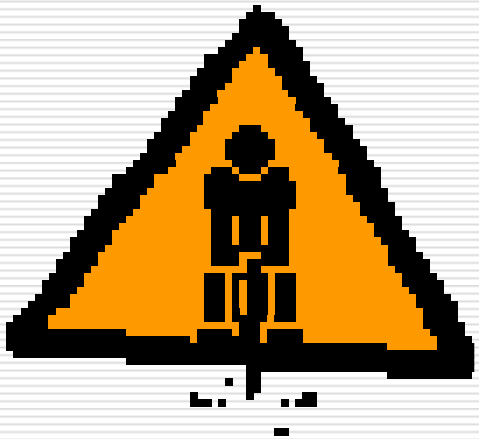
PIR Samples (20')

François Perrad
francois.perrad@gadz.org
PAUSE ID : PERRAD
Parrot SVN : fperrad

Introduction

- Some PIR samples
- Good patterns & idioms
- Not the boring part
- Only the fun part



CAUTION

CODE

LWP/UserAgent get

```
$ cat get.pir
.sub 'main' :main
    .param pmc args
    $S0 = shift args      # progname
    .local string url
    url = shift args
    load_bytecode 'LWP/UserAgent.pir' # or .pbc
    .local pmc ua, response
    ua = new ['LWP';'UserAgent']
    ua.'show_progress'(1)
    response = ua.'get'(url, 'close' :named('Connection'))
    $S0 = response.'content'()
    say $S0
.end

$ parrot get.pir http://www.parrot.org/ > home.html
** GET http://www.parrot.org/ ==> 200 OK (1s)
```

Object Oriented

```
.namespace ['MyApp';'MyClass']

.sub '' :init :load :anon
    $P0 = newclass ['MyApp';'MyClass']
    $P0.'add_attribute'('member')
.end

.sub 'init' :vtable :method
    $P0 = box 3.14
    setattribute self, 'member', $P0
.end

.sub 'member' :method  # getter
    $P0 = getattribute self, 'member'
    .return ($P0)
.end
```

Object Oriented

```
.sub 'member' :method    # setter
    .param pmc value
    setattribute self, 'member', value
.end

.sub 'member' :method    # getter/setter
    .param pmc value :optional
    .param int has_value :opt_flags
    unless has_value goto L1
    setattribute self, 'member', value
    .return ()
  L1:
    $P0 = getattribute self, 'member'
    .return ($P0)
.end
```

Iterators

```
    $P0 = split ',', “abc,def,ghi”    # array
    $P1 = iter $P0
  L1:
    unless $P1 goto L2
    $S0 = shift $P1
    ...
    goto L1
  L2:

    $P0 = iter MyHash
  L3:
    unless $P0 goto L4
    .local pmc key, val
    key = shift $P0
    val = MyHash[key]
    ...
    goto L3
  L4:
```

Literal data

```
$P0 = new 'Hash'
$P0['key1'] = 'some text'
$P0['key2'] = 3.14
$P0['key3'] = $P1

$P2 = new 'FixedIntegerArray'
set $P2, 2
$P2[0] = 3.14
$P2[1] = 2.78

$P3 = new 'ResizableIntegerArray'
push $P3, 3.14
```

Calling convention

```
    $P0 = hash( 'some text' :named('key1'), \
                3.14 :named('key2'), \
                $P1 :named('key3') )

    $P2 = array( 3.14, 2.78 )

# helpers
.sub 'hash'
    .param pmc kv :slurpy :named
    .return (kv)
.end

.sub 'array'
    .param pmc args :slurpy
    .return (args)
.end
```

Calling convention

```
.sub 'runtests'
    .param pmc files :slurpy
    .param pmc opts :slurpy :named
    ...

.sub 'spectest'
    .param pmc kv :slurpy :named
    run_step('build', kv :flat :named)
    runtests('t/test.rb', 'ruby' :named('exec'))
.end

.sub 'sanity'
    .param pmc kv :slurpy :named
    $P0 = glob('t/0*.t')
    $S0 = 'parrot lua.pbc'
    runtests($P0 :flat, $S0 :named('exec'))
.end
```

I/O

```
.sub 'slurp'
    .param string filename
    $P0 = new 'FileHandle'
    $S0 = $P0.'readall'(filename)
    .return ($S0)
.end

.sub 'spew'
    .param string filename
    .param string content
    $P0 = new 'FileHandle'
    $P0.'open'(filename, 'w')
    $P0.'puts'(content)
    $P0.'close'()
    .return ()
.end
```

I/O

```
.include 'stat.pasm'
.sub 'unlink'
    .param string filename
    .param int verbose          :named('verbose') :optional
    .param int has_verbose      :opt_flag
    $I0 = stat filename, .STAT_EXISTS
    unless $I0 goto L1
    $I0 = stat filename, .STAT_ISREG
    unless $I0 goto L1
    unless has_verbose goto L2
    unless verbose goto L2
    print "unlink "
    say filename
  L2:
    new $P0, 'OS'
    $P0.'rm'(filename)
  L1:
.end
```

I/O

```
.sub 'gzip'
    .param string filename
    .local pmc fh, gh
    fh = new 'FileHandle'
    $S0 = fh.'readall'(filename)
    $P0 = loadlib 'giphandle'
    gh = new 'GzipHandle'
    $S1 = concat filename, '.gz'
    gh.'open'($S1, 'wb')
    gh.'puts'($S0)
    gh.'close'()
    unlink(filename)
.end
```

Join vs Concat

```
.sub 'message'
    .param pmc args :slurpy
    $S0 = join '', args
    $P0 = getstderr
    print $P0, $S0
.end

    $S1 = $I1
    message('bad value ', $S1, ' in this case')
    …
    $S0 = 'bad value '
    $S1 = $I1
    $S0 .= $S1
    $S0 .= ' in this case'
    message($S0)
```

References

- Le Langage PIR (part 1-4)

  by Christian Aperghis-Tramoni, in Linux Magazine

- Parrot Developer's Guide: PIR

  by Allison Randal & Andrew Whitworth

- http://trac.parrot.org/parrot/browser/trunk/docs/book/pir