YAPC::NA 2016

Perl Dependencies

@atoomic

Nicolas Rochelemagne

~~YAPC::~~NA 2016

Perl Dependencies

@atoomic

Nicolas Rochelemagne

YAPC::NA 2016

TPC

Perl Dependencies

@atoomic

Nicolas Rochelemagne

Perl Dependencies

1. Discovering INC
2. Listing perl dependencies
3. Process memory 101
4. Combining both ?
5. Memory details ?
6. Going further

bit.do/b7N79

Let’s play with %INC

@INC vs %INC

- **@INC**: **where** to **look** for modules

- **%INC**: **loaded** modules
  and where they come from

s01.pl

```perl
#!/usr/bin/env perl

use strict;
use warnings;

use v5.022;

say q{# @INC:};
map { say $_ } @INC;

say '';
say q{# %INC:};
map { say $_, ' => ', $INC{$_} } sort keys %INC;

1;
```

s01.pl

```perl
#!/usr/bin/env perl

use strict;
use warnings;

use v5.022;

say q{# @INC:};
map { say $_ } @INC;

say '';
say q{# %INC:};
map { say $_, ' => ', $INC{$_} } sort keys %INC;

1;
```

```
• [nicolas@MacBook:inc-talk] (master) > perl samples/s01.pl
# @INC:
/Users/nicolas/.dotfiles/perl-must-have/lib
/Users/nicolas/perl5/lib/perl5/
./lib
/Users/nicolas/perl5/perlbrew/perls/perl-5.22.1/lib/site_perl/5.22.1/darwin-2level
/Users/nicolas/perl5/perlbrew/perls/perl-5.22.1/lib/site_perl/5.22.1
/Users/nicolas/perl5/perlbrew/perls/perl-5.22.1/lib/5.22.1/darwin-2level
/Users/nicolas/perl5/perlbrew/perls/perl-5.22.1/lib/5.22.1
.

# %INC:
strict.pm => /Users/nicolas/perl5/perlbrew/perls/perl-5.22.1/lib/5.22.1/strict.pm
warnings.pm => /Users/nicolas/perl5/perlbrew/perls/perl-5.22.1/lib/5.22.1/warnings.pm
```

s02.pl

```perl
#!/usr/bin/env perl

use strict;
use warnings;

use v5.022;

use FindBin;

say q{# @INC:};
map { say $_ } @INC;

say '';
say q{# %INC:};
map { say $_, ' => ', $INC{$_} } sort keys %INC;

1;
```

s02.pl

```perl
#!/usr/bin/env perl

use strict;
use warnings;

use v5.022;

use FindBin;

say q{# @INC:};
map { say $_ } @INC;

say '';
say q{# %INC:};
map { say $_, ' => ', $INC{$_} } sort keys %INC;

1;
```

```
• [nicolas@MacBook:inc-talk] (master) > perl samples/s02.pl
# @INC:
/Users/nicolas/.dotfiles/perl-must-have/lib
/Users/nicolas/perl5/lib/perl5/
./lib
/Users/nicolas/perl5/perlbrew/perls/perl-5.22.1/lib/site_perl/5.22.1/darwin-2level
/Users/nicolas/perl5/perlbrew/perls/perl-5.22.1/lib/site_perl/5.22.1
/Users/nicolas/perl5/perlbrew/perls/perl-5.22.1/lib/5.22.1/darwin-2level
/Users/nicolas/perl5/perlbrew/perls/perl-5.22.1/lib/5.22.1
.

# %INC:
Carp.pm => /Users/nicolas/perl5/perlbrew/perls/perl-5.22.1/lib/5.22.1/Carp.pm
Cwd.pm => /Users/nicolas/perl5/perlbrew/perls/perl-5.22.1/lib/5.22.1/darwin-2level/Cwd.pm
Exporter.pm => /Users/nicolas/perl5/lib/perl5/Exporter.pm
File/Basename.pm => /Users/nicolas/perl5/perlbrew/perls/perl-5.22.1/lib/5.22.1/File/Basename.pm
File/Spec.pm => /Users/nicolas/perl5/perlbrew/perls/perl-5.22.1/lib/5.22.1/darwin-2level/File/Spec.pm
File/Spec/Unix.pm => /Users/nicolas/perl5/perlbrew/perls/perl-5.22.1/lib/5.22.1/darwin-2level/File/Spec/Unix.pm
FindBin.pm => /Users/nicolas/perl5/perlbrew/perls/perl-5.22.1/lib/5.22.1/FindBin.pm
XSLoader.pm => /Users/nicolas/perl5/perlbrew/perls/perl-5.22.1/lib/5.22.1/XSLoader.pm
constant.pm => /Users/nicolas/perl5/perlbrew/perls/perl-5.22.1/lib/5.22.1/constant.pm
strict.pm => /Users/nicolas/perl5/perlbrew/perls/perl-5.22.1/lib/5.22.1/strict.pm
vars.pm => /Users/nicolas/perl5/perlbrew/perls/perl-5.22.1/lib/5.22.1/vars.pm
warnings.pm => /Users/nicolas/perl5/perlbrew/perls/perl-5.22.1/lib/5.22.1/warnings.pm
warnings/register.pm => /Users/nicolas/perl5/perlbrew/perls/perl-5.22.1/lib/5.22.1/warnings/register.pm
• [nicolas@MacBook:inc-talk] (master) >
```

