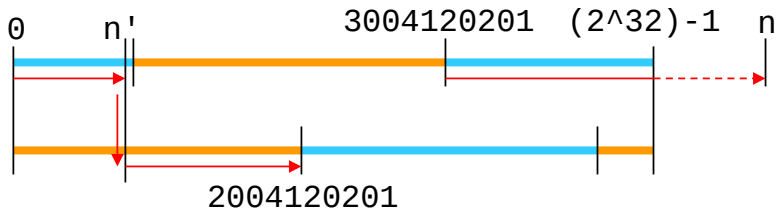
SOAに関する失敗例(その2:続き)

◆n=3004120201+(2^31)-1=5151603848>2^32

◆n'=n-2^32=856636552

3004120201 < n'

n' < 2004120201



時刻表記

◆ SOAや$TTLの時間の表記には
  ◆ w(week)、d(day)、h(hour)、m(min)
  などの単位が使えるらしい。
◆ ARMのSetting TTLsの項目には
  ◆ All of these TTLs default to units of seconds, though units can be explicitly specified, for example, 1h30m.
  とこっそり(?)書いてある。

ゾーンデータ

◆ 1行1RR
  ◆ (…)でくくると行をまたげる
    ◆とRFC1035には書いてあるが、SOA以外では見たことがない。
    ◆BIND8でAAAAの記述に含めてみたらエラーになった。
◆ RRの他、
  ◆ $ORIGIN(RFC1035)
  ◆ $INCLUDE(RFC1035)
  ◆ $TTL(RFC2308)
  ◆ $GENERATE(BINDの独自拡張)
  のディレクティブ。

ゾーンデータ(続き)

- コメント記号は;
  - #はコメント記号ではない。
    - UNIXの常識に反している。
      - DNSサーバの最初の実装、JeevesはUNIXではなくTOPS-20で開発された。
    - named.confと不統一でまぎらわしい。

ゾーンデータ(続き)

- RRのフォーマット

  <owner> [<TTL>] [<class>] <type > <RDATA>

  - <owner>
    - それ以降のフィールドが対応づけられるドメイン名。
      - DNSの用語としてはホスト名も「ドメイン名」。
    - 行頭が空白だと直前のエントリのownerが引き継がれる。
  - <TTL>
    - 省略時は$TTLの値が使われる。
  - <class>
    - 省略時は直前のエントリのclassが引き継がれる。普通IN。
  - <type>,<RDATA>
    - SOA,NS,A,AAAA,MX,PTR,CNAME,…

ゾーンデータ(続き)

- 相対表記/絶対表記
  - ドメイン名の末尾が'.'で終わっていれば絶対表記。
    - ownerとNS,MX,CNAME,PTRのRDATA
    - A、AAAAのRDATAは関係ない。
  - '.'で終わっていなければ、あるドメインを起点とした相対表記。

ゾーンデータ(続き)

- ORIGIN
  - 相対表記のときに後ろに補われるドメイン名。
  - ゾーンデータ中では@で参照できる。
    - これがSOAのRNAMEに@が使えない理由。
  - 最初はnamed.confの中でそのゾーンデータと対応付けられているゾーン名がORIGIN。
  - $ORIGINディレクティブで変更できる。

相対表記に関する失敗例

◆ RRの末尾に‘.’を忘れる。

```
@   IN   SOA …(…)
    IN   NS   thomas.example.co.jp
    IN   NS   ns.myisp.ad.jp
```

は

```
@   IN   SOA …(…)
    IN   NS   thomas.example.co.jp.example.co.jp.
    IN   NS   ns.myisp.ad.jp.example.co.jp.
```

に展開されてしまう。

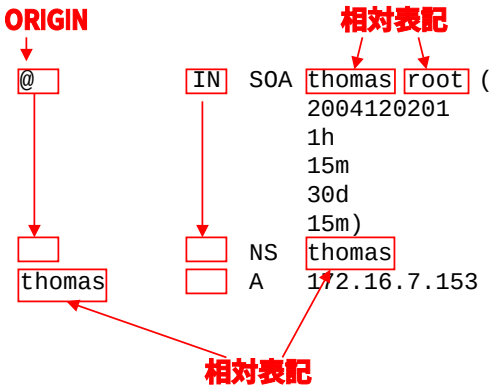
ゾーンデータ(続き)

◆ tips
- ◆ 相対表記、絶対表記に関しては、自分の流儀を決めておく。
  - ◆相対表記は使わない。必ず絶対表記。
  - ◆ownerは必ず相対表記、RDATAは必ず絶対表記。
  - ◆相対表記できるところは必ずする。
    - ◆ etc
- ◆ 設定ミスを見つけやすくなる。

ゾーンデータ(続き)

```
example.co.jp.  IN  SOA  thomas.example.co.jp.  root.example.co.jp. (
                         2004120201
                         1h
                         15m
                         30d
                         15m)
example.co.jp.           IN  NS   thomas.example.co.jp.
thomas.example.co.jp.    IN  A    172.16.7.153
```


ゾーンデータ(続き)



サブドメイン

◆ サブドメインを追加するには
  ◆ ゾーンを分ける方法
    ◆ゾーンを分けて、別のネームサーバに委任。
    ◆ゾーンは分けるが、自ホストのネームサーバに委任。
  ◆ ゾーンを分けない方法
  の2.5通りの設定がある。
◆ どの方法をとるかは運用上の選択。

サブドメイン(続き)

◆ ゾーンを分ける方法

```
$ORIGIN example.co.jp.
branch          IN  NS  toby.branch.example.co.jp.
toby.branch     IN  A   192.168.0.1
```

これがglue

```
$ORIGIN branch.example.co.jp.
@           IN  NS  toby
toby        IN  A   192.168.0.1
henrietta IN  A   192.168.0.2
```

サブドメイン(続き)

◆ゾーンを分けない方法

```
$ORIGIN example.co.jp.
toby.branch         IN  A  192.168.0.1
henrietta.branch    IN  A  192.168.0.2
```


BINDの基本的な設定

namedが動くまで

BIND9のインストール

- ftp://ftp.isc.org/isc/bind9/からget。
  - 資料作成時点では9.3.0が最新リリース。
- コンパイル時設定はconfigureを使う。
  - ドキュメントには
    - ANSI C
    - basic POSIX support
    - 64bit integer type

    をサポートしたUNIXをrequireする。
    - COMPAQ Tru64 UNIX 5.1B、FreeBSD 4.10, 5.2.1、HP-UX 11.11、NetBSD 1.5、Slackware Linux 8.1、Solaris 8, 9, 9(x86)、Windows NT/2000/XP/2003でbuildに成功した。

  と書いてある。
    - ISCでは未確認の動作報告もいくつか紹介されている。
- 詳細はREADMEのBuildingのセクション参照。

BIND9のインストール(続き)

- --prefixはどうする?
  - デフォルトでは/usr/local
  - --prefix=/usrとすれば/usr/sbin、/usr/binなどのバイナリを上書き。
    - 間違って別バージョンを起動してしまう心配はない。
    - /etc/rc*は書き換えなくても済むかも。
    - 引き返せない(^^;
    - make installworld(FreeBSDの場合)すると、OSのバイナリで上書きされてしまう。
  - --prefix=/usr/local/bind-9.3.0とすれば、9.3.1(?)へのバージョンアップのときにも9.3.0のバイナリを温存できる。
    - トラブル時の切り戻しが楽。
    - /etc/rc*はその都度書き換え。
    - あるいはsymbolic linkで-9.3.0を隠蔽してもいい。

BIND9のインストール(続き)

- --sysconfdirはどうする?
  - named.confやrndc.confなどのデフォルトでの探索先。
  - resolv.confには影響しない(/etcで決め打ち)。
    - そもそもBIND9のレゾルバじゃなきゃ無関係。
  - 手許の参考書に合わせておく?
    - 「DNS & BIND」なら/etc
    - 今日の説明は/etcで
  - OSに入ってきたバイナリとそろえておく?
    - FreeBSDなら/etc/namedb
  - --prefixを指定すれば$PREFIX/etcがデフォルト。
  - --prefixを指定しないと、デフォルトは/usr/local/etcではなく/etc。

BIND8のインストール

- ftp://ftp.isc.org/isc/bind/srcからget。
  - 資料作成時点の最新は8.4.5
- configureは使っていない。
- src/port/*OS*/Makefile.setを編集。
- srcで

  make depend

  make all
- 詳しくはsrc/INSTALL参照。

この資料で参照しているBINDのバージョン

- この資料を作成するための動作確認などに使ったバージョンは主に9.3.0と8.4.5。
- 何ヶ所かで動作確認に基づく挙動の紹介やARMとの相違の指摘があるが、バージョンによる違いもあるかも。

