CPANPLUS
Beyond The Shell

Who the hell is this guy?

- By day, a Network Services Specialist
- By night, perl hacker and CPAN Tester
- CPAN ID and IRC nickname, 'BinGOs'
- email: chris@bingosnet.co.uk
- Using Perl since 1995
- CPAN Tester since 2005
- CPANPLUS maintainer since Oct 2009

MANIFEST

MANIFEST

- CPANPLUS?
- Configuration
- Module objects
- Backend and API
- Distribution classes
- Source Engines
- Shells and Plugins

CPANPLUS ?

History

- Genesis at YAPC::Europe 2001
- Code started October 2001
- First release March 2002
- Complete rewrite 2004
- First release in core with v5.9.5 July 2007

Competitors

CPAN.pm

CPAN.pm

- The original
- Has an API
- Not extendable
- Code not for mere mortals

cpanminus

cpanminus

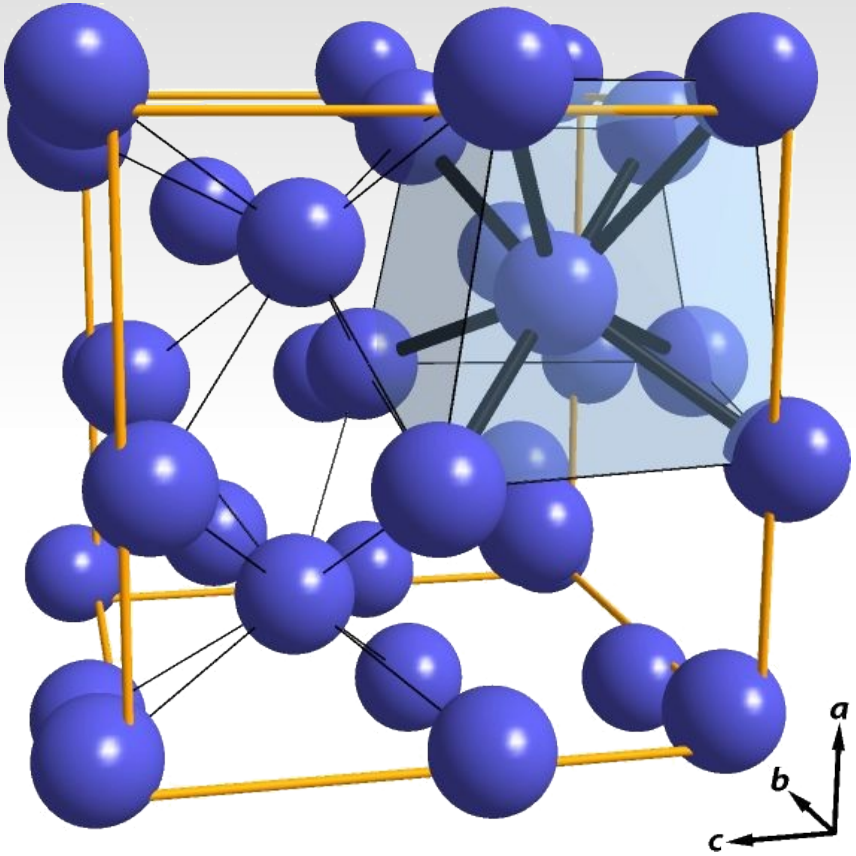
- The newest
- Easy to use – zero config
- No API
- Great for users
- Doesn't send CPAN Tester reports

