

@INC にみる Perl のやりかたがいっぱい

tag [perl](#)

こんにちは。最近では PHP ばかり書いている、永遠の Perl 初心者 [hatyuki](#) です。
とつぜん質問ですが、みなさんが最も多くみている Perl のエラーはなんですか？
自分の場合は、う〜ん。。。。

```
Can't locate Hoge.pm in @INC (@INC contains: ~~~~ ).  
BEGIN failed--compilation aborted.
```

かな？いつまでも進歩がないのが伺えますね！

皆さんご存知の通り、このエラーは "@INC" で指定されたディレクトリの中にモジュールが見つからなかった場合に発生するエラーです。つまり、モジュールをインストールし忘れているか、モジュールがどこに置いてあるのかを適切に設定していないか、の (およそ) どちらかですね。さてさて、Casual Perler な皆さんは、このエラーを回避するためにどんな方法を使って @INC にサーチパスを追加していますか？

というわけで、今日はサーチパスを追加する方法について。です。

PHP で例えるなら

```
<?php  
set_include_path( get_include_path() . PATH_SEPARATOR . dirname(__FILE__) . '/lib' );
```

をいかに ~~た〜さんの不思議な方法~~ で書くかということですね。

それについて調べてみたところ、なんとなく「Perl (の特徴) っぽいなあ」と感じる部分がありましたので、それがお伝えできればと思います。

とはいえ、Casual Track ですので、まずはモジュールについて紹介していきたいと思います。

'use lib' を使ったベーシックな方法

```
use lib './lib';
```

たぶんきっと、一番ベーシックな方法ではないでしょうか。タイプ数も少ないですし！

今でも書き捨てる script では、こんな感じで書いている方も多いのでは？

しかし、この方法だと './lib' と相対パスで指定しているため、実行する環境によってはうまく動作しないため、長く使う script などではあまりよい方法ではなさそうですね。

また、絶対パスで指定した場合でも script の置き場をかえるたびに書き換えが必要になり、管理が煩雑になりますよね。

FindBin を使うよ

```
use FindBin;  
use lib "$FindBin::Bin/lib";
```

[FindBin](#) というモジュールを使うと、実行中のファイルが置いてあるディレクトリまでの絶対パスを得ることができます。

これを使って、サーチパスを追加する方法です。

[miyagawa](#) さんの [Remedie](#) や、[Plagger](#) といったアプリケーションも、この方法でサーチパスを設定していました。

Advent Calendar でほかの方が書かれているサンプルスクリプトの中でも何回か登場しているようで、比較的メジャーな方法のように思います。

- remedie からの抜粋

```
use FindBin;

my $base_dir;
BEGIN { $base_dir = "$FindBin::Bin/.." }

use lib "$base_dir/extlib"; # to load local::lib
use local::lib "$base_dir/cpanlib";
use lib "$base_dir/lib", "$base_dir/extlib"; # to prefer extlib for URI::Fetch etc.
```

FindBin を使った場合は、実行中のファイルのパスを得ることができるため、サーチパスを追加するだけでなく、設定ファイルや、その他ファイルのパスを指定する必要がある場合にも利用できますね。

また、FindBin は Perl 5.00307 から標準モジュールになったようですので、ほとんどの環境で CPAN からのインストールなしに利用できるのもメリットですね。

ちなみに、モジュールが標準モジュールかどうかを調べるときは `corelist` というコマンドを使うとべんりですね！

```
$ corelist FindBin
FindBin was first released with perl 5.00307
```

ただし、この `corelist (Module::CoreList)` が標準モジュールとなったのが 5.8.9 からのようです。。。

FindBin::libs で楽したいな

```
use FindBin::libs;
# local::lib 用のディレクトリはざんねんながら。。。
```

CPAN からダウンロードできるモジュールで [FindBin::libs](#) というモジュールがあります。

このモジュールを利用すると、それっぽいディレクトリを勝手に探して*[1](#)、勝手にサーチパスに追加してくれます。

相当いじわるなディレクトリ構成にしない限りは、確実に見つけ出してくれるためとっても便利です。

最近は書き捨てる script では `common::sense` とともに、よくつかっています。

```
#!/activeperl
use common::sense;
use FindBin::libs;

use MyApp::Anachronism;
```

標準モジュールではないせいなのか、公開されている script ではあまり使われていないように思います。が、OneShot な script では便利なモジュールだと思いますので、ぜひ使ってみてください。

それ Project::Libs でできるよ

FindBin::libs では残念ながら local::lib 用のディレクトリをサーチパスに追加することができませんが、[antipop](#) さんが[最近作成](#)された [Project::Libs](#) なら、その辺も含めてよしなにしてくれるようです。

基本的には、以下のように書くだけ。

```
use Project::Libs;
```

local::lib を使っている場合には Project::Libs が便利そうですね。

正直まだあまり使ったことがないため、詳しくは [antipop](#) さんの記事をご覧ください。

plackup の甘い罠

FindBin / FindBin::libs / Project::Libs というモジュールを紹介しましたが、これらのモジュールを plackup から起動するような PSGI スクリプトで利用すると、ちょっとした罠にはまります。あれ？はまったのは私だけですか？

