



Perl6 最新情報

Tatsuhiko Miyagawa
<miyagawa@edge.co.jp>

もくじ

- History of Perl6
- Perl6 はどんな言語?
- Parrot
- perl 5.8

もくじ

- **History of Perl6**
- Perl6 はどんな言語?
- Parrot
- perl 5.8

Why Perl6: きっかけ

- 2000/06/18
 - perl5-porters meeting @ TPC4
 - “*We’re f***ked*” – *Jon Orwant*
- 2000/06/19
 - Keynote “State of the Onion”
 - Perl6 開発をアナウンス – Larry Wall

Why Perl6: motivation

- 技術的な理由
 - Perl5のコードは理解不能
 - オブジェクト指向などの見直し
 - XS の撤廃
- 社会的/政治的理由
 - 他言語へのアピール

Perl6 People

- Larry Wall: language designer
- Damian Conway: Evil Assistant
- Dan Sugalski: internal chief
- Simon Cozens: code chief (ex.)
- Jeff Goff: code chief
- Nathan Torkington: project manager

RFC: Request For Comment

- 2000/08/01 – 09/30
 - perl6-language@perl.org
 - Perl6 はどのような言語であるべきかのRFC
 - 変わるべきもの、変わるべきでないもの
 - 総数は 361個
 - <http://dev.perl.org/rfc/>

Camel Lot #6

- 2000/10-11
 - [Atlanta Linux Showcase](#)
 - Perl/Ruby Conference @ 京都
- Perl6 の進む道
 - Larry が理解できる
 - High-level / Low-level / Meta Language

Perl + Python = Parrot?

- 2001/04/01
 - [Programming Parrot](#)
 - Perl と Python を融合した新言語
 - Oreilly.com のエイプリルフールネタ
 - Perl6 のコードネームは Parrot に決定

Perl6 internals

- 実装は C
- PDD = Perl Design Documents
 - <http://dev.perl.org/perl6/pdd/>
- Parser – Compiler - Optimizer
- register based VM
- 言語に非依存(?)

Perl + Python = Parrot!

- CLR としての Parrot
 - 2001 OSCon (TPC5) BOF
 - language-dev メーリングリスト
 - Simon Cozens が中心
 - Python-dev
- Python, Ruby など動的インタプリタ言語の共通ランタイム

Apocalypse

- 2001/04/03 にスタート
- Larry Wall が RFC をもとに Perl6 を解説
 - RFC を PSA (= Problem, Solution, Acceptance) で分類
- ラクダ本の chapter に合わせた進行
- Perl.com で連載

Exegesis

- Damian Conway による Apocalypse (黙示録) への注釈書
- Apocalypse と同期 (=2 から開始)
- サンプルコードの解説が中心

ラクダ本にあわせると ...

- Apo1 2001/04/02
- Apo2 2001/05/03
- Apo3 2001/10/02
- Apo4 2002/01/18

ラクダ本は Chapter 33 までである

Timeline?

- Damian のスケジュール
(2001夏: YAPC時点)
 - 2001 年末 Design finish
 - 2002 May Alpha release
 - 2002 July Beta release
 - 2002 Oct. Perl 6.0.0

もくじ

- History of Perl6
- **Perl6 はどんな言語?**
- Parrot
- perl 5.8

アロー演算子

- `->` は `.` に
 - オブジェクト指向言語らしく？

```
$obj.method(@args);  
$value = $ref.{bar}; # . は省略可  
$obj = Class.bless(¥%data);
```

文字列連結演算子

- . は _ に
 - ~ という話もあったが ...

```
$str = $foo _ $bar;
```

```
$str = $foo_bar _ $baz;
```

```
$str = foo_ _ bar();
```

変数のプレフィクス

- `$@%` ルールが変更に
 - 配列の要素は `@`
 - ハッシュの要素は `%`

<code>@foo[2];</code>	<code># \$foo[2]</code>
<code>%foo{bar};</code>	<code># \$foo{bar}</code>
<code>\$bar.[2];</code>	<code># \$bar->[2]</code>
<code>\$bar.{baz};</code>	<code># \$bar->{baz}</code>

特殊変数/三項演算子

- @ARGV は @ARGS に
- エラー変数はまとめて \$! に
- ? : は ?? :: に

```
$foo = $bool ?? 1 :: 2;
```

プロパティ is, but

- Perl5 の attribute の拡張
- compile-time / run-time

```
my $pi is constant = 3.14;  
method set is rw { ... }  
$fh is chomped;  
return 1 but false; # run-time
```

ビルトインの型

- コンパイラ/最適化ザへのヒント
- BOOL, INT, NUM, STR, REGEX

```
my INT $i;# プロパティが使える
```

```
my int $i;# プロパティ使えない
```

