T7: IPv6 実践講座～トラブルシューティング、セキュリティ、アプリ構築まで～

3. IPv6 プログラミング

スクリプト言語と IPv6 – 2012 –

2012/11/20

関根 佳直

本セッションの概要

2012年11月現在のスクリプト言語の IPv6 対応状況を, 実例を交えながら概観します
("スクリプト言語での IPv6 対応ネットワークプログラミングの解説" ではありません)

本セッションで扱うプログラミング言語

- Perl http://www.perl.org/
- PHP http://www.php.net/
- Python http://www.python.org/

C/C++ での IPv6 に対応したソケットプログラミングについては, IW 2011 の加藤さんの大変素晴らしい資料があるので, そちらを参照する等してください
(下記ページよりダウンロード可能。その他の資料もおすすめです)

"プロトコル非依存のソケットプログラミング基礎" (加藤淳也@NTT)
http://www.nic.ad.jp/ja/materials/iw/2011/proceedings/t5/

プログラミング言語の IPv6 対応とは？

プログラミング言語に求められる機能が多様化している今日においては，以下のような様々な要素を考慮する必要があり，**一言で “○○は IPv6 に対応している” と言うのは (現時点では) 難しい状況となっている**

“プログラミング言語の IPv6 対応” で考慮すべき要素

- **ソケット**
  - IPv6 用 (プロトコルファミリが PF_INET6) のソケットを扱えるか (= 狭義の IPv6 対応)
- **名前解決 (DNS)**
  - ホスト名から IPv6 アドレスが引けるか (正引き)
  - IPv6 アドレスからホスト名が (簡単に) 引けるか (逆引き)
- **各種 (L7) ネットワークプロトコル**
  - IPv6 で Web アクセス (HTTP) やメール送信 (SMTP) 等が行えるか
- **その他**
  - なくても IPv6 での通信はできるが，あると便利なもの
  - 例) アドレス表記の変換ができるか

おことわり

本セッションでは, Perl, PHP, Python について, 前記の項目の IPv6 対応状況を概説します

– おことわり –

- 各言語の対応状況や特性によって, 内容・量に偏りがあります (問題がある部分について重点的に説明)
- クライアントサイドのみ扱います
- “バージョン x から対応” といった表現では, 基本的に開発版は対象として含めていません

時間がないので駆け足で行きます

予備知識: ポリシーテーブル

ポリシーテーブルとは？

- 使用するアドレス (≒プロトコル) を選択するための仕組み
- この設定により IPv6 を優先したり, 逆に IPv4 を優先したりすることができる (getaddrinfo(3) が返すアドレスの順番が変わる)

定義

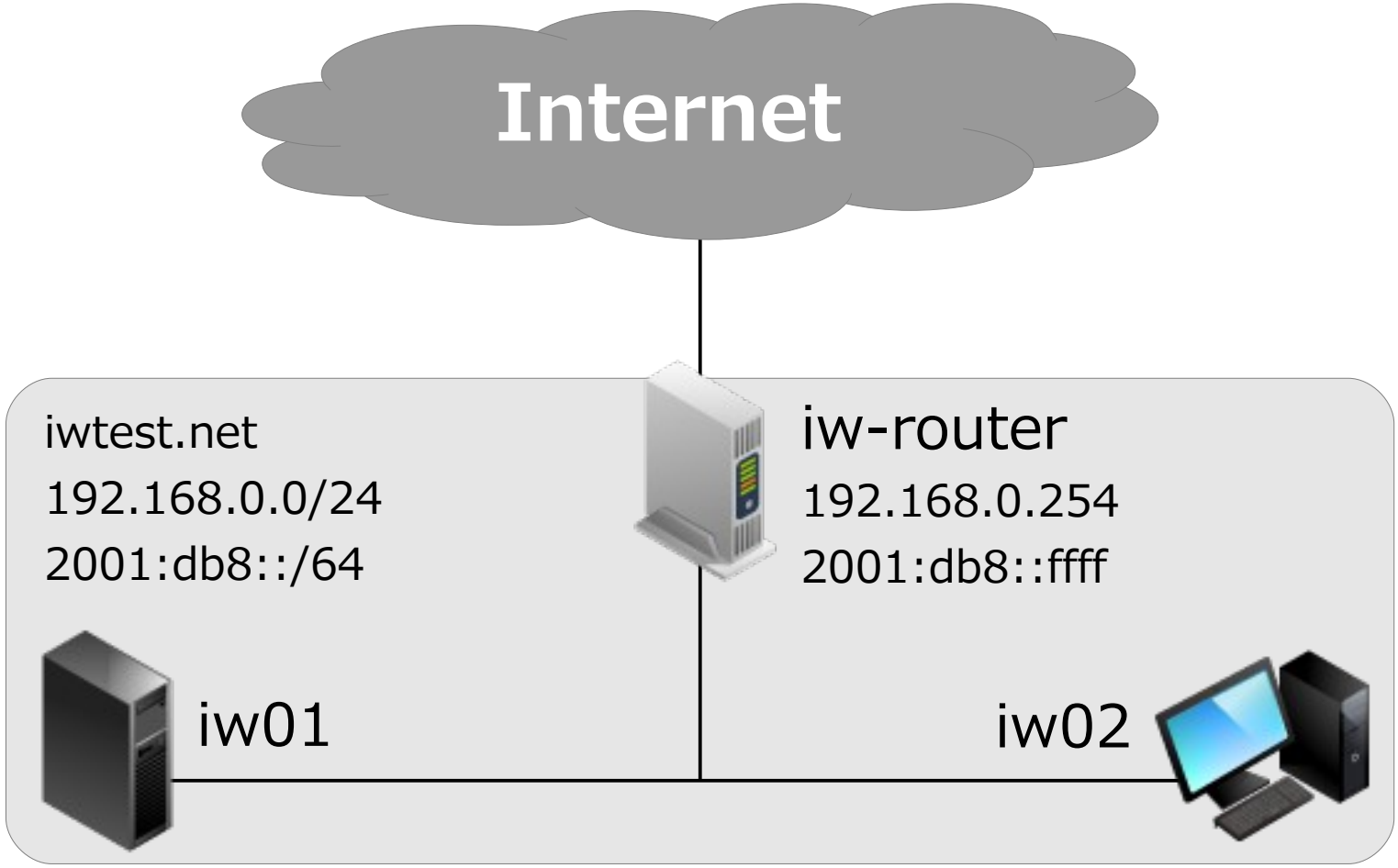
- RFC 3484 “Default Address Selection for Internet Protocol version 6 (IPv6)” (2003/02)
- RFC 6724 (2012/09) でアップデート
- デフォルトポリシーテーブルは IPv6 優先