BINDの設定ファイル

- named.conf
  - サービスするゾーン名
    - master/slaveの別
    - 定義ファイル(master)/dumpファイル(slave)
    - アクセス制限
  - ログの出力方法/内容
  - (r)ndcコマンドが使う制御チャネル

  などnamed全体に関わる設定を記述。

BINDの設定ファイル(続き)

◆ デフォルトではconfigureの--sysconfdirで指定したディレクトリが探索される。
  ◆BIND8ではsrc/port/*OS*/Makefile.setで指定。
  ◆named -c *filename*で別のファイルを指定可。
◆ コメント記号は

```
#      (行末まで)
//     (行末まで)
/*
          :   (複数行可)
 */
```


BINDの設定ファイル(続き)

◆ masterとなるゾーンの定義ファイル
  ◆ ゾーン毎にそのゾーンに関するリソースレコード群を記述。
  ◆ ファイル名はnamed.confの中で指定。

BINDの設定ファイル(続き)

- named.root
  - rootサーバの指定。
  - BIND9ではlib/dns/rootns.cにハードコーディングされているので、特別な場合以外は設定不要。
  - BIND8はハードコーディングされていないので、かならず必要。
  - 普通、内容はサイトに依存しないので、設定するならftpでgetしてきたのをそのまま使えばよい。
    - ftp://rs.internic.net/domain/named.root
    - ftp://ftp.nic.ad.jp/internic/rs/domain/named.root
  - 閉じた名前空間を形成するためには自分で設定。
  - ファイル名は任意(named.confの中で指定)だが、資料によってはroot.cacheという名前を使っている。

BIND9固有の設定ファイル

- rndc.conf
  - rndcの設定ファイル。
  - 鍵情報の他、使用する制御チャネルなど。
  - named側の設定はnamed.confに。
- rndc.key
  - rndcがnamedにアクセスする際の認証鍵。
  - named.conf中の設定やrndc.confがなければrndcとnamedの双方がこのファイルを参照。
- BIND8はrndcではなくndc。
  - UNIXドメインソケットを使うときは、鍵ではなくファイルシステムのpermissionで保護。
  - BIND8でもINETドメインソケットも設定できるが、認証機構がないので、使ってはいけない。
  - named.confのcontrols{...}を必要に応じて設定。

BINDの設定ファイル(続き)

- resolv.conf
  - namedの設定ファイルではなくレゾルバライブラリの設定ファイル。
    - どのネームサーバにrecursive queryするか。
      - RFC1035の視点ではstub resolverのバックエンド
    - 省略時ドメイン名
  - ネームサーバ(のホスト)だけでなくDNSを使う全ホストで設定。
    - WindowsとかMacOSにも相当する設定がある。

システム設計

- 漫然と設定してはいけない。提供する機能、対象を明確にする。
- 要求仕様
  - 以下のゾーンのprimary masterになる。
    - example.co.jp
    - 152/29.7.16.172.in-addr.arpa
    - 8.9.a.b.c.d.e.f.4.0.5.0.e.f.f.3.ip6.arpa
    - それぞれns.myisp.ad.jpにsecondaryをしてもらう。
    - World Wideに対してサービスすることが目的。
  - レゾルバ向けrecursiveサーバになる。
  - 以下のゾーンのprimary masterになる。
    - localhost
    - 127.in-addr.arpa
    - 1.0.0.0.(途中 略).0.0.0.0.ip6.arpa(::1の逆索きゾーン)
    - プライベートアドレスなどの逆索きゾーン
    - 不要な問い合わせを外部にリークしないことが目的。

システム設計(続き)

- コンセプト
  - 要求仕様を満たす最小限の設定をする。
  - アクセス制限などもきちんとするのが望ましいが、先送りにする。
    - Advanced topicsのセクション参照

named.conf

```
options {
        directory "/usr/local/etc/namedb";
        listen-on-v6 {
                any;
        };
};
```

namedのプロセスは
このディレクトリにchdir()する。
相対パスの起点。

忘れがち

これがないと、カーネルがサポートしていても
namedはv6を聴かない。

named.conf(続き)

```
controls {
        unix "/var/run/ndc" perm 0600 owner 0 group 0;
};
```

BIND9では不要。今回の例ではrndc.keyを使う。
明示的に書くなら構文が異なるので注意。
BIND8のデフォルトは、この内容に見えるが、
マニュアルに明示的には書いてないので、
一応入れた。

named.conf(続き)

```
include "conf/martian.conf";
```

後で登場する
(/usr/local/etc/namedb/)conf/martian.conf
の内容をこの箇所に差し込む。
直接ここへ書いてもよいが、可読性向上のために
別ファイルにした。

named.conf(続き)

```
zone "." IN {
        type hint;
        file "named.root";
};
```

BIND9では同等の内容が
ハードコーディングされているので、
特別なとき以外は設定しなくてよい(してもよい)。

named.conf(続き)

```
zone "localhost" IN {
        type master;
        file "master/localhost";
};
zone "127.in-addr.arpa" {
        type master;
        file "master/127.in-addr.arpa";
};
```

省略すればIN

named.conf(続き)

```
zone "1.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0
.0.0.0.0.0.0.0.0.ip6.arpa" {
        type master;
        file master/1000.0000.0000.0000.0000.0000
.0000.0000.ip6.arpa";
};
```

それぞれ1行です。

named.conf(続き)

```
zone "example.co.jp" {
        type master;
        file "master/example.co.jp";
};
zone "152/29.7.16.172.in-addr.arpa" {
        type master;
        file "master/152_29.7.16.172.in-addr.arpa";
};
zone "8.9.a.b.c.d.e.f.4.0.5.0.e.f.f.3.ip6.arpa" {
        type master;
        file "master/89ab.cdef.4050.eff3.ip6.arpa";
};
```

conf/martian.conf

- named.confに直接書いてもいいが、ここでは別ファイルにまとめてincludeしてみた。
  - named.confの可読性向上

```
zone "10.in-addr.arpa" {
        type master;
        file "master/empty";
};
zone "254.169.in-addr.arpa" {
        type master;
        file "master/empty";
};
```


conf/martian.conf(続き)

```
zone "16.172.in-addr.arpa" {
        type master;
        file "master/empty";
};
                :    172.16/12はoctet-boundry
                :    じゃないので面倒。
zone "31.172.in-addr.arpa" {
        type master;
        file "master/empty";
};
```

conf/martian.conf(続き)

```
zone "168.192.in-addr.arpa" {
        type master;
        file "master/empty";
};
```


master/empty

ないと警告される

```
$TTL 1d
@   IN   SOA thomas.example.co.jp. root.example.co.jp. (
             2004120201
             1h         時間は秒単位ではなく
             15m        m(分)、h(時間)、d(日)、w(週)を使って表記した。
             4w
             15m )
    NS   thomas.example.co.jp.
```

省略時は直前のエントリのクラスが引き継がれる。

master/localhost

```
$TTL 1d
@  IN  SOA thomas.example.co.jp. root.example.co.jp. (
           2004120201
           1h
           15m
           4w
           15m )
        NS    thomas.example.co.jp.
        A     127.0.0.1
        AAAA  ::1
```


master/127.in-addr.arpa

```
$TTL 1d
@       IN      SOA     thomas.example.co.jp.
root.example.co.jp. (
                        2004120201
                        1h
                        15m
                        4w
                        15m )
        NS      thomas.example.co.jp.
1.0.0   PTR     localhost.
```

1行です。

master/1000.0000.0000.0000 .0000.0000.0000.0000.ip6.arpa

```
$TTL 1d
@  IN  SOA thomas.example.co.jp. root.example.co.jp. (
           2004120201
           1h
           15m
           4w
           15m )
        NS      thomas.example.co.jp.
        PTR     localhost.
```


master/example.co.jp

```
$TTL 1d
@       IN      SOA     thomas.example.co.jp.
root.example.co.jp. (
                        2004120201
                        1h
                        15m
                        4w
                        15m )
                NS      thomas.example.co.jp.
secondaryにも → NS      ns.myisp.ad.jp.
委任            MX      10 gordon.example.co.jp.
                MX      20 henry.example.co.jp.
```