s03.pl

```perl
#!/usr/bin/env perl

use strict;
use warnings;

use v5.022;

use FindBin;
use lib "$FindBin::Bin/../lib";

use MyPackage ();

say q{# @INC:};
map { say $_ } @INC;

say '';
say q{# %INC:};
map { say $_, ' => ', $INC{$_} } sort keys %INC;

1;
```

MyPackage.pm

```perl
#!/bin/perl

package MyPackage;

use strict;
use warnings;

sub hello { print "hello\n" }

1;
```

s03.pl

```perl
#!/usr/bin/env perl

use strict;
use warnings;

use v5.022;

use FindBin;
use lib "$FindBin::Bin/../lib";

use MyPackage ();

say q{# @INC:};
map { say $_ } @INC;

say '';
say q{# %INC:};
map { say $_, '=>', $INC{$_} } keys %INC;

1;
```

MyPackage.pm

```perl
#!/bin/perl

package MyPackage;

use strict;
use warnings;
```

```
• [nicolas@MacBook:inc-talk] (master) > perl samples/s03.pl
# @INC:
/Users/nicolas/workspace/talks/inc-talk/samples/../lib
/Users/nicolas/.dotfiles/perl-must-have/lib
/Users/nicolas/perl5/lib/perl5/
./lib
/Users/nicolas/perl5/perlbrew/perls/perl-5.22.1/lib/site_perl/5.22.1/darwin-2level
/Users/nicolas/perl5/perlbrew/perls/perl-5.22.1/lib/site_perl/5.22.1
/Users/nicolas/perl5/perlbrew/perls/perl-5.22.1/lib/5.22.1/darwin-2level
/Users/nicolas/perl5/perlbrew/perls/perl-5.22.1/lib/5.22.1
.

# %INC:
Carp.pm => /Users/nicolas/perl5/perlbrew/perls/perl-5.22.1/lib/5.22.1/Carp.pm
Config.pm => /Users/nicolas/perl5/perlbrew/perls/perl-5.22.1/lib/5.22.1/darwin-2level/Config.pm
Cwd.pm => /Users/nicolas/perl5/perlbrew/perls/perl-5.22.1/lib/5.22.1/darwin-2level/Cwd.pm
Exporter.pm => /Users/nicolas/perl5/lib/perl5/Exporter.pm
File/Basename.pm => /Users/nicolas/perl5/perlbrew/perls/perl-5.22.1/lib/5.22.1/File/Basename.pm
File/Spec.pm => /Users/nicolas/perl5/perlbrew/perls/perl-5.22.1/lib/5.22.1/darwin-2level/File/Spec.pm
File/Spec/Unix.pm => /Users/nicolas/perl5/perlbrew/perls/perl-5.22.1/lib/5.22.1/darwin-2level/File/Spec/Unix.pm
FindBin.pm => /Users/nicolas/perl5/perlbrew/perls/perl-5.22.1/lib/5.22.1/FindBin.pm
MyPackage.pm => /Users/nicolas/workspace/talks/inc-talk/samples/../lib/MyPackage.pm
XSLoader.pm => /Users/nicolas/perl5/perlbrew/perls/perl-5.22.1/lib/5.22.1/XSLoader.pm
constant.pm => /Users/nicolas/perl5/perlbrew/perls/perl-5.22.1/lib/5.22.1/constant.pm
lib.pm => /Users/nicolas/perl5/perlbrew/perls/perl-5.22.1/lib/5.22.1/darwin-2level/lib.pm
strict.pm => /Users/nicolas/perl5/perlbrew/perls/perl-5.22.1/lib/5.22.1/strict.pm
vars.pm => /Users/nicolas/perl5/perlbrew/perls/perl-5.22.1/lib/5.22.1/vars.pm
warnings.pm => /Users/nicolas/perl5/perlbrew/perls/perl-5.22.1/lib/5.22.1/warnings.pm
warnings/register.pm => /Users/nicolas/perl5/perlbrew/perls/perl-5.22.1/lib/5.22.1/warnings/register.pm
• [nicolas@MacBook:inc-talk] (master) >
```