設定

- Linux: ip addrlabel {add|del} および /etc/gai.conf
- FreeBSD: ip6addrctl {add|delete} および /etc/ip6addrctl.conf
- Windows: netsh interface ipv6 {add|delete} prefixpolicy

最近の OS のデフォルト設定は IPv6 優先と考えてよい

検証環境



サーバ (DNS authoritative & cache / Web / メール)
192.168.0.1
2001:db8::1

クライアント
192.168.0.2
2001:db8::2

http://iconhoihoi.oops.jp/ のアイコンを使用させて頂きました

検証環境の DNS 設定

iwtest.net のゾーンファイル (抜粋)

```
@           IN   NS     iw01
            IN   MX     10 iw01

iw01        IN   A      192.168.0.1
            IN   AAAA   2001:db8::1
iw01-v4     IN   A      192.168.0.1
iw01-v6     IN   AAAA   2001:db8::1

iw02        IN   A      192.168.0.2
            IN   AAAA   2001:db8::2
iw02-v4     IN   A      192.168.0.2
iw02-v6     IN   AAAA   2001:db8::2
```

検証方法

HTTP

- サーバ側はアドレスベースのバーチャルホストにより, IPv4 アドレスと IPv6 アドレスで異なったコンテンツを表示するように設定する
- http://iw01.iwtest.net/ の表示結果により, 対象のコードがどちらのプロトコルで接続したかが分かる
- http://iw01-v6.iwtest.net/ を表示できるかどうかで, 対象のコードが IPv6 に対応しているかが分かる

SMTP

- メールヘッダの Received および MTA のログから, 対象のコードがどちらのプロトコルを使用してメールを送信したかが分かる
- 接続先のホストに iw01-v6 を指定してメールを送信できるかどうかで, 対象のコードが IPv6 に対応しているかが分かる

Perl

Perl

Perl のバージョン

最新版: 5.16.2 (2012/11/01)

Perl とネットワークプログラミング

標準ライブラリ (コアモジュール) で基本的なネットワークプログラミングが可能 (ソケット, HTTP クライアント, SMTP クライアント)
その他の機能が欲しい場合は, CPAN[*1] のモジュール等を使用する

[*1] Comprehensive Perl Archive Network http://www.cpan.org/

Perl と IPv6

長らく言語本体の対応が遅れていたが, **Perl 5.14 から (ようやく) 本格的に IPv6 をサポート**

Perl and IPv6 – Perl supports IPv6
http://www.perl.org/about/whitepapers/perl-ipv6.html

それより前のバージョンでも, CPAN モジュールを利用すれば IPv6 を使うことは可能となっている

Perl の IPv6 対応状況概略

ソケット

コアモジュールの Socket は 5.10 から部分的に対応, **5.14 でフル対応**。CPAN モジュールにも対応しているものがある

名前解決

対応 (Socket::getaddrinfo(), Socket::getnameinfo(), CPAN Net::DNS)

HTTP クライアント

標準では非対応 (コアモジュール HTTP::Tiny。LWP 等のメジャーなモジュールも非対応)

SMTP クライアント

標準では非対応 (コアモジュール Net::SMTP)