1行です。

相対表記、絶対表記は流儀を決めておく。

master/example.co.jp(続き)

```
localhost       CNAME   localhost.
thomas          A       172.16.7.153
                AAAA    3ffe:504:fedc:ba98::1
edward          A       172.16.7.154
henry           A       172.16.7.155
gordon          A       172.16.7.156
                AAAA    (  ←―― BIND9ならこういうこともできる
        3ffe:504:fedc:ba98:1234:5678:9abc:def0)↙
james           A       172.16.7.157
percy           A       172.16.7.158
```


master/152_29.7.16.172.in-addr.arpa

```
$TTL 1d
@       IN      SOA     thomas.example.co.jp.
root.example.co.jp. (
                        2004120201
                        1h
                        15m
                        4w
                        15m )
                NS      thomas.example.co.jp.
                NS      ns.myisp.ad.jp.
```

1行です。

master/152_29.7.16.172.in-addr.arpa

```
153             PTR     thomas.example.co.jp.
154             PTR     edward.example.co.jp.
155             PTR     henry.example.co.jp.
156             PTR     gordon.example.co.jp.
157             PTR     james.example.co.jp.
158             PTR     percy.example.co.jp.
```


master/89ab.cdef.4050.eff3.ip6.arpa

```
$TTL 1d
@ IN  SOA thomas.example.co.jp. root.example.co.jp. (
          2004120201
          1h
          15m
          4w
          15m )
      NS      thomas.example.co.jp.
      NS      ns.myisp.ad.jp.
1.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0 PTR thomas.example.co.jp.
0.f.e.d.c.b.a.9.8.7.6.5.4.3.2.1 PTR gordon.example.co.jp.
```

1行です。

rndc.key(BIND9のみ)

- *$PREFIX*/sbin/rndc-confgen -aで生成。
  - /etc/rndc.keyというファイルを生成する。
  - named
    - /etc/rndc.keyに書かれた鍵で認証。
      - controls{}がなければ、localhostからのアクセスだけを許可。
      - controls{}があってkeys節がなければ、allow節にしたがう。
  - rndc
    - /etc/rndc.confがなければ/etc/rndc.keyに書かれた鍵でlocalhostへアクセス。
  - 凝った設定をするなら
    - named.confにcontrols{}ディレクティブを記述
    - /etc/rndc.confを作成。

rndc.key(BIND9のみ:続き)

- FAQ:rndc-confgenがハングアップする。
  - 少なくともFreeBSDとNetBSDの一部のバージョンではこの現象が発生し得る。
  - /dev/randomの速度が遅いのが原因。
    - 本当にハングアップしているのではなく、所要時間が極端に長くかかっている。
  - /dev/urandamを使う(乱数の質は落ちるらしい)。
  - -r keyboard(手で乱数(?)を生成)。
  - rndcontrolを使う(FreeBSD/i386でのみ確認)
    - /etc/rc.confでrand_irqsを設定。

rndc-confgen(BIND9のみ)

```
thomas# rndc-confgen –a -r keyboard
start typing:
.................................
.........................
.........................
.........................
.........................
.........................
.........................
.........................
stop typing.
```

コマンドを起動

この間、適当にタイピングしている。

rndc.key(BIND9のみ:続き)

- rndc.keyが読めればrootじゃなくてもrndcコマンドが実行できる。
  - ファイルシステムのpermissionで制限が必要。
    - -aオプションで生成すれば適切に設定される。
  - 例えばwheelな人はsuしなくてもrndcできるような設定もできる。
  - BIND8のnamed.confのcontrols{...}も同様。

named-checkconf

- named.confの構文をチェック。
  - lintとかapachectl configtestのようなもの。
  - BIND8には含まれないが、BIND8用のnamed.confにも適用可。

```
thomas# named-checkconf ./named.conf
./named.conf:49: missing ';' before 'file'
```


named-checkzone

- zoneファイルの構文をチェック。
- ExpireがRefreshより短いのは検出できなかった。
  - あくまで字句、構文のチェック。
- BIND8の配布物には含まれないが、BIND8用のゾーンデータにも適用可。
  - 細部はあてにならない点もあるので、だいたいの確認に。

```
thomas# named-checkzone example.co.jp.
example.co.jp

dns_rdata_fromtext: example.co.jp:21: near
'localhost.': bad dotted quad

zone example.co.jp/IN: loading master file
example.co.jp: bad dotted quad
```

named-checkzone(続き)

◆ BIND9.3.0に含まれるnamed-checkzoneは名前の文字セットもチェックできるようになった。

```
thomas# named-checkzone -k check-names
example.co.jp example.co.jp

example.co.jp:20: _percy.example.co.jp: bad owner
name (check-names)

zone example.co.jp/IN: loaded serial 2004120201

OK
```

◆ manに書いてあるのとキーワードが違う(RT#1727)。

```
-k mode

   Perform  "check-name"  checks  with  the
specified failure mode.  Possible modes are "fail",
"warn" (default) and "ignore".
```


namedの起動(BIND9)

```
thomas# /usr/local/bind-9.3.0/sbin/named  &; tail -f
/var/log/messages

[1] 83877

Oct  6 12:16:24 thomas named[83880]: starting BIND
9.3.0

Oct  6 12:16:24 thomas named[83880]: loading
configuration from '/etc/named.conf'

        :

        :

Oct  6 12:16:24 thomas named[83880]: running
```

1行です。

rndcによる動作確認(BIND9)

```
thomas# rndc status
number of zones: 27
debug level: 0
xfers running: 0
xfers deferred: 0
soa queries in progress: 0
query logging is OFF
recursive clients: 0/1000
tcp clients: 0/100
server is up and running
```


namedの起動(BIND8)

```
thomas# /usr/local/bind-8.4.5/sbin/named &; tail -f
/var/log/messages
[1] 83826
Oct  6 11:56:40 thomas named[83826]: starting
(/usr/local/etc/namedb/named.conf).  named 8.4.5-REL
Fri Oct  1 12:22:57 JST 2004
kohi@thomas:/.amd_mnt/alphonse/u/share/usr/local/src/b
ind/bind-8.4.5/src/bin/named
Oct  6 11:56:40 thomas named[83826]: master zone
"10.in-addr.arpa" (IN) loaded (serial 2004120201)
        :
Oct  6 11:56:40 thomas named[83828]: Ready to answer
queries.
```

1行です。

ndcによる動作確認(BIND8)

```
thomas# /usr/local/bind-8.4.5/sbin/ndc status
named 8.4.5-REL Fri Oct  1 12:22:57 JST 2004
kohi@type95:/.amd_mnt/alphonse/u/share/usr/local/src/
bind/bind-8.4.5/src/bin/named
config (/etc/named.conf) last loaded at age: Tue Oct
5 15:37:34 2004
number of zones allocated: 64
debug level: 0
xfers running: 0
xfers deferred: 0
soa queries in progress: 0
query logging is OFF
server is up and running
```


一応、起動したみたいなら

- ps auxww | grep named
  - 本当に走っているか?
- ログを見てみる。
  - エラーや警告はないか?

dig

- DNSの検索ツール
- BIND9にもnslookupは付属しているが、そのうちにサポートされなくなりそう。
- dig *@server name type*
  - 詳しくはマニュアル参照

digによる動作確認

- チェックポイント
  - authorityを持っているはずのゾーン
    - 名前は索けるか?
    - ちゃんとauthoritative answerを返しているか?
    - ゾーン転送はできるか?
      - 必須ではないが、できなければ何か変なことが多い。
    - PTRやNSがthomas.example.co.jp.example.co.jpになっていないか?
  - 外部の名前は索けるか?
    - レゾルバに対するサービスの確認

digによる動作確認

◆ authorityを持っているゾーン

```
thomas# dig +norec @127.0.0.1 thomas.example.co.jp
```

再帰的検索を要求しない

```
; <<>> DiG 9.3.0 <<>> +norec @127.0.0.1
thomas.example.co.jp
;; global options:  printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id:
6590
;; flags: qr aa ra; QUERY: 1, ANSWER: 1, AUTHORITY: 2,
ADDITIONAL: 2
```

Authoritative Answer

digによる動作確認(続き)

```
;; QUESTION SECTION:
;thomas.example.co.jp.       IN    A

;; ANSWER SECTION:                              NSではなくAが返ってくること
thomas.example.co.jp. 86400  IN    A     172.16.7.153

;; AUTHORITY SECTION:
example.co.jp.        86400  IN    NS    thomas.example.co.jp.
example.co.jp.        86400  IN    NS    ns.myisp.ad.jp.
```

NSレコードも正しいか?

ns.myisp.ad.jp.example.co.jpになっていないか?