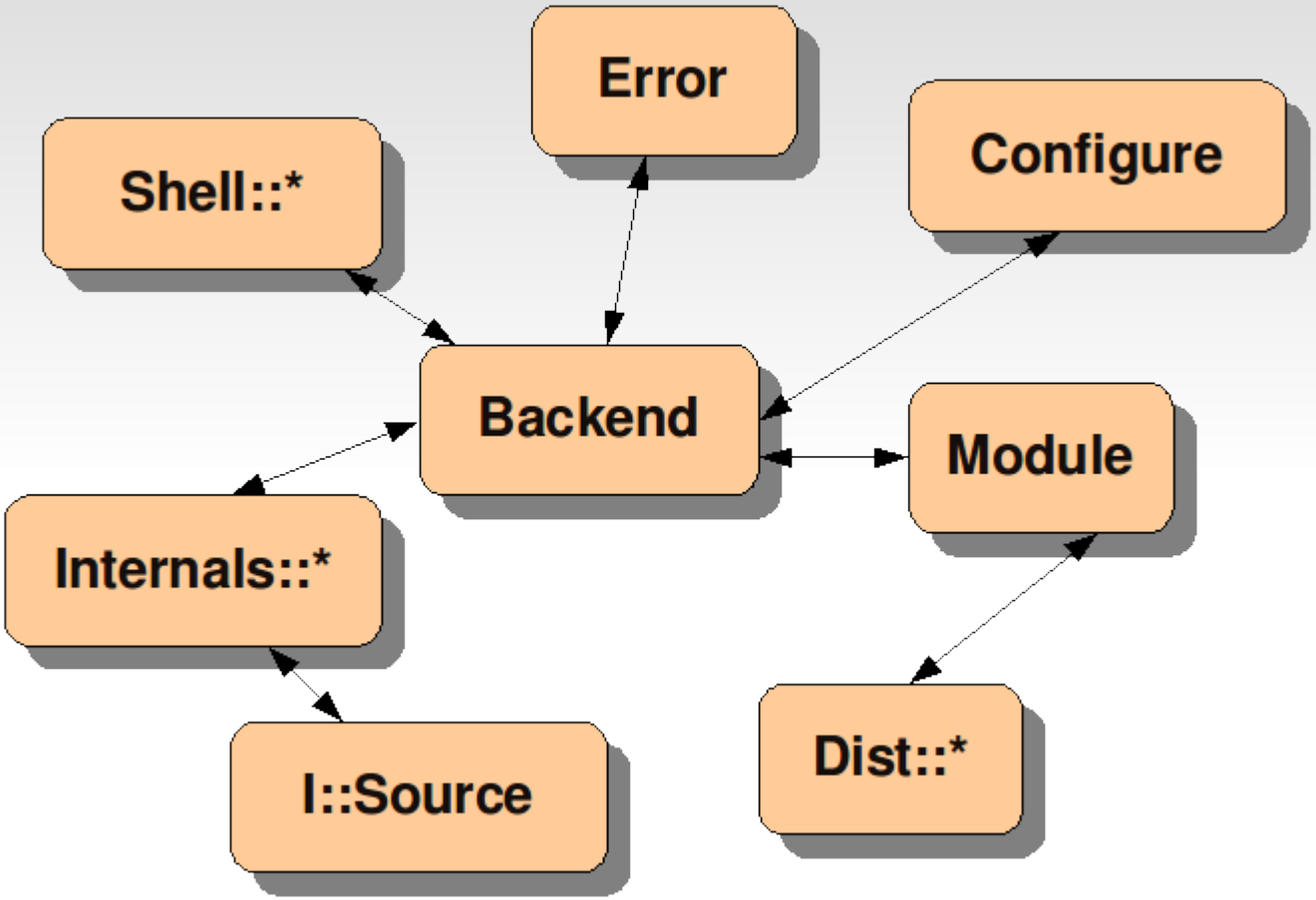
CPANPLUS

CPANPLUS

- 'do what CPAN.pm does, but do it better.'
- Modular design and API
- Different types of use
- Extendable

Structure




Error
Shell::*
Configure
Backend
Module
Internals::*
I::Source
Dist::*

Configuration

- Determines settings
- Two types: System and User
- Defaults and documented in CPANPLUS::Config
- 's reconfigure' in the shell
- Programmically configure:

```
use strict;
use warnings;
use CPANPLUS::Configure;

my $conf = CPANPLUS::Configure->new();
$conf->set_conf( verbose => 1 );
$conf->set_conf( makeflags => 'UNINST=1' );
$conf->set_conf( buildflags => 'uninst=1' );
$conf->set_conf( enable_custom_sources => 0 );
$conf->set_conf( show_startup_tip => 0 );
$conf->set_conf( write_install_logs => 0 );
$conf->save();
exit 0;
```