IPv6 アドレスの処理

CPAN モジュールにより対応 (CPAN Net::IP)

Linux での Perl のバージョン

主要 Linux ディストリビューションのパッケージでインストールされる Perl のバージョン (2012/11/19 現在)

ディストリビューション		Perl のバージョン
RHEL[*1] / CentOS	5.8	5.8.8
	6.3	5.10.1
Fedora	16	5.14.3
	17	5.14.3
Debian	5.0.10	5.10.0
	6.0.6	5.10.1
Ubuntu	11.10	5.12.4
	12.04.1 LTS[*2]	5.14.2

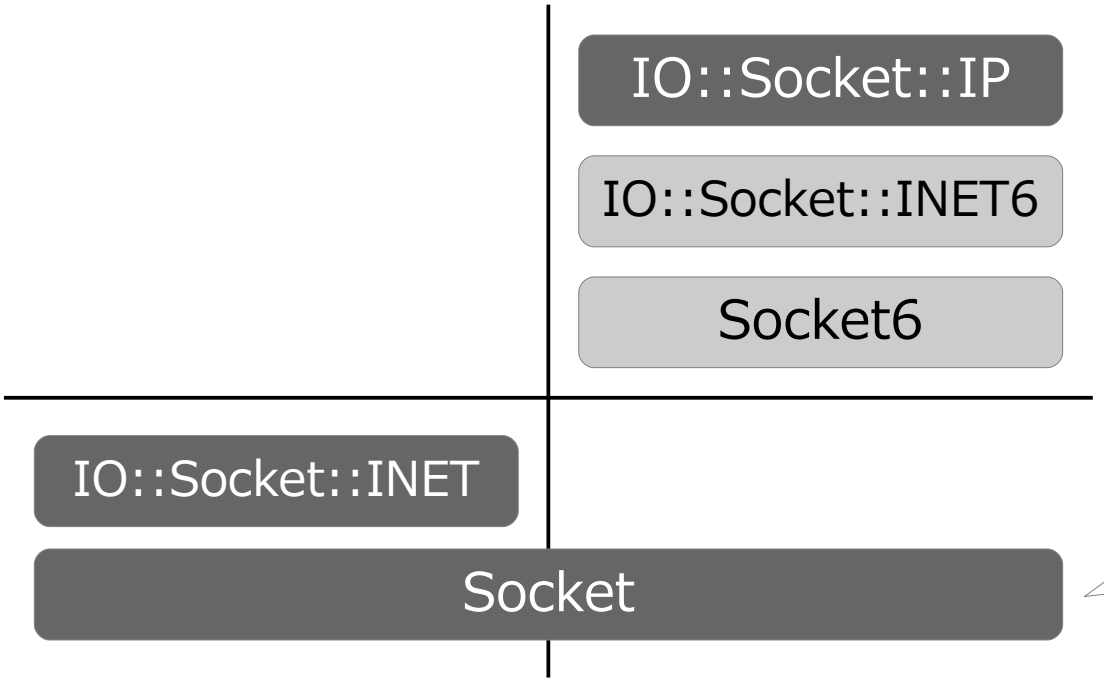
※ パッケージ名はいずれも “perl”

※ ソースからビルドすれば, どのディストリビューションでも最新版を含む任意のバージョンをインストール可能

[*1] Red Hat Enterprise Linux　[*2] long-term support

Perl のソケット

“TMTOWTDI[*1]” のフレーズ通り, ソケットに関しても色々なモジュールが存在し, 他の言語に比べて複雑な状況 (■ が取り上げるもの)



[*1] “*There's more than one way to do it*”, 同じことをするのに何通りものやり方があるという Perl のモットー

ソケット (Socket)

BSD ソケットインタフェースを提供するコアモジュール

Perl 5.14 付属の Socket 1.94 から IPv6 フルサポート

- 定数 (AF_INET6, IN6ADDR_ANY, IN6ADDR_LOOPBACK, …)
- 名前解決 (getaddrinfo(), getnameinfo())
- アドレス表現の変換 (inet_pton(), inet_ntop())
- 構造の変換 (pack_sockaddr_in6(), unpack_sockaddr_in6())

IPv6 関連のものは, ほとんどがデフォルトでエクスポートされないことに注意 (明示的にインポートする必要あり)

Perl 5.14 より前の Socket でも IPv6 を使えないわけではないが, あまり現実的ではない (CPAN の Socket6 を使った方が便利)

- Perl 5.10.x では AF_INET6 / PF_INET6 の定義があるだけ
- Perl 5.12.x では inet_pton() / inet_ntop() が追加されただけ

ソケット (IO::Socket::INET)

コアモジュール (Perl 5.6.0～)
抽象度が高いため, Socket より簡単にソケットを扱える

しかし, “*Object interface for AF_INET domain sockets*”
であるため, **IPv4 しか扱えない**

多くのネットワーク系モジュールが IO::Socket::INET を使用しているが, それらのモジュールも当然ながら IPv6 非対応となっている (対応方法は後述)

Perl の IPv6 対応が遅れている元凶と言えなくもない...