$TTLの値が適用されている

digによる動作確認(続き)

ゾーン転送

```
thomas# dig @127.0.0.1 example.co.jp AXFR

; <<>> DiG 9.3.0 <<>> @127.0.0.1 example.co.jp AXFR
;; global options:  printcmd
example.co.jp.        86400 IN  SOA  thomas.example.co.jp.
root.example.co.jp. 2004120201 3600 900 2419200 900
example.co.jp.        86400 IN  NS   ns.myisp.ad.jp.
example.co.jp.        86400 IN  NS   thomas.example.co.jp.
example.co.jp.        86400 IN  MX   10 gordon.example.co.jp.
example.co.jp.        86400 IN  MX   20 henry.example.co.jp.
```

m,h,d,wが展開されている
(digのバージョンによる)

digによる動作確認(続き)

```
edward.example.co.jp. 86400 IN   A   172.16.7.154
gordon.example.co.jp. 86400 IN   A   172.16.7.156
gordon.example.co.jp. 86400 IN   AAAA 3ffe:504:fedc
:ba98:1234:5678:9abc:def0
henry.example.co.jp.  86400 IN   A   172.16.7.155
james.example.co.jp.  86400 IN   A   172.16.7.157
localhost.example.co.jp. 86400 IN   CNAME   localhost.
percy.example.co.jp.  86400 IN  A    172.16.7.158
thomas.example.co.jp. 86400 IN  A    172.16.7.153
thomas.example.co.jp. 86400 IN  AAAA 3ffe:504:fedc:ba98::1
example.co.jp.        86400 IN  SOA  thomas.example.co.jp
root.example.co.jp. 2004120201 3600 900 2419200 900
```

digによる動作確認(続き)

- 逆索きの確認
  - 153.7.16.172.in-addr.arpaのPTRはISP側の設定が済まないと索けない。
    - 現時点では索けなくても異常ではない。
  - 153.152/29.7.16.172.in-addr.arpaのPTRで動作を確認。
    - 現時点で索けなければ異常。
  - 7.16.172.in-addr.arpaのauthorityは持っていない。
    - ゾーン転送できない。
  - 152/29.7.16.172.in-addr.arpaはauthorityを持っている。
    - ゾーン転送できなければいけない。

digによる動作確認(続き)

- 外部の名前の検索

```
thomas# dig @127.0.0.1 internetweek.jp

; <<>> DiG 9.3.0 <<>> @127.0.0.1 internetweek.jp
;; global options:  printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 59235
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 2,
ADDITIONAL: 0


;; QUESTION SECTION:
;internetweek.jp.         IN    A


;; ANSWER SECTION:
internetweek.jp.    86400 IN    A       210.199.223.86
```

dig(続き)

- +traceオプションで再帰的検索をシミュレートできる。
  - BIND9のdigだけなので、BIND8とBIND9が両方インストールされている環境では注意。
    - which digで確認
    - full pathで起動
  - あくまでdigがシミュレートしているので、namedのキャッシュの状態などは反映されない。

dig(続き)

```
kohi@thomas[8]% dig +trace internetweek.jp

; <<>> DiG 9.3.0 <<>> +trace internetweek.jp
;; global options:  printcmd
.                       518134  IN      NS      J.ROOT-SERVERS.NET.
.                       518134  IN      NS      K.ROOT-SERVERS.NET.
.                       518134  IN      NS      L.ROOT-SERVERS.NET.
.                       518134  IN      NS      M.ROOT-SERVERS.NET.
.                       518134  IN      NS      A.ROOT-SERVERS.NET.
.                       518134  IN      NS      B.ROOT-SERVERS.NET.
.                       518134  IN      NS      C.ROOT-SERVERS.NET.
.                       518134  IN      NS      D.ROOT-SERVERS.NET.
```

dig(続き)

```
.                       518134  IN      NS      E.ROOT-SERVERS.NET.
.                       518134  IN      NS      F.ROOT-SERVERS.NET.
.                       518134  IN      NS      G.ROOT-SERVERS.NET.
.                       518134  IN      NS      H.ROOT-SERVERS.NET.
.                       518134  IN      NS      I.ROOT-SERVERS.NET.
;; Received 436 bytes from 127.0.0.1#53(127.0.0.1) in 11 ms

jp.                     172800  IN      NS      F.DNS.jp.
jp.                     172800  IN      NS      D.DNS.jp.
jp.                     172800  IN      NS      E.DNS.jp.
jp.                     172800  IN      NS      A.DNS.jp.
jp.                     172800  IN      NS      B.DNS.jp.
jp.                     172800  IN      NS      C.DNS.jp.
;; Received 229 bytes from 192.36.148.17#53(I.ROOT-SERVERS.NET)
in 336 ms
```


dig(続き)

```
internetweek.jp.        86400   IN      NS      ns2.nic.ad.jp.
internetweek.jp.        86400   IN      NS      ns1.nic.ad.jp.
;; Received 108 bytes from 2001:2f8:0:100::153#53(F.DNS.jp) in
20 ms

internetweek.jp.        86400   IN      A       210.199.223.86
internetweek.jp.        86400   IN      NS      ns1.nic.ad.jp.
internetweek.jp.        86400   IN      NS      ns2.nic.ad.jp.
;; Received 124 bytes from 202.12.30.133#53(ns2.nic.ad.jp) in 9
ms
```

dnswalk

- 正索き/逆索きの整合性や文字セットなどをチェックするツール。
- http://www.visi.com/~barr/dnswalk/
- perlスクリプト。
  - Net::DNSモジュールが必要。
- 今のところv6はサポートしていない。
- cronで実行し、ログのサマリーと一緒にメールすると便利。
  - 不可避な警告やエラーが出るなら、前日の結果とdiffを取ると抑制できる。

ありがちだが間違った質問

- Q:「上位のネームサーバを教えて下さい。」
- A:「ありません。」
  - DNSはrootから下降検索するのに、自分からrootへ向かって上昇検索するという誤解。
    - でもどこに設定するつもりだろう?
    - (RFC1035でいう)resolverのネームサーバでISPのネームサーバにforwardersを向ける…とか?

ありがちだが注意を要する質問

- Q:「slaveのIPアドレスを教えてください。」
- A(昔):「お客様の設定には不要です。」
  - 自分のところのゾーンデータにslaveのホストのAを書こうとしている。
- A(今):「どういう目的にご利用ですか?」
  - allow-transfer{}の設定には必要。
  - 「内部名/外部名」対策かも。
    - ごめんなさい。今日は割愛します。
    - IW2002 DNS Dayの森下さんのプレゼン参照。
    - http://www.nic.ad.jp/ja/materials/iw/2002/main/dns/PM1-morishita.pdf

ありがちだが注意を要する質問(続き)

- 開示するならアドレスは死守。
- 将来の保証ができないなら、「自分でAを索いてown riskで設定して下さい。」
- いずれにしてもよく事情を聞いてから。

rndc(BIND9)

◆動作中のnamedを制御するコマンド。

```
thomas# rndc -help
rndc: illegal option -- h
Usage: rndc [-c config] [-s server] [-p port]
        [-k key-file ] [-y key] [-V] command

command is one of the following:

  reload        Reload configuration file and zones.
  reload zone [class [view]]
                Reload a single zone.
  refresh zone [class [view]]
                Schedule immediate maintenance for a zone.
  retransfer zone [class [view]]
                Retransfer a single zone without checking serial
number.
  freeze zone [class [view]]
                Suspend updates to a dynamic zone.
```


rndc(BIND9:続き)

```
  thaw zone [class [view]]
                Enable updates to a frozen dynamic zone and
  reload it.
    reconfig    Reload configuration file and new zones only.
    stats       Write server statistics to the statistics file.
    querylog    Toggle query logging.
    dumpdb      Dump cache(s) to the dump file (named_dump.db).
    stop        Save pending updates to master files and stop
  the server.
    stop -p     Save pending updates to master files and stop
  the server
                reporting process id.
```

rndc(BIND9:続き)

```
   halt          Stop the server without saving pending updates.
  halt -p        Stop the server without saving pending updates
reporting
                 process id.
  trace          Increment debugging level by one.
  trace level    Change the debugging level.
  notrace        Set debugging level to 0.
  flush          Flushes all of the server's caches.
  flush [view]   Flushes the server's cache for a view.
  flushname name [view]
                 Flush the given name from the server's cache(s)
  status         Display status of the server.
```


rndc(BIND9:続き)

```
recursing     Dump the queries that are currently recursing
(named.recursing)
  *restart      Restart the server.

* == not yet implemented
Version: 9.3.0
```

rndc(BIND9:続き)