名前付きパラメータ

- サブルーチンの引数
- 変更する場合は write プロパティ

```
sub add ($bar, $baz) {  
    return $bar + $baz;  
}  
  
sub inc ($i is rw) {  
    $i++;  
}
```

Switch

- switch は英語っぽくない
- given ... when ...

```
given ($foo) {  
  when 1: { }  
  when Classname: { }  
  when $re: { }  
}
```


when

- マッチするルールは複雑
- このルールを “smart-match” と名付ける
- =~ はデフォルトで smart-match

```
# smart-matching
if ($foo =~ $bar) { ... }
```

// 演算子

- 左辺が undef のとき右辺を評価
- Perl5 でありがちなバグを回避
- || -> or, // -> err

```
$num = shift || 10; # perl5  
$num = shift // 10; # perl6  
$bar = func() or die; # perl5  
$bar = func() err die; # perl6
```

例外処理

- Exception はオブジェクト
- CATCH ブロック

```
try { ... }  
CATCH { # smart-match for $!  
  when Exception::Foo : { }  
  when /division by zero/ : { }  
}
```

トピカライザー

- \$_ は “topic”
- given, for (-> topic allow)

```
for @obj -> $obj {  
    print;      # print $obj  
    .method(); # $obj.method()  
}
```

Currying

- クローージャの引数を簡単に指定
- functional programming

```
$add = { $^x + $^y };  
print $add(1,2); # 3  
  
$plus_seven = $add(7);  
print $plus_seven(11); # 18
```

hyper-operator

- スカラ用の演算を配列に適用
- map でやると汚くなる処理を簡単に

```
@foo = @bar ^- @baz;
```

```
@files ^=~ s/¥.jpg$//;
```

Misc.

- /x がデフォルト
- local は temp に
- pseudo-hash must die!
- 変数展開は \$()
- スレッドは ithreads ベース
- 標準モジュールなし CPAN.pm のみ

Perl6への反応

- "Perl6 is just for Damians"
 - Damian 「変えたくないものは変えなくてもOK。
それが DWIM (Do What I Mean)」
- Perl6 in Perl5!
 - Perl6::*
 - Perl6 like modules

もくじ

- History of Perl6
- Perl6 はどんな言語?
- **Parrot**
- Perl 5.8

Parrot

- Perl6 (+ Python, etc.)用 VM
- 最新版は 0.0.5
- <http://www.parrotcode.org/>
- perl6-internals@perl.org

Register Machine

- 4つのタイプに32個のレジスタ
 - Integer, Number, String, PMC
 - PMC = Parrot Magic Cookie
 - I0..I31, N0..N31, S0..S31, P0..P31

Parrot Assembler

■ Parrot アセンブラ言語

```
    set I0, 42
    bsr DOUBLE
    print I0
    print " was returned¥n"
    end
DOUBLE: mul I0, I0, 2
    ret
```

Parrot Assembler

- 関数 (add, sub, mul ...)
- 分岐 (eq, ne, lt, le, gt ...)
- PMC

PMC

- Parrot Magic Cookie
- 抽象データ型
 - Array
 - PerlUndef
 - PerlInt
 - PerlNum
 - PerlString, etc.

PMC (2)

■ new と set

```
new P0, PerlInt
set P0, 123
new P1, PerlInt
set P1, 321
add P1, P1, P0
print P1
print "¥n"
end
# prints out 444
```

バインディング

- Jako
- Cola
- BASIC
- mini-scheme
- Java bytecode to Parrot
- B::Parrot
- mod_parrot

もくじ

- History of Perl6
- Perl6 はどんな言語?
- Parrot
- **perl 5.8**

perl5.8

- 5.6 のバグ修正と機能拡張
- 最新 stable
- 5.8.0 RC1 が 2002/06/01 にリリース
- 2002年夏にも正式版リリース

5.8 の新機能

- Unicode
 - UTF8 完全対応
 - Encode.pm
- PerlIO
- ithreads
- Test, Test, Test!

Encode.pm

- 100種類以上のエンコーディングに対応
- encoding.pm で jperl

```
#!/usr/local/bin/perl5.7.3  
  
use encoding "euc-jp";  
  
my $name = "みやがわたつひこ";  
  
print length $name;
```

Encode.pm (2)

- Encode.pm を利用したアプリケーション
 - [Encode::InCharset](#)
 - [piconv online](#)
 - [Apache::GuessCharset](#)
 - [Encode::Punycode](#)

PerlIO

- 文字コード変換 圧縮/伸長など
- IO::Scalar はもういらない

```
open(FH, "<:utf8", "file");  
open $fh, ">", ¥$variable;
```

ithreads

- Interpreter Threads
- インタプリタベースのマルチスレッド
- multiplicity
- threads::shared

SEE ALSO

- <http://dev.perl.org/>
- <http://www.parrotcode.org/>
- <http://use.perl.org/>
- <http://www.perl.com/>
- perl5-porters
- perl6-language