パール5.6.0の新機能

- 概要
  - インストールとOSサポート
  - スレッドプログラミング
  - ユニコードサポート
  - SUBの新しい文法
  - I／O、大きいファイルサポート
  - オブジェクトプログラミング
  - モジュールのチェンジ

インストールとOSサポート

- 新しいインストールフラグ
  - usethreads useithreads
    - 複数interpreter使える（perldoc perlembed)
  - usethreads use5005threads
    - パールからスレッド使える
  - usemultiplicity
    - CからperlclonenのAPIが使える
  - use64bitint, uselongdouble, uselongall
    - 64ビットのintとロングdoubleが使える（64ビットハード）
  - uselargefiles(2GB以上のファイルが使える)
  - usesocks（NECのsockライブラリが使える）

インストールとOSサポート

- 新しいOS
  - VM/ESA
  - BS2000
  - GNU/Hurd
  - EPOC on Psion (PDAコンピュータ)
  - cygwin
- Win32のチェンジ
  - fork()とサーバプログラミング
  - XSUBのドキュメント
  - kill()はプロセスのIDのサポートができます
  - Shellモジュールサポート

スレッド例

- パールからスレッド
  - コンパイルとき -Dusethreads -Duse5005threads

```
use Thread;
my $thr = new Thread ¥&mysub, $param1, $param2; # スレッド作成
$thr->detach(); # return しない場合
$retval = $thr->join(); # return する場合
sub mysub {  foreach (@_) { do something... } }
```

  - スレッドについて
    - 軽い
    - スレッドのコミュニケーションが簡単
    - select()なくて1プロセスのサーバ
    - スレッドの問題: SMP race conditions, ロック、不安定

Unicode

- 16ビットキャラクターで100万以上の文字表示
- パース、ソート等処理は簡単
- キャラクタープルパーティでどんな言語でもregular expressionが使える
- キャラクタname、コードでキャラクタを表示
- ASCII compatible コード（UTF-8)

Perlとユニコード

- PerlはUTF8を使います
  - Unicode editorでパールプログラムが作れる
  - Perl変数名はUnicodeキャラクタで
  - Regular expression, 文字列opsはUnicode文字列に対応
  - キャラクタクラスを使ってテキストの処理ができます
  - 昔のプログラムもちろんそのまま使える
- 新しいops:
  - ¥x{hex}, ¥N{NAME} => キャラクター表示
  - In regexp:
    - ¥c, ¥C => バイトをmatch
    - ¥N{name}, ¥N{script:name} =>キャラクターmatch

Perlとユニコード

- 新しいops:
  - In regexp:
    - ¥p{}, ¥P{} =>ユニコードプロパティmatch
    - ユニコードのプロパーティについて
      - lib/5.6.0/unicode の下のモジュールを参加してください
  - use byte; でバイトで処理します
  - use utf8; でソースコードにutf8 を使える
  - use charnames ‘:full’;でキャラクタフールnameを使える

    use charnames ‘:full’; print “¥N{KATAKANA LETTER A}”; # ア
  - use charnames ‘:short’; でユニコードscript:charを使える

    use charnames ‘:short’; print “¥N{katakana:a}”;

Perlとユニコード

- オペレータとユニコード
  - 文字列のオペレータ（chop,substr, pos, index, rindex, sprintf, write, length）はunicodeに自動にswitchする
  - vec, pack, unpack は自動にswitchしない
  - sort(), chomp(), ファイルopsはswitch行為がない
  - pack とunpack は新しいUテンプレートルールがあります
  - とりあえず、I/Oとlocaleの設定はユニコードに対応がない

SUBの新しい文法

- SUBはmodifierのサポートがあります
  - いまmodifierの三つがあります

    sub foo : method; # fooはクラスmethodにします。lockedと同時を使う場合はオブジェクト($_[0])をロックされています。

    sub foo : locked; # fooはスレッドに使うときsubは

    ロックされています

    sub foo : lvalue; # fooにassignができます
- 将来はユーザモジュールのなかカスタムmodifierの設定がでいます。

  たとえば？

  sub foo : heavy;

  if ($processor > “986”) { &foo } else { die “computer too slow” }

I／O、大きいファイルサポート

- 5.6.0からopen(my $var, “file”)が使える。このscopeから出るとき、ファイルハンドル自動にcloseになります
- open(FH, mode, name)のサポート

  例： open (FH, “>”, “newfile”);

  open(FH, “+<“, “dbfile”);

- openモジュールでモードの設定ができます。

  use open FH => modeでモードの設定ができます。5.6.0のモードはcrlfとrawモードのサポートがあります。次のバージョンはこのモジュールでUnicode等のモードの設定をします。

オブジェクトプログラミング

- 新しいphashというものがあります。
  - Phashは？Arrayのスピードとサイズでnamed attributesが使えます。

    例:

    use fields; my $struct = fields::phash(f1=>1, f2=>”two”);

    $struct->{new}; # error

    print $struct; # prints ARRAY(0x815a3e4)

    例:

    package foo; use fields ‘f1’, ‘f2’;

    sub new { my $class = ref $_[0] || $_[0]; fields::new($class) }

オブジェクトプログラミング

```
package main;
$obj = foo->new();
print $obj; # prints foo=ARRAY(0x….)
$obj->{f1} = 12; # OK
```

- our変数declaration(strict ‘vars’の下で使います)
  - our $varでグローバル変数をdeclareする。
  - ourの使い方はuse vars ‘$var’と同です。
  - Scopingルールはmyと同じです。

面白いモジュール

- attributes モジュール
  - subのmodifierを設定します
  - subのmodifier情報取れます
- B,Byteloaderモジュール
  - パールコンパイラ
    - perlcc -b test.pl (bytecodeを作ります）
    - perl test.plc (コンパイルされたファイルを実行)
  - perlcompileドキュメント

新しいドキュメントページ

- perlunicode（ユニコード）
- perltootc (パールでOOP)
- perlfork (Win32でfork())
- perllexwarn(warningsについて)
- perlthrtut (スレッドプログラミング)