ソケット (IO::Socket::IP)

IPv4 / IPv6 に対応したソケットインタフェースを提供する CPAN モジュール

http://search.cpan.org/dist/IO-Socket-IP/

IO::Socket::INET の置き換えとして設計されており (*"A drop-in replacement for IO::Socket::INET"*), コンストラクタやメソッドは互換性がある (一部例外あり)

今後, IO::Socket::INET でやっていたことをやりたい場合は, このモジュールを使うのがよい

IO::Socket::INET と IO::Socket::IP の比較

IO::Socket::INET と IO::Socket::IP による TCP クライアントの例 (*$host* の *$port* に TCP で接続)

IO::Socket::INET

```
use IO::Socket::INET;
:
my $sock = IO::Socket::INET->new(
    PeerAddr => $host,
    PeerPort => $port,
    Proto    => 'tcp'
) or die “Error: $!\n”;
:
```

IO::Socket::IP

```
use IO::Socket::IP;
:
my $sock = IO::Socket::IP->new(
    PeerAddr => $host,
    PeerPort => $port,
    Proto    => 'tcp'
) or die “Error: $!\n”;
:
```

赤字の部分 (use およびコンストラクタ) を変更するだけで IPv4 専用だったコードが IPv4 / IPv6 両対応になる (はず)

(変更後のプロトコルの優先順位はポリシーテーブルの設定に従う)

※ もちろん, IPv4 アドレスが直書きしてあるような部分については, 別途対応する必要あり

DNS (Net::DNS)

Net::DNS

DNS リゾルバ (CPAN モジュール)

http://search.cpan.org/dist/Net-DNS/

IPv6 関連 RR の検索に対応

- IPv6 関連の RR (AAAA, IPv6 アドレスの PTR) は問題なく引ける
- AAAA を引いた結果の文字列表現は :: による省略がされない (Net::DNS::RR の print() 等)
- IP アドレスはそのままの形式で逆引きできる (in-addr.arpa. / ip6.arpa. 形式にする必要がない)
- IPv6 アドレスを逆引きするときは :: で省略したアドレスを渡すことも可能

DNS (Net::DNS) 続き

```
use Net::DNS;
:
my $r = Net::DNS::Resolver->new;
# 名前 $name の RR $type を検索
my $result = $r->search($name, $type);
foreach my $i ($result->answer) {
    print $i->rdatastr, "¥n";
}
```

```
# $name='internetweek.jp', $type='aaaa' の場合
2001:dc2:1000:2006:0:0:c0:ffee
```

(正引きの結果は :: による省略なし, leading-zero なし)

```
# $name='2001:dc2:1000:2006::c0:ffee', $type='ptr' の場合
internetweek.jp.
```

(IP アドレスはそのままの形式で逆引きができ, :: で省略しても問題なし)

HTTP クライアント

メジャーどころ*は IPv6 非対応　*自分の中での認識です

HTTP::Tiny

非常にシンプルな HTTP クライアント (Perl 5.13.9 からコアモジュール)

IPv6 非対応 (IO::Socket::INET を使用しているため)

HTTP::Lite (CPAN)

軽量な HTTP クライアント

http://search.cpan.org/dist/HTTP-Lite/

IPv6 非対応 (socket() に PF_INET を渡しているため)

LWP::UserAgent (CPAN)

多機能な HTTP クライアント

http://search.cpan.org/dist/libwww-perl/

IPv6 非対応 (内部で使用している Net::HTTP が IO::Socket::INET のサブクラスのため)

SMTP クライアント (Net::SMTP)

Net::SMTP

SMTP クライアント (Perl 5.7.3 からコアモジュール)

IPv6 非対応 (IO::Socket::INET のサブクラスのため)

接続先のホストに...

- **デュアルスタックのホスト[*1]を指定 → IPv4 で接続する**
- **IPv6 のみのホスト[*2]を指定 → エラーになる**
- **localhost を指定 → その先の動作は環境[*3]に依存する**

[*1] A レコード / AAAA レコードの両方を持つホストという意味

[*2] AAAA レコードしか持たないホストという意味。IPv6 アドレスを直に指定してもエラーになる

[*3] ホストに IPv6 の接続性があるか, MTA が IPv6 を使用する設定になっているか。これらを満たしていれば IPv6 で送信することも可能

IPv6 アドレスの処理 (Net::IP)

Net::IP

IPv4 / IPv6 アドレス処理のための様々な機能を提供する

CPAN モジュール

http://search.cpan.org/dist/Net-IP/

次のようなメソッドを提供する

- version()
  IP のバージョンを返す (4 or 6)
- ip()
  IPv6 アドレスの場合, 最も冗長な表現を返す
- short()
  できるだけ省略された表記を返す
- reverse_ip()
  逆引き用の表記 (PTR レコードの形式) を返す