たとえば、以下のように app.psgi 書き

```
use FindBin;
warn $FindBin::Bin;

my $app = sub {
    return [
        200,
        [ 'Content-Type' => 'text/plain' ],
        [ 'Hello World!' ],
    ];
};

$app;
```

これを plackup で実行してみると

```
$ plackup app.psgi
/usr/bin at app.psgi line 2.
```

結果、\$FindBin::Bin の値は plackup がインストールされているディレクトリとなってしまいます。[*2](#)

当然この状態で 'use lib' をしても正しくサーチパスが設定されないため "Can't locate" を拝む結果となります。

特殊リテラル __FILE__ と File::Basename を使ってがんばる

というわけで FindBin を使った方法は便利ですが plackup を使うようなアプリではちょっと相性が悪いみたいですね。

では、plackup から起動する最近の PSGI なアプリはどうやって、サーチパスを追加しているのでしょうか。

tokuhirom さんが開発されている [Amon2](#) では、以下のようにサーチパスを追加していました。

- Amon2 \$dist.psgi より抜粋

```
use File::Spec;
use File::Basename;
use local::lib File::Spec->catdir(dirname(__FILE__), 'extlib');
use lib File::Spec->catdir(dirname(__FILE__), 'lib');
```

これは、最初にあげた PHP の例とやっていることは同じですね。

FindBin は実行中の script のディレクトリパスを返すのに対し、__FILE__ はこの特殊リテラルが書かれているファイルのパスを得ることができます。

この違いは、以下のようなサンプルスクリプトを書いてみるとわかります。

- ./sample.pl

```
use FindBin;
use lib "$FindBin::Bin/lib";
use Sample;
warn $FindBin::Bin;
warn __FILE__;
```

- ./lib/Sample.pm

```
package Sample;
use FindBin;
warn $FindBin::Bin;
warn __FILE__;
1;
```

- 実行結果

```
/home/hatyuki at lib/Sample.pm line 3.
lib/Sample.pm at lib/Sample.pm line 4.
/home/hatyuki at sample.pl line 4.
sample.pl at sample.pl line 5.
```

得られたパスが絶対パス (FindBin) か、相対パス (__FILE__) かの違いはありますが、とにかくこれで起点となるパスを得ることができました。

あとは、結果がディレクトリではなくファイルのパスとなっているため、ここからディレクトリのパスを得るため [File::Basename](#) を利用しています。

また、*nix と Windows でディレクトリセパレータが異なる ('/', '\') ため、それを吸収するために [File::Spec](#) を利用して、パスを生成しています。

File::Basename も File::Spec も標準モジュールであるという点もうれしいですね。

また、File::Basename や File::Spec は、ファイルやディレクトリのパスを扱うときにはいつでも使えるモジュールですので、使い方を覚えておきたいですね。

ワンライナー野郎御用達

ここからは、ちょっと脱線。

```
$ perl -Ilib -Mlocal::lib=extlib script.pl
$ plackup -Ilib -Mlocal::lib=extlib app.psgi
```

ワンライナーが大好きなみなさんは、きつとこうやって '-I' オプションを使って実行してるのではないかと。

local::lib を使っている場合には '-M' で local::lib を指定すればおっけーですね。

plackup でも同じように利用できるため、覚えておいても損はなさそうです。

中には常に '-I' オプションをつけて実行するため 'use lib' を使ったことがない強者がいるとかいないとか。

漢なら黙って @INC を直に

```
BEGIN { push @INC, '/path/to/my/lib' }
```

もうとにかく「ナマが好き」とかいう人がいるとかいないとか。

そんな人は @INC になまでそうにゆうしてるんじゃないですか。よくわかんないですけど。

BEGIN の外に出すと悲しい思いをするので、中に書いてください、なかに。よくわかんないですけど。

きつとこの分野に詳しい人とかいると思います。

ちなみに IRC クライアントの [tiarra](#) はこの方法でサーチパスを追加していました。

まとめ

というわけで、サーチパスを追加する方法について、いろいろご紹介してみました。

「たかだかサーチパスを追加するだけなのに、なんでこんなに方法があるんだ？」と思う方もいらっしゃると思いますが、まさに Perl のスローガン「やり方はひとつじゃない」という側面を垣間みることができるのではないのでしょうか。

このことは良いことでも、悪いことでもなく「Perl」という一つの言語の特徴だと思います。

多分ここで取り上げた方法以外にも、まだまだたくさんの方があるかと思っています。

と、いうわけで、みなさんもいろんな方法で Casual に Perl を楽しんでください！

さてさて、明日は 915626f6af93c5d5d7927594b4a4829a さんこと piarra さんの出番です！
おたのしみに！

*1: ある程度ディレクトリの構造が複雑でも、がんばって探してくれます

*2: Plack::Util::_load_sandbox あたりの処理の結果