- User config is stored in user's home directory
  - `~/.cpanplus/lib/CPANPLUS/Config/User.pm`
- You can set env var APPDATA to change this
- Or use CPANPLUS::Config::BaseEnv

Module Objects

- Created from CPAN source indexes
- One for each module on CPAN
- Large number of accessor methods
- Some interesting methods:
  - install()
  - prepare()
  - create()
  - test()

Backend & API

- The programmer's interface to the CPANPLUS
- Various ways of accessing module objects
  - module_tree()
  - search()
  - parse_module()
- installed()
- local_mirror()
- autobundle()

```
use strict;
use warnings;
use CPANPLUS::Configure;
use CPANPLUS::Backend;

my $conf = CPANPLUS::Configure->new();
$conf->set_conf( cpantest => 0 );
$conf->set_conf( verbose => 1);
$conf->set_conf( prereqs => 1 );

my $cb = CPANPLUS::Backend->new( $conf );
$cb->reload_indices( update_source => 1 );
my $su = $cb->selfupdate_object;

$su->selfupdate( update => 'dependencies', latest => 1 );

$cb->module_tree( $_ )->install() for
      qw(
          CPANPLUS
          File::Temp
          Compress::Raw::Bzip2
          Compress::Raw::Zlib
          Compress::Zlib
          ExtUtils::CBuilder
          ExtUtils::ParseXS
          ExtUtils::Manifest
          Module::Build
          CPANPLUS::YACSmoke
          Test::Reporter::Transport::Socket
      );

$_->install() for map { $su->modules_for_feature( $_ ) }
     qw(prefer_makefile md5 storable cpantest);
```

```
use strict;
use warnings;
use CPANPLUS::Configure;
use CPANPLUS::Backend;

my $conf = CPANPLUS::Configure->new();
$conf->set_conf( verbose => 1);
$conf->set_conf( prereqs => 1 );

my $cb = CPANPLUS::Backend->new( $conf );
$cb->reload_indices( update_source => 1 );

$cb->local_mirror( path => '.', index_files => 1, verbose => 1 );
```

Internals

- Backend inherits from CP::Internals
- CP::Internals inherits from CP::Internals::*
- Provides a number of callbacks
- $cb->_register_callback()
- 'install_prerequisite'
- 'proceed_on_test_failure'

Distribution classes

- Two special installer Dist classes provided:
  - CPANPLUS::Dist::MM
  - CPANPLUS::Dist::Build
- CPANPLUS::Dist::Base for creating custom distribution classes
- cpan2dist to create dists in the format specified

Custom Dist Classes

- use base qw[CPANPLUS::Dist::Base];
- format_available()
- init()
- prepare()
- create()
- install()
- uninstall()

```
package CPANPLUS::Dist::PAR;
use strict;
use base    'CPANPLUS::Dist::Base';

use CPANPLUS::Error;
use File::Basename              qw[basename];
use Params::Check               qw[check];
use Module::Load::Conditional   qw[can_load];
use Locale::Maketext::Simple    Class => 'CPANPLUS', Style => 'gettext';

sub format_available {
    return unless can_load( modules => {
                            'PAR::Dist' => 0
                        } );
    return 1;
}
```

```
sub create {
    ### just in case you already did a create call for this module object
    ### just via a different dist object
    my $dist        = shift;
    my $self        = $dist->parent;
    my $dist_cpan   = $self->status->dist_cpan;
    $dist           = $self->status->dist   if      $self->status->dist;
    $self->status->dist( $dist )            unless  $self->status->dist;

    my $cb   = $self->parent;
    my $conf = $cb->configure_object;

    $dist->SUPER::create( @_ ) or return;

    msg( loc("Creating PAR dist of '%1'", $self->name), 1);

    ### par::dist is noisy, silence it
    ### XXX this doesn't quite work -- restoring STDOUT still has
    ### it closed
    #*STDOUT_SAVE = *STDOUT; close *STDOUT;
    my $par = eval {
        ### pass name and version explicitly, as parsing doesn't always
        ### work
        PAR::Dist::blib_to_par(
            path    => $self->status->extract,
             version => $self->package_version,
             name    => $self->package_name,
        );
    };
```

```
### error?
    if( $@ or not $par or not -e $par ) {
        error(loc("Could not create PAR distribution of %1: %2",
                  $self->name, $@ ));
        return;
    }

    my ($to,$fail);
    MOVE: {
        my $dir = File::Spec->catdir(
                        $conf->get_conf('base'),
                        CPANPLUS::Internals::Utils
                           ->_perl_version(perl => $^X),
                        $conf->_get_build('distdir'),
                        'PAR'
                    );
        my $to  = File::Spec->catfile( $dir, basename($par) );

        $cb->_mkdir( dir => $dir )              or $fail++, last MOVE;
        $cb->_move( file => $par, to => $to )   or $fail++, last MOVE;
        msg(loc("PAR distribution written to: '%1'", $to), 1);

        $dist->status->dist( $to );
    }

    return $dist->status->created(1) unless $fail;
    return;
}
```