IPv6 アドレスの処理 (Net::IP) 続き

```
use Net::IP;
my $address = new Net::IP("2001:DB8::0123:0045:0006:07");
```

処理対象の IP (v6) アドレス

```
# IP のバージョン
print $address->version();
# 6

# 最も冗長な表記
print $address->ip();
# 2001:0db8:0000:0000:0123:0045:0006:0007

# 最も省略された表記
print $address->short();
# 2001:db8::123:45:6:7

# 逆引き用の表記
print $address->reverse_ip();
# 7.0.0.0.6.0.0.0.5.4.0.0.3.2.1.0.0.0.0.0.0.0.0.0.0.0.0.0.8.b.d.0.1.0.0.2.ip6.arpa.
```

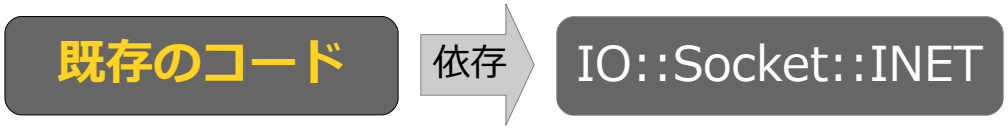
IPv6 アドレスのアルファベットは小文字になる

IO::Socket::INET 依存コードの IPv6 対応

IO::Socket::INET を使用しているコードを IPv6 に対応させるにはどうすればよいか？

IO::Socket::INET に依存しているコードが...

- **自分で修正可能なコードの場合** (直接使用しているような場合)



→ **IO::Socket::IP を使うように修正する**

- **自分で修正したくないコードの場合** (CPAN モジュール等)

→ **Net::INET6Glue を使う**

※ もちろん, IPv4 アドレスが直書きしてあるような部分については, 別途対応する必要あり

Net::INET6Glue による IPv6 対応

Net::INET6Glue

IO::Socket::INET6[*1] からシンボルテーブルを IO::Socket::INET にコピーすることで, IO::Socket::INET を IO::Socket::INET6 のように動作させる CPAN モジュール

(詳細は Net::INET6Glue::INET_is_INET6.pm を参照)

http://search.cpan.org/dist/Net-INET6Glue/

使い方

IO::Socket::INET に依存した CPAN モジュール等を使用している既存のコードの先頭で, “use Net::INET6Glue;” するだけ

[*1] IPv4 / IPv6 に対応したソケットインタフェースを提供する CPAN モジュール

Net::INET6Glue による IPv6 対応 続き

LWP や Net::SMTP 等の IO::Socket::INET 依存モジュールを使用したプログラムで, Net::INET6Glue により IPv6 での通信ができるようになったことを確認済み

```
# HTTP::Tiny (IO::Socket::INET 依存 = IPv4 専用) を使用した
# コードを Net::INET6Glue によって IPv6 に対応させる例

use Net::INET6Glue;                    ← これを追加するだけ
use HTTP::Tiny;
:
my $http = HTTP::Tiny->new;
my $response = $http->get($url);
print $response->{content};
```

既存のコード

Net::INET6Glue を使用した場合のプロトコルの優先順位は, システムのポリシーテーブルの設定に従う

Perl まとめ

- **Perl 本体 (コアモジュール Socket) の IPv6 (フル) 対応は Perl 5.14 から**
- **IO::Socket::INET および IO::Socket::INET 依存モジュール/コードは IPv6 非対応**
- **今後は IO::Socket::INET ではなく IO::Socket::IP を使うようにする**
- **IO::Socket::INET 依存のプログラムを手っ取り早く IPv6 に対応させたい場合は Net::INET6Glue が便利**

PHP

PHP

PHP のバージョン

5.4系が最新系列だが, 5.3系もメンテナンスされている

5.4系の最新版: 5.4.8 (2012/10/18)

5.3系の最新版: 5.3.18 (2012/10/18)

PHP とネットワークプログラミング

標準ライブラリで非常に広範囲なネットワークプログラミングが可能

その他の機能が欲しい場合は, 拡張ライブラリ PEAR[*1] のパッケージ等を使用する

[*1] PHP Extension and Application Repository http://pear.php.net/

PHP と IPv6

PHP 5 から IPv6 に対応

http://www.php.net/ChangeLog-5.php (Version 5.0.0 Beta 1 のところ)

PHP の IPv6 対応状況概略

ソケット

対応 (inet_pton(), inet_ntop() は PHP 5.1.0 以降)

名前解決

対応 (dns_get_record(), gethostbyaddr() / PEAR Net_DNS2)

HTTP クライアント

対応 (各種ファイル関数, cURL 等)

SMTP クライアント

対応 (PEAR Net_SMTP) / システムの環境依存 (mail(), PEAR Mail)

IPv6 アドレスの処理

拡張パッケージにより対応 (PEAR Net_IPv6)

Linux での PHP のバージョン

主要 Linux ディストリビューションのパッケージでインストールされる PHP のバージョン (2012/11/19 現在)

