PAR

The Perl Archive Toolkit

Using PAR to package and deploy Perl applications

http://par.perl.org

What is PAR ?

- Is to perl what JAR (Java Archive) is to Java
- Aggregates modules, scripts and other files into a Zip file
- Makes it easy to ensure a consistent versions
- Provides a rich toolkit to solve many distribution problems

Simple Use - Creating PAR archives

- zip foo.par Hello.pm World.pm
- zip -r bar.par lib/
- Can include and run scripts in an archive too

Simple Use - Using PAR Archives

```
# To use Hello.pm from ./foo.par
perl -MPAR=./foo -MHello

use PAR ‘foo.par’;
use Module::From::foo::par;

use PAR ‘/home/foo/*.par’;

use { file => ‘foo.par’, fallback =>  1};
Foo::Bar will be searched in the system libs first and loaded
from foo.par if it wasn't found
```

PAR::Packer

- pp - the Perl Packager
  Builds executables from perl scripts
  Enables obfuscation using input Filters
- par.pl - Make and run PAR archives
  par.pl foo.par test.pl # runs script/test.pl in foo.par
- parl - run PAR archives using built in perl
- tkpp - Graphical Perl Packager

pp - The Perl Packager

- Builds standalone executables from Perl programs using PAR and Module::ScanDeps
- Similar to perlcc but it works, generates smaller executables
- Can use Filters to enable code obfuscation

```
pp hello.pl                # Pack 'hello.pl' into executable 'a.out'
pp -o hello hello.pl       # Pack 'hello.pl' into executable 'hello'
pp -o foo foo.pl bar.pl    # Pack 'foo.pl' and 'bar.pl' into 'foo'
./foo                      # Run 'foo.pl' inside 'foo'
mv foo bar; ./bar          # Run 'bar.pl' inside 'foo'
mv bar baz; ./baz          # Error: Can't open perl script "baz"

pp -p file                 # Creates a PAR file, 'a.par'
pp -o hello a.par          # Pack 'a.par' to executable 'hello'
pp -S -o hello file        # Combine the two steps above

pp -p -o out.par file      # Creates 'out.par' from 'file'
pp -B -p -o out.par file   # same as above, but bundles core modules
                                # and removes any local paths from @INC
pp -P -o out.pl file       # Creates 'out.pl' from 'file'
pp -B -p -o out.pl file    # same as above, but bundles core modules
                           # and removes any local paths from @INC
                           # (-B is assumed when making executables)

pp -e "print 123"          # Pack a one-liner into 'a.out'
pp -p -e "print 123"       # Creates a PAR file 'a.par'
pp -P -e "print 123"       # Creates a perl script 'a.pl'

pp -c hello                # Check dependencies from "perl -c hello"
pp -x hello                # Check dependencies from "perl hello"
pp -n -x hello             # same as above, but skips static scanning

pp -I /foo hello           # Extra include paths
pp -M Foo::Bar hello       # Extra modules in the include path
pp -M abbrev.pl hello      # Extra libraries in the include path
pp -X Foo::Bar hello       # Exclude modules
pp -a data.txt hello       # Additional data files

pp -r hello                # Pack 'hello' into 'a.out', runs 'a.out'
pp -r hello a b c          # Pack 'hello' into 'a.out', runs 'a.out'
                           # with arguments 'a b c'

pp hello --log=c           # Pack 'hello' into 'a.out', logs
                           # messages into 'c'
pp @file hello.pl          # Pack 'hello.pl' but read _additional_
                                # options from file 'file'
```

PAR::Dist::FromCPAN

Create PAR distributions from CPAN

Build .par files from CPAN Modules

Provides script cpan2par

equivalent to cpan install <ModulePattern> but builds a .par archive

e.g.

cpan2par Template

cpan2par --follow Moose