- reload
  - 設定ファイル群の読み直し。
- reload [zone [class [view]]]
  - ゾーン単位で読み直し。
    - reload example.co.jp 152/29.7.16.172.in-addr.arpaはダメ
      - 152/29.7.16.in-addr.arpaはclassと解釈される。
- reconfig
  - named.confを読み直し。
  - 追加されたゾーンだけ読み込み。
- flush
  - キャッシュを消去。
- flushname name [view]
  - 名前単位でキャッシュから消去

ndc(BIND8)

- 動作中のnamedを制御するコマンド

```
thomas# ndc
Type   help  -or-   /h   if you need help.
ndc> help
(builtin) start - start the server
(builtin) restart - stop server if any, start a new one
getpid
status
stop
exec
reload [zone] ...
reconfig [-noexpired] (just sees new/gone zones)
dumpdb
```

ndc(BIND8:続き)

```
stats [clear]
trace [level]
notrace
querylog
qrylog
help
quit
args
```


DNSの運用

namedが動き出してから

DNSの運用

- DNSの運用
  - ネームサーバの運用
  - 周囲との連携
    - 運用開始
    - 設定の変更
    - ちゃんと動いていないとき

DNSの運用(続き)

- 運用開始時
  - 上位ゾーンからauthorityの委任を受ける。
    - Internet Registryデータベースと連動。
    - 学内、社内の担当部署へ依頼。
    - 必要事項
      - ゾーン名
      - ホスト名(master、slaveとも)
      - glueが必要ならIPアドレス
    - 自分が設定した通りの内容で登録依頼する。
      - NS RRの設定はthomas.example.co.jpなのに、ns.example.co.jpと登録してはいけない。

DNSの運用(続き)

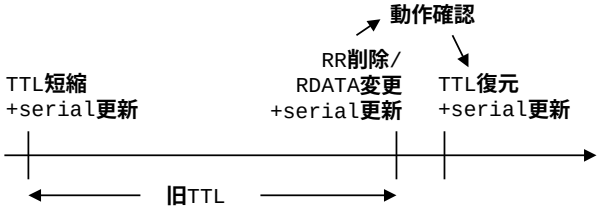
- slaveを依頼する。
  - 必要事項
    - 依頼するゾーン名
    - masterのIPアドレス
  - slaveのホスト名を教えてもらう。
    - NS RRを記述
- RFC2317の設定を依頼する。
  - ゾーン名はISP/学内、社内の担当部署から指示を受ける。
  - 必要事項
    - ホスト名(master、slaveとも)

DNSの運用(続き)

- 設定の変更
  - 単にRRを追加するだけ。
    - 間違えないように追加すればよい。
  - RRの変更、削除
    - 事前にTTLを短くしておかないと、古いデータがよそのキャッシュに残ってしまう。
  - いずれの場合もserialの変更を忘れずに。
    - 変更がslaveに伝播しないことがある。
      - タイミング、ネットワーク上の位置、時の運などで新しいデータが返ってきたり古いデータが返ってきたり。

DNSの運用(続き)

- いずれの場合も新しいデータを読み込んだらすぐチェック。
  - RDATAを間違えたデータがよそのキャッシュに載ってしまうとやっかい。
  - slaveについては、NOTIFYのおかげで昔より気が楽になった。



DNSの運用(続き)

- 子供のゾーンの追加
  - NSと必要に応じてglueのAを追加。
- slaveになる。
  - named.confに設定を追加。
- いずれも
  - 設定を追加。
  - rndc reload
  - 動作確認。

```
zone "branch.example.co.jp" {
        type slave;
        masters {
                192.168.0.1;
        };
        file "branch.example.co.jp";
};
```

DNSの運用(続き)

- ネームサーバのリナンバー、別ホストへの載せ替え
  - 上位ゾーンでもNS RRやglueを変更してもらう。
    - Internet Registryデータベースの更新と連動
    - 学内、社内の担当部署への依頼
    - ちゃんとしないとlame delegationの原因に。
  - slaveには参照するmasterを変更してもらう。
    - 更新が伝播しない。
    - やがてexpire。

DNSの運用(続き)

- 自分のところのネームサーバはちゃんと動いているか?
  - ログにエラーは出ていないか?
  - authorityを持っているはずのゾーンのRRをno-recurseで索いてみる。
    - dig +norecurse *domain.name*
    - authoritative answerか?
    - レスポンスをMRTGでプロットしておくと性能劣化の目安になるかも。

DNSの運用(続き)

- よそのネームサーバもちゃんと動いているか?
  - 自分がmasterの場合
    - slaveはちゃんとauthoritative answerを返しているか?
    - serialは一致しているか?
      - 更新直後のrefreshの間は、ずれていても可。
    - 更新したらちゃんとゾーン転送に来るか?
      - NOTIFYを聴いているslaveならすぐ来る。
      - そうでなければrefreshまでの間に。

```
xfer-out: info: client 10.12.34.56#1425: transfer of
'example.co.jp/IN': AXFR started

security: error: client 10.12.34.56#2073: zone
transfer 'example.co.jp/IN' denied
```


DNSの運用(続き)

- 自分がslaveの場合
  - ちゃんとserialをチェックできているか?
    - ダメでも要注意だが即問題ではない。
      - 下位層の問題(例えばpingが通らない)であれば様子見。
      - DNS的な問題なら対応開始。
  - expireしていないか?
    - していたらダメ。
      - しちゃう前に見つける。

```
general: debug 1: refresh_callback: zone
example.co.jp/IN: serial: new 2004120201, old
2004120201

general: info: zone example.co.jp/IN: refresh:
failure trying master 172.16.7.153#53: timed out

general: info: zone example.co.jp/IN: refresh: retry
limit for master 172.16.7.153#53 exceeded
```

slaveが動かない

- IP reachabilityは大丈夫か?
  - 下位層に問題があったら、いくらDNSを追いかけてもムダ。
- ルータやファイアウォールでのフィルタリングは間違っていないか?
  - DNSのパケットがsrc/destとも53番で固定だったのはBIND4までの話。
  - BIND8からはactive open側は任意の番号。

slaveが動かない(続き)

- allow-transfer{}でslaveからのアクセスまで拒否していないか?
  - 運用開始後、セキュリティ設定を強化したときは要注意。

```
security: error: client 10.12.34.56#2073:
 zone transfer 'example.co.jp/IN' denied

xfer-in: error: transfer of 'example.co.jp/IN' from
172.16.7.153#53: failed while receiving responses:
REFUSED
```

slaveが動かない(続き)

- ゾーン名は間違っていないか?
  - RFC2317の逆索きだとありがち。
    - 最初は正しく設定したのに、わざわざ/24相当のゾーンに直してしまう人が少なからず居る ;_;
- masterはauthorityをなくしていないか?
  - 設定変更後は即座にチェック。
  - digの出力のflagsにaaは含まれているか?

DNSが動かない:事例

- お客様からサポート要請
  - 「自分のドメインはアクセスできるが、外がアクセスできない。」
- 回線/ルーティング障害?? すわ一大事!!
- 詳しくヒアリングしてみると
  - 中からはauthorityを持っている名前は索けるが、外部の名前が索けない。
  - 外からはそのサーバが索けない。
- 実際にアクセスしてみると
  - '.'のNSが索けない。

DNSが動かない:事例(続き)

- 推測
  - named.rootが悪い?
    - 正しく入手したものだった。
    - 外部から索けないことの説明がつかない。
- 結論
  - BIND4時代の知識でファイアウォールで53番同士のパケットしか通してなかった。
    - 使っていたのはBIND8。
    - query-source port 53;を仮に設定してもらい切り分け。

Advanced topics

お品書き

◆MXとMTA
◆アクセス制限
◆logging
◆ログの監査
◆大規模ネームサーバ構築tips
◆ロバスト性の向上
◆設定ファイルの履歴管理
◆ルータに関する登録
◆組織間の接続点に関する登録
◆-t(chroot)オプション
◆-u(setuid)オプション

MXとMTA

◆ MX RR
  ◆ そのドメイン宛のメールをどのホストに配送すればよいかの指定。
◆ MXを書けば、MTAがそのドメイン宛のメールを受領できるようになるわけではない。
  ◆ MTAでもそれ相応の設定が必要。
    ◆local configuration errorなどでメールを紛失。
◆ 1つのドメインに複数のMXを設定できる。
  ◆ preferenceが小さいほど優先。