ディストリビューション		PHP のバージョン[*1]
RHEL / CentOS	5.8	5.1.6 (php) / 5.3.3 (php53)
	6.3	5.3.3 (php)
Fedora	16	5.3.18 (php)
	17	5.4.8 (php)
Debian	5.0.10	5.2.6 (php5)
	6.0.6	5.3.3 (php5)
Ubuntu	11.10	5.3.6 (php5)
	12.04.1 LTS	5.3.10 (php5)

※ ソースからビルドすれば, どのディストリビューションでも最新版を含む任意のバージョンをインストール可能

[*1] ( ) 内はパッケージ名

DNS (標準ライブラリ)

dns_get_record(*$hostname* [, *$type* = DNS_ANY])

引数で指定した RR の情報を取得して, 配列で返す

AAAA レコードの検索に対応

IPv6 アドレスの逆引きも可能だが, ip6.arpa. 形式で指定する必要があるため, gethostbyaddr() を使う方が便利

```
// internetweek.jp の IPv6 アドレス (AAAA レコード) を検索
$result = dns_get_record('internetweek.jp', DNS_AAAA);
// $result は次のような配列になる
// array(
//   array(
//     "host"  => "internetweek.jp",
//     "type"  => "AAAA",
//     "ipv6"  => "2001:dc2:1000:2006::c0:ffee",
//     "class" => "IN",
//     "ttl"   => 300
//   )
// )
```

AAAA を検索するために指定する定数

DNS (標準ライブラリ) 続き

gethostbyaddr(*$ip_address*)

指定した IP アドレスに対応するホスト名を返す
名前は IPv4 専用という感じがする[*1]が, **IPv6 アドレスの逆引きもできる**[*2] (:: で省略したアドレスを渡すことも可能)

```
echo gethostbyaddr('192.41.192.130');
// internetweek.jp
```

```
echo gethostbyaddr('2001:dc2:1000:2006::c0:ffee');
// internetweek.jp
```

[*1] RFC 2133 により, 最近の (?) gethostbyaddr(3) は IPv6 アドレスも扱えるらしい

[*2] PHP のソースコードの ext/standard/dns.c で定義されている php_gethostbyaddr() の中で, IPv6 アドレスも扱えるように記述されていることを確認済み

ファイル関数による HTTP アクセス

PHP では, fopen() で open するファイル名として URL を指定して, ファイルのように取り扱うことができる[*1]

サポートするプロトコル/ラッパー

http://jp1.php.net/manual/ja/wrappers.php

ファイル関数による HTTP アクセスは IPv6 対応

(URL に IPv6 アドレスを使用する場合は [2001:db8::1] 形式で指定)

fopen() の他にも, 同じ fopen_wrappers を使用している file() (ファイル全体を配列として返す関数) や file_get_contents() (ファイル全体を文字列として返す関数) も同様に IPv6 対応

[*1] allow_url_fopen ディレクティブで fopen_wrappers を有効にしている場合

ファイル関数による HTTP アクセス 続き

以下はいずれも https://internetweek.jp/ の内容を取得して表示する例

fopen()

```
$fh = fopen('https://internetweek.jp/', 'r');
if ($fh === FALSE) {
    die(“cannot open URL\n”);
}
while (($line = fgets($fh)) !== FALSE) {
    echo $line;
}
fclose($fh);
```

file()

```
$contents = file('https://[2001:dc2:1000:2006::c0:ffee]/'); // IPv6 アドレス は [] で囲む
if ($contents === FALSE) {
    die(“cannot open URL\n”);
}
foreach ($contents as $line) {
    echo $line;
}
```

file_get_contents()

```
$contents = file_get_contents('https://internetweek.jp/');
if ($contents === FALSE) {
    die("cannot open URL\n");
}
echo $contents;
```

HTTP クライアント (cURL)

libcurl を使用し各種プロトコルでのデータ転送を行う

IPv6 対応

```
// URL の内容を文字列として取得する例
$ch = curl_init($url);
// 取得した内容をブラウザに渡さない
curl_setopt($ch, CURLOPT_RETURNTRANSFER, TRUE);
$result = curl_exec($ch);
:
// 処理
:
curl_close($ch);
```

メール送信

mail()

システムの MTA を使用してメールを送信するビルトイン関数

したがって, **動作はシステムの環境[*1]に依存**

Net_SMTP (PEAR) http://pear.php.net/package/Net_SMTP

low-level の処理が可能な SMTP クライアント

IPv6 対応 (ポリシーテーブルには従わないことに注意)

Mail (PEAR) http://pear.php.net/package/Mail

送信に使用するバックエンドを次の中から選択可能なパッケージ

- mail() 関数
- システムの sendmail
- ダイレクト SMTP 接続 (Net_SMTP を使用)

したがって, **動作はバックエンド依存になる**

[*1] ホストに IPv6 の接続性があるか, MTA が IPv6 を使用する設定になっているか。これらを満たしていれば IPv6 で送信することも可能

メール送信 (Net_SMTP)

接続先のホストに...