Source Engines

- Two engines available with CPANPLUS:
  - Memory
  - SQLite
- Memory:
  - Hashes of author and module objects
- SQLite:
  - Uses DBI and DBD::SQLite
  - Tie interface to link hash structure to DB

```
$cb->_author_tree || $cb->_module_tree
    $cb->_check_trees
       $cb->__check_uptodate
            $cb->_update_source
       $cb->__update_custom_module_sources
            $cb->__update_custom_module_source
    $cb->_build_trees
       ### engine methods
       {   $cb->_init_trees;
           $cb->_standard_trees_completed
           $cb->_custom_trees_completed
       }
       $cb->__create_author_tree
          ### engine methods
          { $cb->_add_author_object }
       $cb->__create_module_tree
          $cb->__create_dslip_tree
          ### engine methods
          { $cb->_add_module_object }
       $cb->__create_custom_module_entries

$cb->_dslip_defs
```

- One other source engine available on CPAN
  - CPANPLUS::Internals::Source::CPANIDX
- Low-memory usage
- Uses CPANIDX website to get module data
- (Yeah, requires Internet access)
- Comes with two scripts in *examples/*

```
CPAN Terminal> s conf no_update 1
CPAN Terminal> s conf source_engine CPANPLUS::Internals::Source::CPANIDX
CPAN Terminal> s save
```

Custom Sources

- Add your own sources list to the index
- Put dist files in a repository
- Create an index (*packages.txt*) in the repo root
  - write_custom_source_index()
- In the shell use the CustomSource plugin
  - */cs --add <url>*
  - *s conf enable_custom_sources 1*

Shells & Plugins

- User interfaces providing access to CP::Backend etc.
- CPANPLUS comes with Default and Classic
- Others available on CPAN
- Default shell supports plugins:
  - CPANPLUS::Shell::Default::Plugins namespace
  - Detected using Module::Pluggable
- Comes with Remote, Source and CustomSources

- Remote plugin enables remote usage of CPANPLUS shell.
- Remote end runs CPANPLUS::Daemon

```
cpanpd -P 666 -u my_user -p secret
```

```
CPAN Terminal> /connect --user=my_user --pass=secret remotehost 666

    Connection accepted

    Successfully connected to 'localhost' on port '11337'

    Note that no output will appear until a command has completed
    -- this may take a while


CPAN Terminal@localhost> o; i *
```

On CPAN now

CPANPLUS::Dist::*

- CPANPLUS::Dist::Arch
- CPANPLUS::Dist::Deb
- CPANPLUS::Dist::Fedora
- CPANPLUS::Dist::Gentoo
- CPANPLUS::Dist::Mdv
- CPANPLUS::Dist::PAR
- CPANPLUS::Dist::RPM
- CPANPLUS::Dist::SUSE

CPANPLUS::Shell::*

- CPANPLUS::Shell::Wx
- CPANPLUS::Shell::Tk
- CPANPLUS::Shell::Curses
- CPANPLUS::Shell::Wx::PODReader

Questions?