PL/Parrot
Embedding the Parrot Virtual Machine in PostgreSQL

Jonathan "Duke" Leto and David Fetter

Parrot Virtual Machine

- Process (Application) Virtual Machine
- Register-based
- Continuation Passing Style
- Design Goals
    - Pluggable
    - Interoperable
    - Dynamic

Why Embed Parrot VM in PostgreSQL?

- PL's are (very) hard to write and maintain
- Framework for DSL's
- Platform independent, fast, stored procedures
- Allow various PL's to communicate
- Freeze/thaw subtransaction-level states

History of PL/Parrot

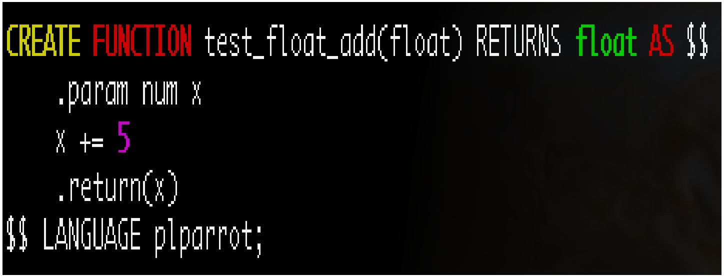
Current Features

- PL/PIR(U)
- Pass and return basic datatypes
- Basic security model (Don't do that)
- Growing Test Suite
- Enthusiastic and friendly community

Bugs

- Documentation
- SPI
- Triggers
- Parrot Bugs
  - IMCC Syntax Errors
  - Loading libraries from Embed API
  - Security API

Example Code



```
CREATE FUNCTION test_float_add(float) RETURNS float AS $$
    .param num x
    x += 5
    .return(x)
$$ LANGUAGE plparrot;
```

Future Goals

- PL/Rakudo - Perl 6 in your database!
- PL/Pynie - Python in your Parrot in your database!
- Tools to help create a new DSL with PL/Parrot

Get involved!

- Try PL/Parrot on your system and submit detailed bug reports
- Fork on github and hack on stuff!
- Help with GitHub Issues

Thanks

- PL/Parrot team: Joshua Tolley, David E. Wheeler, Daniel Arbelo Arrocha + others
- Everyone working on Parrot VM and PostgreSQL

Resources

- http://github.com/leto/plparrot
- #plparrot on freenode
- http://parrot.org
- @parrotvm / !parrot on twitter/identi.ca