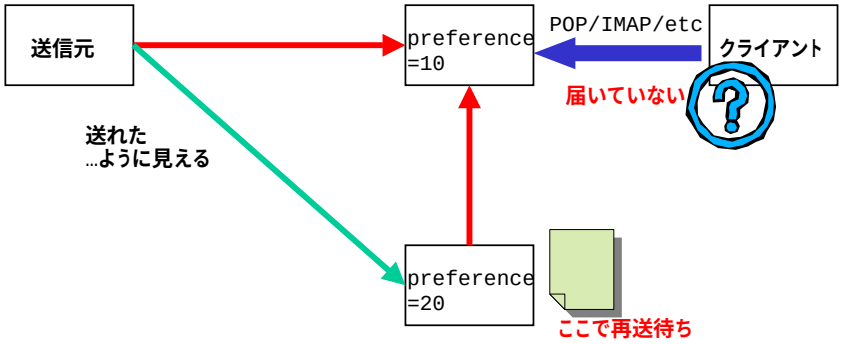
MXとMTA(続き)

- バックアップMXに配送されても、送信元では無事送れたように見えてしまうことが多い。
  - トラブル時のトレースが困難。
- 不用意なホストを記述すると、エラーになる。
  - 直前のスライド。
  - MXを向ける以上はMTAもきちんと設定する。
  - 断りなくよそのサーバにMXを向けてはいけない。
    - ISPのサーバにMXを向ける顧客…。
    - 昔はただの(?)不正使用だった。
    - 今は普通third party relayでハネられる。

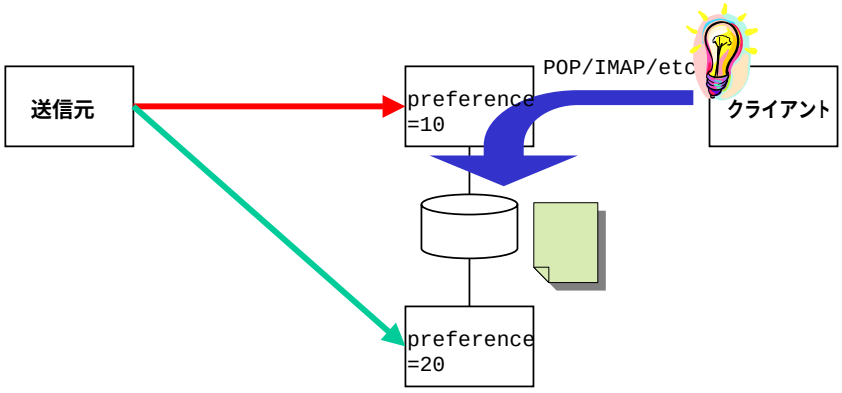
MXとMTA(続き)

- MXを複数記述するなら、設定の次元ではなく設計の次元から考えなければいけない。
  - DNSだけでなくmailについても理解が必要。
- ローカル配送しないでmailboxが復旧するまでspoolするだけのバックアップMXは不要では?
  - RAIDストレージにmailboxを置き複数ホストで共有して…とか
  - 外に見せるMXを複数台用意して内部のmailboxへstatic配送、という設計では有効。
- 深い議論は安藤一憲さんのチュートリアルで :-)

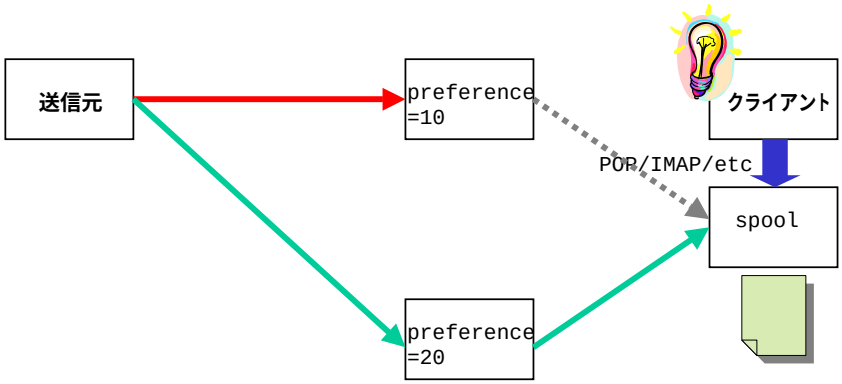
MXとMTA(続き)



MXとMTA(続き)



MXとMTA(続き)



$GENERATE

◆ゾーンデータ中に

```
$GENERATE 193-254 dhcp$ A  172.16.7.$
```

と書くと

```
dhcp193  A  172.16.7.193
dhcp194  A  172.16.7.194
       :
dhcp254  A  172.16.7.254
```

に展開される。

$GENERATE(続き)

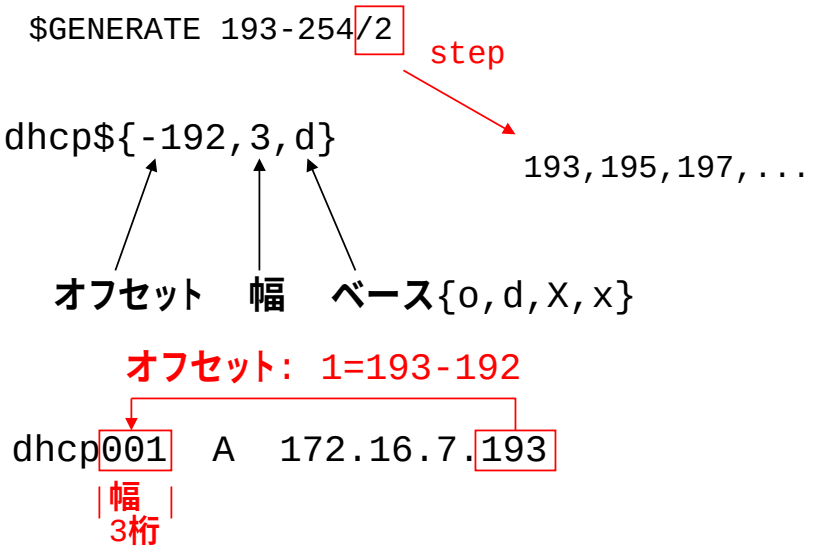
```
$GENERATE 193-253/2 dhcp${-192,3,d} A 172.16.7.$
```

と書くと

```
dhcp001  A  172.16.7.193
dhcp003  A  172.16.7.195
      :
dhcp061  A  172.16.7.253
```

に展開される。

$GENERATE(続き)



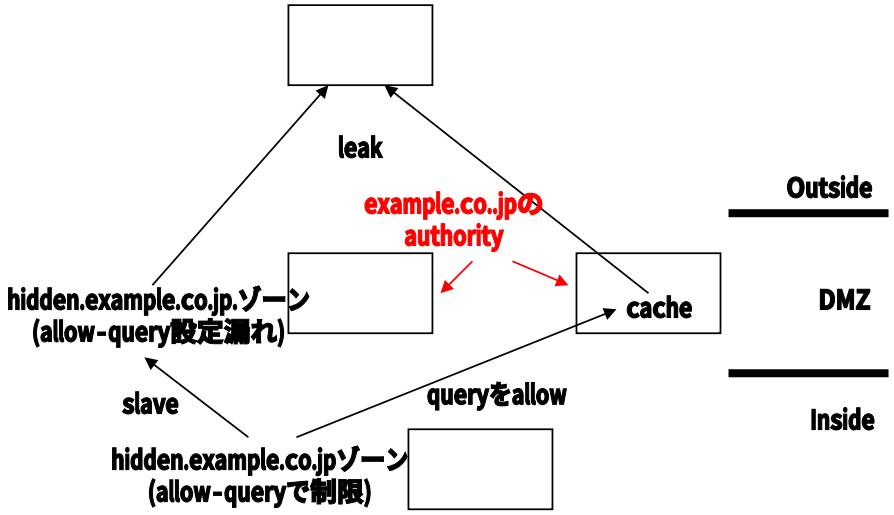
アクセス制限

- そのネームサーバの役割を明確にする。
  - (RFC1035でいう)resolverか?
    - サービスを提供すべき範囲は?
  - 何かのゾーンのauthorityか?
- ツール
  - allow-query
  - allow-transfer
  - allow-recursion
  - recursion

allow-query

- query全般のアクセス制限。
  - options{}とzone{}に書ける。
    - zone{}に書いた方が強い。
  - 不正使用の拒否。
  - ファイアウォールの内側など、外に見せたくないゾーン。
- queryを許可しているホストのネームサーバのキャッシュやslaveを経由して外に漏れることも。
  - パズルをがんばる。

allow-queryに関する失敗例



allow-transfer

- ゾーン転送のアクセス制限。
  - allow-query同様、options{}とzone{}に書ける。
- slaveには許可しないとダメ。
  - 最初からダメなら見つけやすいが…
  - 最初O.K.でも失敗が続くとexpireしてしまう。
    - 動き出してから設定を強化するときは要注意。
- 自分にも許可しておかないと動作確認をするときに不便。
  - localhostというaclが組み込みで定義されている。
    - が、v6アドレスは含まれない。
- Brute Forceには無力だがエレガントなアタックを試みる輩には提供しているサービスなどのヒントを与えてしまう。
  - slaveでも適切に設定。

allow-transfer(続き)

