Deploying And Supporting Perl 6

Jonathan Worthington

UKUUG Spring 2007 Conference

Deploying And Supporting Perl 6

Jonathan Worthington

UKUUG Spring 2007 Conference

Deploying And Supporting Perl 6

Overview

- Yesterday: the Perl 6 language
- Today:
  - What is Perl 6?
  - Perl 6 implementations
  - Perl 5 to Perl 6 migration
  - Modules in Perl 6
  - CPAN6

What is Perl 6?

- Perl 5 = implementation + test suite
  - There was no official language specification
  - Only the one implementation
- Perl 6 = specification + test suite
  - Language specification (informal)
  - Test suite
  - Many implementations possible

Implementations

Pugs

- Started out as an implementation of Perl 6 in Haskell
- Provides fast feedback to the language designers
- Provides a Perl 6 implementation for people to play with
- Being used to develop and run the Perl 6 test suite

Pugs

- Pugs has spawned many interesting side-projects
  - Compiling Perl 6 down to JavaScript => run in the browser
  - 6 on 5 => Run Perl 6 on Perl 5
  - Various Perl 5 modules that provide Perl 6 semantics
  - v6 – Perl 6 in Perl 5

Parrot

- A project to implement a virtual machine for dynamic languages
  - Similar to the JVM and the .NET CLR
  - However, these VMs focussed on static languages
- Started at the same time as the Perl 6 specification
- Separation of runtime and language

Deploying And Supporting Perl 6

Parrot

- Designed to run more than just Perl 6
- Today implementations are underway for...
  - Tcl
  - PHP
  - Ruby
  - Python
  - Many more…

Parrot

- Aims to run Perl 6 programs fast!
  - Compile Perl 6 to bytecode => don't have to parse the source every time
  - JIT compiler => potential for high performance code (close to C)
- So hopefully, less Perl extensions need to be written in C to get performance
- Written in Perl = no C to compile ☺

Parrot

- Lower memory footprint
  - Bytecode files are mmap'd on platforms that support it
  - Just one copy of a bytecode file in memory, shared by Parrot instances
- So even if the compiler is implemented in Perl 6 and compiled to Parrot bytecode, it's still shared

Parrot

- Native Calling Interface
  - You can write Parrot programs that call into C libraries
  - Pure Parrot bytecode – no C compiler needed
- Further decreases the number of C extensions needed in Parrot and thus Perl 6

Parrot

- Parrot is written in C
  - A C compiler is available on pretty much any platform => portability
- Take advantage of platform specific performance advantages when available, but with a fallback
  - JIT is highly CPU specific => fallback to interpreter – still reasonably fast

The Standard Grammar

- Grammar = formal definition of the syntax of a language
- The Perl 6 standard grammar is nearly complete
  - Being defined in the Perl 6 grammar language itself!
- Pugs and Parrot implementations will both use it to parse Perl 6 soon

What I Expect You'll Be Deploying

- No official implementation, just an official specification, test suite and grammar
- However, the Parrot implementation is what you will most likely be deploying
  - Performance
  - Portability

Migration

The Problems

- Perl 6 is not source-code backward compatible to Perl 5
  - A program that is valid Perl 5 usually won't be valid Perl 6
- Massive deployed base of Perl 5 code that needs to keep running
  - Including CPAN
- Need to gradually introduce Perl 6

Using Perl 5 Modules In Perl 6

- You can use Perl 5 modules in Perl 6

```
use perl5:DBI;
use perl5:My::Fave::Module;
```

- Means that the current CPAN remains usable in Perl 6
- You can start introducing Perl 6 into a Perl 5 environment for new things, without having to re-write everything

Using Perl 5 Modules In Perl 6

- This is implemented in Pugs today
- Note that it requires embedding a Perl 5 interpreter
- Bridge between them maps Perl 5 objects into Perl 6 space and vice versa.

