

# A Toolchain for the Parrot Ecosystem (40')

---

François Perrad

[francois.perrad@gadz.org](mailto:francois.perrad@gadz.org)

PAUSE ID : PERRAD

Parrot SVN : fperrad

fperrad@FPW'10



# What are the goals ?

---

- ❑ A Toolchain for the Parrot Ecosystem
- ❑ Not for Parrot itself
  
- ❑ Build/Test/Install/Package languages or libraries/modules based on Parrot
- ❑ Without any dependency except an installed Parrot
- ❑ Easy to use and widely used
- ❑ Shipped with Parrot

# The Parrot way

---

```
$ perl Configure.pl
```

```
# generates a Makefile from a  
template
```

```
$ make
```

## □ Main Issues

- Various make : gmake, nmake, ..
- OS Shell dependencies or Perl5  
ExtUtils::Command

# Configuration

---

□ Parrot embeds its own configuration data

```
$ parrot_config --dump
```

□ From PIR code

```
$P0 = getinterp
```

```
cfg = $P0[ '.GLOBALS_CONFIG_HASH' ]
```

```
$S0 = cfg[ 'cflags' ]
```

# The DistUtils way

---

- With Python, the classic mantra is :

```
$ python setup.py
```

```
$ python setup.py test
```

```
$ python setup.py install
```

- `setup.py` contains :

```
import distutils
```

```
setup(
```

```
    a lot of named parameters
```

```
)
```

- see Python Distribution Utilities

<http://docs.python.org/distutils/>

# With Parrot

---

- ❑ All the rules are coded in the `distutils` library
- ❑ A module author just must write a script `setup.pir` or `setup.nqp`

```
pir::load_bytecode('distutils.pir');  
setup( @steps,  
       ... many key/values here ...  
);
```

```
$ parrot setup.pir clean update build test  
$ parrot-nqp setup.nqp
```

# Steps / Targets

---

- help
- build / clean
- test / smoke
- install / uninstall
- update / patch
- sdist, sdist\_gztar, sdist\_zip, sdist\_rpm, manifest
- bdist, bdist\_rpm, bdist\_wininst, spec, control, ebuild
- plumage

# Build items

---

- dynpmc
- dynops
- pir\_nqp
- pir\_pir
- inc\_pir
- pbc\_pir
- exe\_pbc
- installable\_pbc



# Example

---

```
.sub 'main' :main
  .param pmc args
  $S0 = shift args
  load_bytecode 'distutils.pbc'

  $P0 = new 'Hash'
  $P1 = new 'Hash'
  $P1['hello.pbc'] = 'hello.pir'
  $P0['pbc_pir'] = $P1
  $P2 = new 'Hash'
  $P2['parrot-hello'] = 'hello.pbc'
  $P0['installable_pbc'] = $P2
  .tailcall setup(args :flat, $P0 :flat :named)
.end
```

# Test / Smoke items

---

- prove\_exec / test\_exec
  - default value = parrot
- prove\_files / test\_files
  - default value = t/\*.t
- prove\_archive / smolder\_archive
- smolder\_url
- smolder\_tags, smolder\_comments,  
smolder\_extra\_properties

# Packaging & metadata

---

- doc\_files
- manifest\_includes, manifest\_excludes
- name
- version
  - `$ parrot setup.pir --version 0.5 sdist`
- abstract
- description
- authority
- license\_type
- license\_uri
- copyright\_holder

# Extensibility

---

- A API for custom step
  - register\_step
  - register\_step\_before
  - register\_step\_after
  - run\_step
  
- The `osutils` library

# Osutils library

---

- Functions :

system, mkdir, mkpath, install, cp, chmod, unlink, rmtree, basename, dirname, cwd, chdir, chomp, glob, getenv, setenv, slurp, spew, append, tempdir, tmpdir, catfile, splitpath

- from : perlfunc, File::Basename, File::Spec, ...

# Other libraries

---

- ❑ TAP/Harness, TAP/Parser, TAP/Formatter (todo: AnsiColor & ParallelSession)
- ❑ Archive/Tar (only creation)
- ❑ Archive/Zip (only creation)
- ❑ LWP/Protocol (only file & http)
- ❑ LWP/UserAgent (with proxy ?)
- ❑ URI
- ❑ GzipHandle PMC (zlib wrapper)

# Side effect on Parrot

---

```
$ parrot t/harness.pir --archive --send-to-smolder
```

- ❑ run & parse ~12000 tests
- ❑ creat an archive \*.tar.gz
- ❑ POST it to [Smolder](#) server

**Eating your own dog food**

# Related work on Parrot

---

## □ ops\_pct

- Rewrite the Ops preprocessor (ops -> C) with the PCT
- Branch merged on May 2010

## □ pmc\_pct

- Rewrite the PMC preprocessor (PMC ->C) with the PCT
- Branch without recent work



# Concurrency, the next big step

---

```
$ parrot setup.pir --jobs nb_cores
```

- For build & test (with TAP library)
  - Design ready
  - Waiting for Parrot

# The Plumage Project

---

- ❑ fetch/update/configure/build/test/install a project
- ❑ Each project is described by a JSON file
- ❑ Plumage recognizes `setup.pir` as build method
- ❑ Distutils could generate Plumage file from metadata
- ❑ See <http://trac.parrot.org/parrot/wiki/ModuleEcosystem>
- ❑ Currently broken (after 2.3.0 release)
- ❑ Not server for JSON files