- **デュアルスタックのホスト[*1]を指定**
  → **IPv4 で接続する** (依存している Net_Socket の実装による)
- **IPv6 のみのホスト[*2]を指定**
  → **IPv6 で接続する** (IPv6 アドレスを指定する場合は [2001:db8::1] 形式で指定すれば OK)
- **localhost を指定**
  → **その先の動作はシステムの環境[*3]に依存する**

[*1] A レコード / AAAA レコードの両方が登録されているホストという意味
[*2] AAAA レコードしか登録されていないホストという意味
[*3] ホストに IPv6 の接続性があるか, MTA が IPv6 を使用する設定になっているか。これらを満たしていれば IPv6 で送信することも可能

IPv6 アドレスの処理 (Net_IPv6)

Net_IPv6

IPv6 アドレスに関する処理を行う PEAR パッケージ

http://pear.php.net/package/Net_IPv6

checkIPv6(*$address*)

引数が IPv6 アドレスの表記として正しいかを boolean で返す

```
// :: が2ケ所ある誤った表記
checkIPv6('2001:db8::1::1');
// FALSE
```

compress(*$address*, *$force* = FALSE)

できるだけ省略した表記を返す

```
echo compress('2001:db8:0:0:0:0:0:1');
// 2001:db8::1

echo compress('2001:db8:0:0:0123:0045:0006:07');
// 2001:db8::123:45:6:7
```

IPv6 アドレスの処理 (Net_IPv6) 続き

uncompress(*$address*, *$leading_zero* = FALSE)

省略されたアドレスを冗長な表記にして返す

第2引数を TRUE にすると, leading-zero が付与され, 各コロンの間が4文字の固定長になる

```
// leading-zero なし
echo uncompress('2001:dc2:1000:2006::c0:ffee');
// 2001:dc2:1000:2006:0:0:c0:ffee

// leading-zero あり
echo uncompress('2001:dc2:1000:2006::c0:ffee', TRUE);
// 2001:0dc2:1000:2006:0000:0000:00c0:ffee
```

PHP まとめ

PHP は 5 以降 IPv6 対応

※ 紹介した範囲外では対応していないものがあるかもしれません

Python

Python

Python のバージョン

2系と3系の両方が使用されている

2系の最新版: 2.7.3 (2012/04/09)

3系の最新版: 3.3.0 (2012/09/29)

Python とネットワークプログラミング

標準ライブラリで, 各種ネットワークプログラミングが可能 (ソケット, HTTP クライアント/サーバ, SMTP クライアント/サーバ etc.)

その他の機能が欲しい場合は, 拡張ライブラリ PyPI[*1] のパッケージ等を使用する

[*1] Python Package Index http://pypi.python.org/pypi

Python と IPv6

Python 2.2 から IPv6 に対応

http://docs.python.org/release/2.2.3/whatsnew/node10.html

Python の IPv6 対応状況概略

ソケット

対応 (socket)

名前解決

対応 (socket.getaddrinfo(), socket.getnameinfo())

HTTP クライアント

対応 (urllib, httplib (Python 2) / urllib.request, http.client (Python 3))

SMTP クライアント

対応 (smtplib)

IPv6 アドレスの処理

拡張モジュールにより対応 (PyPI の IPy 等)

Linux での Python のバージョン

主要 Linux ディストリビューションのパッケージでインストールされる Python のバージョン (2012/11/19 現在)

ディストリビューション		Python のバージョン[*1]
RHEL / CentOS	5.8	2.4.3 (python)
	6.3	2.6.6 (python)
Fedora	16	2.7.3 (python) / 3.2.3 (python3)
	17	2.7.3 (python) / 3.2.3 (python3)
Debian	5.0.10	2.4.6 (python2.4) / 2.5.2 (python)
	6.0.6	2.6.6 (python) / 3.1.3 (python3)
Ubuntu	11.10	2.7.2 (python) / 3.2.2 (python3)
	12.04.1 LTS	2.7.3 (python) / 3.2.3 (python3)

※ ソースからビルドすれば, どのディストリビューションでも最新版を含む任意のバージョンをインストール可能

[*1] ( ) 内はパッケージ名

ソケット (socket)

BSD ソケットインタフェースを提供するモジュール

IPv6 対応 (Python 2.2 以降)

- socket オブジェクト
  ソケットの作成や送受信関連のメソッドを提供
- getaddrinfo(), getnameinfo()
  プロトコル非依存の名前解決 (Python 2.2 以降)
- inet_pton(), inet_ntop()
  IPv4 / IPv6 アドレスの文字列/バイナリ変換 (Python 2.3 以降)
- has_ipv6
  プラットフォームで IPv6 がサポートされているかを表す真偽値定数 (Python 2.3 以降)