The Perl 5 to Perl 6 Source Translator

- The current Perl 5 parser is the only thing that can really parse Perl 5
- Modified to optionally keep hold of all the things it used to throw away – comments, POD, etc.
- Generates an XML representation of a Perl 5 program – enough to reproduce the original program with comments, etc.

The Perl 5 to Perl 6 Source Translator

- For testing purposes, a printer was implemented to turn this XML back into Perl 5
- Now we can translate Perl 5 to Perl 5 ☺
- This is being modified to generate Perl 6 instead
  - Was worked on as a Google Summer of Code project

Recognizing Perl 5

- The Perl executable needs some way of knowing if it's being fed Perl 5 or Perl 6
- Every Perl 5 module starts with a package directive

```
package My::Business::Logic;
```

- This isn't valid Perl 6 syntax => module identified as Perl 5; similarly, **`module`** and **`class`** are not valid in Perl 5

Module Naming

Long Module Names

- Every Perl 6 module and class that is placed on CPAN or into some other archive will be required to declare its long name
- Includes the name itself and…
  - A version number
  - A URI identifying the publishing author or authority

Declaring Long Module Names

- Examples

```
# Full syntax: specify as adverbs
class My::Thing:ver<2.5.2>:auth<CPAN:FRED>;
class Cat:ver<1.0.2>:auth<mailto:c@tz.com>;

# The shorter but equivalent syntax
class My::Thing:<2.5.2 CPAN:FRED>;
class Cat:<1.0.2 mailto:c@tz.com>;
```

- Within the class itself, the short name is declared as an alias to the long name

Using Modules

- If you do not care what version or author, a straightforward use works

```
use My::Thing;
use My::Thing:ver(Any):auth(Any); # Same
```

- Alternatively, can require a particular version, or at least a certain version, or that its from a particular author(ity)

```
use My::Thing:ver<2.5.2>;  # Only 2.5.2
use My::Thing:ver(1.5..*); # 1.5 or later
use My::Thing:auth<CPAN:FRED>; # by FRED
```

Advantages

- The long names of the modules are what they are stored under
- Multiple versions of modules from different authorities can co-exist on a single Perl installation
- Ideally, modules would not change their interface in later versions – but they do!
- Now there's a better way to deal with it

CPAN6

Why CPAN6?

- CPAN has served us very well for ten years and will continue to do so for years to come
- Perl 6 and Parrot bring new needs
  - May have modules in many languages on CPAN, not just Perl 6
  - Need to keep older versions around, plus versions by different authors

Why CPAN6?

- Enterprises have their needs too
  - Want to know releases of key modules are trusted, signed off and so on
  - May want to run their own internal archive of modules that integrates well with the module installation tools

Why CPAN6?

- Want to improve a few other things
  - Allow multiple authors per module that can make releases, not just one as is possible now
  - Be able to see the consequences of a module installation better before doing it (what dependencies will it install, how much disk space will it take, and so on)

Releases And Archives

- A release is some piece of software or information that is to be distributed; it's made up of multiple related files
- An archive contains a set of releases
- Archives may have their own "constitution", governing who can make releases, namespace rules and so on
- Can create your own archives

Three Parts

- CPAN6 = set of concepts and ideas for distribution of archives
- Pause6 = management of archives, allowing people to add releases, handling trust issues and so on
- CPAN6.pm = a search and installation tool
- Pause6 and CPAN6.pm replaceable

Trust

- Today: username/password authentication, then you can make a release
- In the future: releases can be signed, perhaps by multiple people
- Archive constitution may require a certain number of signatures on a release before it is trusted

Status

- A great deal of the design work is done
- Implementation of both Pause6 and CPAN6.pm are underway, but there's nothing to play with yet
- A lot depends on the community accepting it
- Get the latest news
  http://www.cpan6.org/

Summary

Deploying And Supporting Perl 6

Don't panic!

- Perl 6 is coming, both in terms of specification and implementation
- Migration issues are being considered and taken seriously
  - Already can use Perl 5 modules from Perl 6, source translator underway
- Module management and installation should be getting less painful

Thank you!

Questions?