- 小さなコマンドで大きな仕事をさせられる。
  - DoSアタックのツールとしては有効。
- 通常のqueryでもdatagramに乗り切らないとTCPにfallbackするので、TCPコネクション自体を拒否するわけではない。
  - SYN flood予防にはならない。
- 関連項目
  - options{transfer-out *N*};
    - 同時に受け付けるゾーン転送の本数の上限。
  - options{tcp-clients *N*}:
    - 同時に受け付けるTCPコネクションの本数の上限。

アクセス制限(続き)

- resolver
  - 正当な範囲だけにallow-recursionすればいい。
    - allow-queryという意見もある。
- authority
  - recursion offでよい。
  - localhost.とその逆索きはサービス不要。
  - 各ゾーンはslaveだけにallow-transfer。

logging

- named.confのloggingの設定。
- category
  - namedが出すログ情報のカテゴリ。
    - database,security,configなど
  - category毎に送出するchannelを割当。
    - 複数可
  - 明示的な設定のないカテゴリのメッセージはdefaultという名前で設定されているchannelへ振り分けられる。
- channel
  - ログ情報の送出先。
    - 送出先はfile,syslog,stderr,null
    - ファイル名、facilityなどを指定。

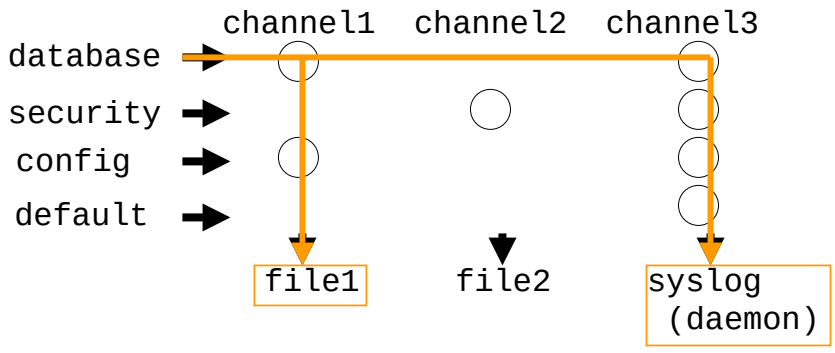
logging(続き)

```
logging {
        channel channel1 {
                file “file1”;
        };
        channel channel2 {
                file “file2”;
        };
        channel channel3 {
                syslog daemon;
                severity debug;
        };
```

logging(続き)

```
        category database {
                channel1;
                channel3;
        };
        category security {
                channel2;
                channel3;
        };
        category config {
                channel1;
                channel3;
        };
        category default {
                channel3;
        };
};
```


logging(続き)



logging(続き)

- namedが起動してからnamed.confの解析が済むまでのログ情報はsyslog(daemon)に送られる。
  - BIND9.3.0からコンパイル時の設定により、デフォルトのfacilityをdaemon以外に変更できるようになった。README参照。
  - BIND8は8.2.3で同様の機構が導入されている。
- logging{}で別のchannelを指定していても起動直後のログはそのchannelには出ない。

logging(続き)

- named.confに以下のような設定を入れると、メッセージにcategory、severityが一緒に記録される。
  - ログ採取のチューニングに便利。

```
logging {
        channel to_syslog {
                syslog local3;
                severity info;
                print-category yes;
                print-severity yes;
        };
        category default {
                to_syslog;
        };
};
```

ログの監査

- ログにはさまざまな情報が出ている。
  - ちゃんと見ることは重要。
- でもあまり量が多いと、ちゃんと見るのも大変。
  - 重要なメッセージが埋もれてしまう。
- その解決策を根性論に立脚した肉体労働に見出すのは不毛。
  - せっかく計算機を使ってるんだから…。

ログの監査(続き)

- なぜ量が多くなるか?
  - 同じ内容のメッセージが繰り返し出ている。
  - 正常動作の報告が出ている。
- ではどうする?
  - severityは必要に応じて調整。
  - nullを活用。
  - サマリーを作成してメールでレポート。
    - 不審な点は生ログをチェック。
    - もっとリアルタイムな監視は矢萩さんのチュートリアルで:-)

ログの監査(続き)

- サマリーの作成
  - #!/usr/bin/perl(sedでもrubyでも…)
    - タイムスタンプを削除。
    - pidを削除。
      - fileで採取するとつかないが、syslog経由だとつく。
    - active open側ソケットのポート番号を削除。
    - 正常動作に関するメッセージを削除。
  - | sort | uniq | mail
- これでいわゆる「S/N比」(signal/noise)は向上する。

大規模ネームサーバ構築tips

- 多くのゾーンをサービスするネームサーバ
  - 必然的に取り扱うファイルも増える。
    - ゾーンファイル

```
zone "example.co.jp" IN {
     file “slave/reg/e/example.co.jp”;
          :
};
```

      - ゾーン名でハッシュ。
      - “ja/co/example.co.jp”は多分、愚か。
      - 1ディレクトリに存在するファイルを減らすことによりnamei()が高速化され、namedの起動が速くなる、かどうかは知らない :-)
      - 人間の視認性は向上。

大規模ネームサーバ構築tips(続き)

- named.confも大きくなる。

```
include "conf/slave/reg"
include "conf/slave/v4inv"
include "conf/slave/v6inv"
```

  - 1ファイルの寸法を小さく留める。
    - 編集時の扱いやすさ。
    - ロバスト性向上にも貢献。

ロバスト性の向上

- 目指すこと
  - 壊れにくいように。
  - 壊れちゃったらすばやく直せるように。

ロバスト性の向上(続き)

- ファイルの分類
  - 取り返し/あきらめがつくファイルv.s.つかないファイル。
    - ログ
      - なくても動作する。
    - slaveのdumpファイル
      - 建前はmasterから取って来れば復活するはず。
        - ところがmasterが既にダメになっていることも…(^^;。
    - masterのゾーンファイル
    - configファイル
      - がーん ;_;

ロバスト性の向上(続き)

- アクセスの激しいファイル
  - ログ
    - named関連のファイルでは一番アクセスが激しいのでは?
  - slaveのdumpファイル
    - 知らないうちに更新される。
  - masterのゾーンファイル
  - configファイル
    - 設定変更や再起動時ぐらい。
  - アクセスが激しければ、その分、ディスクの負担も大きい。

ロバスト性の向上(続き)

- 取り返し/あきらめがつく/つかない
- アクセスの激しさ
  - 分類の結果は一致。
- /varをわける。
  - /,swap,おしまい、はダメ。
  - クラッシュ時の被害範囲の局所化。
  - 危険要素の閉じ込め。
- ログは普通/varの下。
- slaveのdumpファイルは/varの下に配置。
  - 絶対パスで指定するとか、symbolic linkを使うとか。

設定ファイルの履歴管理

- メリット
  - 不思議な設定に出会ったときの手がかり。
  - 編集ミスで壊したときの復旧の種。
  - 設定ミスの影響範囲の同定。
  - 返金沙汰になったときに、誰の給料を天引きすればいいか。
- デメリット
  - まぁ面倒といえば面倒だが…

設定ファイルの履歴管理(続き)

- RCSで管理。
- SCCSに愛があればSCCSでもよい。
- 極めて個人的見解だが、CVSはちょっと…
  - 設定ファイルはソースと異なり排他制御も重要な要素。
    - CVSもロックを有効にする設定あり。
  - 1つのレポジトリを複数箇所にcheckoutできる。
    - namedが参照しているリビジョンとレポジトリの内容が不一致だと困る。

ルータに関する登録

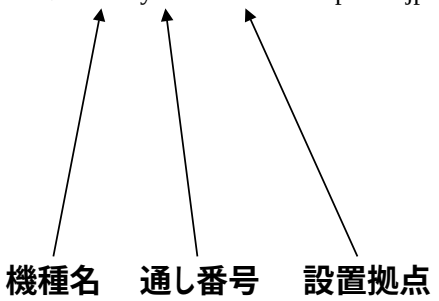
- ルータもDNSに登録しておかないと、tracerouteしたときにIPアドレスしか出てこない。
- でもあまり情報を書きすぎるのはセキュリティ上どうなのか?
  - 機種名
    - 機種依存のセキュリティホール
  - 回線品目
    - DoSアタックの効き目
    - :

ルータに関する登録(続き)

- 1台のルータでもインターフェース毎に別の名前がついていることも多い。
- DNSでは名前に / は使えない。
  - so-6/0/0 ->so6-0-0
- どうしても数が多くなるので機械的な命名則がないと破綻する。
- U.S.の大きなISPは拠点名にIATAの空港コードをつけるのが好きらしい。
  - sea,sjc,nyc…
  - 大手町にあってもnrt!