ソケット (socket) 続き

getaddrinfo(*host*, *port*, *family*=0, *socktype*=0, *proto*=0, *flags*=0)

host: ホスト名 or IP アドレス (or None)

port: サービス名 or ポート番号 (or None)

family: アドレスファミリ (IPv4 と IPv6 のどちらを使うか)。デフォルトは socket.AF_UNSPEC (プロトコルを限定しない)

```
# internetweek.jp に HTTP 接続するための情報を取得 (IPv4/IPv6 問わず)
socket.getaddrinfo('internetweek.jp', 'http', socket.AF_UNSPEC,
    socket.SOCK_STREAM)
# [(10, 1, 6, '', ('2001:dc2:1000:2006::c0:ffee', 80, 0, 0)),
   (2, 1, 6, '', ('192.41.192.130', 80))]

# internetweek.jp に HTTP 接続するための情報を取得 (IPv6 限定)
socket.getaddrinfo('internetweek.jp', 80, socket.AF_INET6,
    socket.SOCK_STREAM)
# [(10, 1, 6, '', ('2001:dc2:1000:2006::c0:ffee', 80, 0, 0))]
```

※ family が socket.AF_UNSPEC の場合に IPv4 アドレスと IPv6 アドレスのどちらが先に来るかは, システムのポリシーテーブルの設定による

ソケット (socket) 続き

host の *port* に getaddrinfo() が返す結果の順に接続を試行する例

IPv4 / IPv6 を限定しない

```
import socket
import sys
:
s = None
for res in socket.getaddrinfo(host, port, socket.AF_UNSPEC, socket.SOCK_STREAM):
    af, socktype, proto, canonname, sockaddr = res
    try:
        s = socket.socket(af, socktype, proto) # ソケット生成
    except socket.error as msg:                # エラーが発生した場合は
        s = None
        continue                               # 次の候補へ
    try:
        s.connect(sockaddr)      # sockaddr で示されるリモートのソケットに接続
    except socket.error as msg: # エラーが発生した場合は
        s.close()                # ソケットを閉じて
        s = None
        continue                 # 次の候補へ
    break                        # 接続に成功した場合は, ループを抜けて以降の処理に移る
if s is None
    sys.exit("cannot connect to " + host + ":" + port)
:
```

HTTP アクセス

urllib (Python 2) / urllib.request (Python 3)

high-level な HTTP アクセスを提供するモジュール

IPv6 対応

(URL に IPv6 アドレスを使用する場合は [2001:db8::1] 形式で指定する)

httplib (Python 2) / http.client (Python 3)

low-level な HTTP アクセスを提供するモジュール

(上記の urllib / urllib.request はこのモジュールを使用している)

IPv6 対応

(接続先ホストに IPv6 アドレスを使用する場合は [2001:db8::1] 形式で指定する)

メール送信

smtplib

SMTP クライアント

IPv6 対応

接続先のホストに localhost を指定した場合は, その先の動作はシステムの環境[*1]に依存する

```
# メール送信の例
import smtplib
:
server = smtplib.SMTP(host)
server.sendmail(from_addr, to_addr, msg)
server.quit
```

接続先ホスト

[*1] ホストに IPv6 の接続性があるか, MTA が IPv6 を使用する設定になっているか。これらを満たしていれば IPv6 で送信することも可能

IP アドレスの処理 (IPy)

IPy

IPv4 / IPv6 アドレス処理のための様々な機能を提供する PyPI パッケージ

http://pypi.python.org/pypi/IPy/

次のようなメソッドを提供する

- strCompressed() / strNormal() / strFullsize()

  それぞれ, できるだけ省略した表記 / :: による省略なしの表記 / 最も冗長な表記 を返す

- reverseName()

  逆引き用の表記 (PTR レコードの形式) を返す

- iptype()

  アドレスの種類を表す文字列を返す

IP アドレスの処理 (IPy) 続き

```
>>> import IPy
>>> i = IPy.IP('2001:0DC2:1000:2006::0C0:FfEe')
```

わざと不自然な表記にした internetweek.jp の IPv6 アドレス

```
>>> i.version()
6

>>> i.strCompressed()
'2001:dc2:1000:2006::c0:ffee'

>>> i.strNormal()
'2001:dc2:1000:2006:0:0:c0:ffee'

>>> i.strFullsize()
'2001:0dc2:1000:2006:0000:0000:00c0:ffee'

>>> i.reverseName()
'e.e.f.f.0.c.0.0.0.0.0.0.0.0.0.0.6.0.0.2.0.0.0.1.2.c.d.0.1.0.0.2.ip6.arpa.'

>>> i.iptype()
'ALLOCATED APNIC'
```

IPv6 アドレスのアルファベットは小文字になる

Python まとめ

Python は 2.2 以降 IPv6 対応

※ 紹介した範囲外では対応していないものがあるかもしれません

スクリプト言語と IPv6 –2012– まとめ

Perl, PHP, Python は IPv6 ready!

あとは使い手の

皆さん次第です