ルータに関する登録(続き)

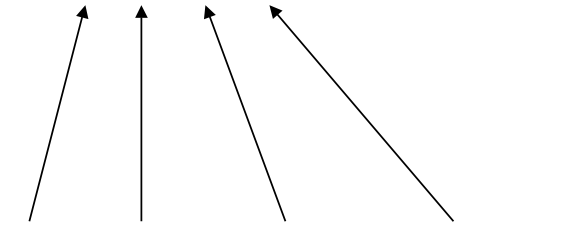
- 例1
  - foundry2.otemachi.example.ad.jp



ルータに関する登録(続き)

◆ 例2

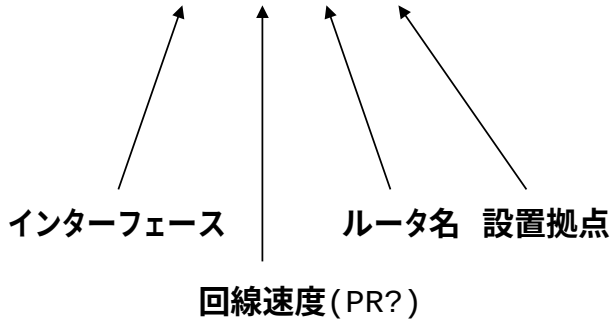
◆ sjc3-nrt3-stm4-2.sjc3.example.net



ルータに関する登録(続き)

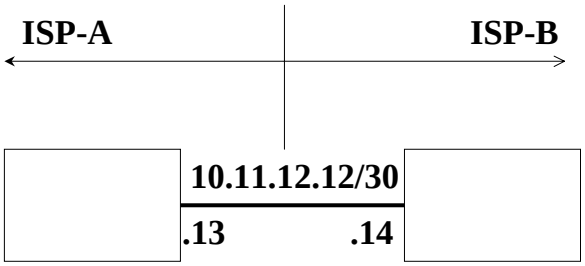
◆ 例3

◆ so6-0-0-2488M.br2.PAO2.example.net



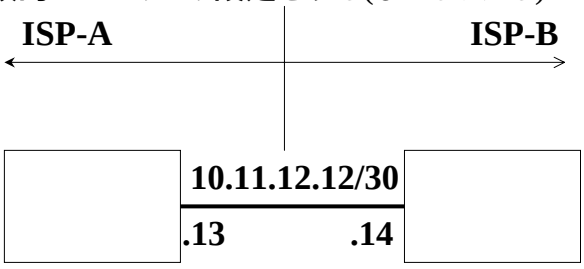
組織間の接続点に関する登録

- ISP-AとISP-Bの接続点
  - ISP-Aがアドレスを割当
  - 10.11.12.12/30
    - ISP-A: 10.11.12.13
    - ISP-B: 10.11.12.14



組織間の接続点に関する登録(続き)

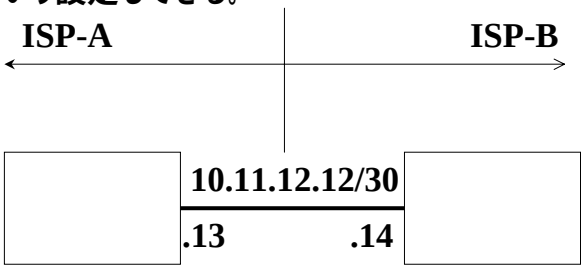
- 経験的にはこういう設定をする(してもらえる)ことが多い。




```
$ORIGIN 12.11.10.in-addr.arpa.
      :
13 PTR ge1-2.router.jp.isp-a.net.
14 PTR isp-b.peer.isp-a.net.
```

組織間の接続点に関する登録(続き)

◆ こういう設定もできる。




```
$ORIGIN 12.11.10.in-addr.arpa.
        :
13 PTR ge1-2.router.jp.isp-a.net.
14 NS  ns.isp-b.ad.jp.
```

4octet(/32)に対応する委任

```
$ORIGIN 14.12.11.10.in-addr.arpa.
@ IN SOA (
        :
        )
  NS  ns.isp-b.ad.jp.
  PTR ge3-0.router.isp-b.ad.jp.
```

4octet(/32)に対応するゾーン

-t(chroot) オプション

◆ named -t
- ◆ いわゆるsandboxにchroot()して動作させる。
- ◆ namedのプロセスを不正に操作されたときに、ファイルシステム中のアクセスできる範囲を制限することによりセキュリティを向上させる。

-t(chroot)オプション(続き)

- BIND8のnamedでslaveをするときは、内部からnamed-xferを起動する。
  - sandboxの下にnamed-xferをインストール。
  - ライブラリをダイナミックリンクするOSなら、共有ライブラリもsandboxの下に。
  - 必要なファイルはlddで調べる。
  - あるいはスタティックリンク。

-t(chroot)オプション(続き)

- ログはfileチャネルでとる。
- あるいはsyslogdが*$SANDBOX*/var/run/logを聴くように工夫。
  - FreeBSDならsyslogd -l、OpenBSDは-aらしい。
  - その種のオプションがなければ別プロセスのsyslogd。
  - BIND8は最初からログがとれない。
  - BIND9は
    - named.confをパースし終えた時点(loggingの設定があるとき)
    - rndc reloadなどを実行した時点(loggingの設定がないとき)

    からとれなくなる。
- /dev/randomをmknodする。
- cp /etc/localtime *$SANDBOX*/etc
  - FreeBSDの場合。他のOSについてはFAQ参照。
  - ログの時刻がUTCになってしまう。

-t(chroot)オプション(続き)

- 「/varをわける」をどう実現するか?
  - *BSDならnullfs?
    - でもコードはメンテナンスされてないみたい…。
  - *BSD用力技
    - FDDにnewfs
    - raw deviceをddで読んでUFSイメージを/varに書き出す。
    - vnconfigして、*$SANDBOX*/varにmount。
  - 他のOSでは…(未検証)
    - *$SANDBOX*は/varの下に構築。
    - ln –s /etc/RCS *$SANDBOX*/etc
    - *$SANDBOX*/etc/RCS/named.conf,vの実体は/varの外に。
    - CVSを使えばもうちょっとエレガントかも。
    - mount localhost:/var *$SANDBOX*/varなんてのもあり?

-u(setuid)オプション

- named -u
  - UIDを変更して動作させる。
  - rootの特権を放棄して、namedのプロセスを不正に操作されたときに行われ得る操作の内容を制限することによりセキュリティを向上させる。
  - 起動時はrootがnamed.confを読めないといけない。
  - (r)ndc reloadなどを実行するときは、setuid後のユーザがnamed.confを読めないといけない。
    - パーミッションに注意。
    - rndc.keyも。

further readings

- RFC1033
  - DOMAIN ADMINISTRATORS OPERTIONS GUIDE
- RFC1035
  - DOMAIN NAMES – IMPLEMENTATION AND SPECIFICATIONS
- RFC1912
  - Common DNS Operational and Configuration Errors

further readings(続き)

- RFC2308
  - Negative Caching of DNS Queries(DNS NCACHE)
- RFC2317
  - Classless IN-ADDR.ARPA delegation
- BIND9 Administrator Refence Manual
  - BIND9.xのソースのdoc/arm(XML,HTML版)
  - http://www.nominum.com/resources/documentation/Bv9ARM.pdf(PDF版)
- BIND9によるDNSサーバの構築と運用
  - http://www.nic.ad.jp/ja/materials/iw/2003/proceedings/T11.pdf
  - 昨年のこのチュートリアルの資料
  - 今回の資料より設定例が充実していますので、ご高覧を。

訂正

◆ 当日、会場でお配りした資料には以下の箇所に誤りがありました。このファイルでは緑色の文字で修正済みですが、SOIのビデオの音声では口頭で訂正していますので、混乱のないよう、ご注意下さい。

	誤	正
#62	hostmaster@example.jp	hostmaster@example.co.jp
#105〜#107	file "empty";	file "master/empty";
#152,#153	スライドの順序が入れ替わっていました。	