s04.pl

```perl
#!/usr/bin/env perl

use strict;
use warnings;

use v5.022;

use FindBin;
use lib "$FindBin::Bin/../lib";

use MyPackage ();
use MultiplePackages ();

say q{# @INC:};
map { say $_ } @INC;

say '';
say q{# %INC:};
map { say $_, ' => ', $INC{$_} } sort keys %INC;

1;
```

MultiplePackages.pm

```perl
#!/bin/perl

package MultiplePackages::Foo;

use strict;
use warnings;

sub hello { print "hello from ".__PACKAGE__."\n" }

package MultiplePackages::Bar;

use strict;
use warnings;

sub hello { print "hello from ".__PACKAGE__."\n" }


1;
```

s04.pl

```perl
#!/usr/bin/env perl

use strict;
use warnings;

use v5.022;

use FindBin;
use lib "$FindBin::Bin/../lib";

use MyPackage ();
use MultiplePackages ();

say q{# @INC
map { say $_
say '';
say q{# %INC
map { say $_

1;
```

MultiplePackages.pm

```perl
#!/bin/perl

package MultiplePackages::Foo;

use strict;
use warnings;

sub hello { print "hello from ".__PACKAGE__."\n" }

package MultiplePackages::Bar;
```

```
• [nicolas@MacBook:inc-talk] (master) > perl samples/s04.pl
# @INC:
/Users/nicolas/workspace/talks/inc-talk/samples/../lib
/Users/nicolas/.dotfiles/perl-must-have/lib
/Users/nicolas/perl5/lib/perl5/
./lib
/Users/nicolas/perl5/perlbrew/perls/perl-5.22.1/lib/site_perl/5.22.1/darwin-2level
/Users/nicolas/perl5/perlbrew/perls/perl-5.22.1/lib/site_perl/5.22.1
/Users/nicolas/perl5/perlbrew/perls/perl-5.22.1/lib/5.22.1/darwin-2level
/Users/nicolas/perl5/perlbrew/perls/perl-5.22.1/lib/5.22.1
.

# %INC:
Carp.pm => /Users/nicolas/perl5/perlbrew/perls/perl-5.22.1/lib/5.22.1/Carp.pm
Config.pm => /Users/nicolas/perl5/perlbrew/perls/perl-5.22.1/lib/5.22.1/darwin-2level/Config.pm
Cwd.pm => /Users/nicolas/perl5/perlbrew/perls/perl-5.22.1/lib/5.22.1/darwin-2level/Cwd.pm
Exporter.pm => /Users/nicolas/perl5/lib/perl5/Exporter.pm
File/Basename.pm => /Users/nicolas/perl5/perlbrew/perls/perl-5.22.1/lib/5.22.1/File/Basename.pm
File/Spec.pm => /Users/nicolas/perl5/perlbrew/perls/perl-5.22.1/lib/5.22.1/darwin-2level/File/Spec.pm
File/Spec/Unix.pm => /Users/nicolas/perl5/perlbrew/perls/perl-5.22.1/lib/5.22.1/darwin-2level/File/Spec/Unix.pm
FindBin.pm => /Users/nicolas/perl5/perlbrew/perls/perl-5.22.1/lib/5.22.1/FindBin.pm
MultiplePackages.pm => /Users/nicolas/workspace/talks/inc-talk/samples/../lib/MultiplePackages.pm
MyPackage.pm => /Users/nicolas/workspace/talks/inc-talk/samples/../lib/MyPackage.pm
XSLoader.pm => /Users/nicolas/perl5/perlbrew/perls/perl-5.22.1/lib/5.22.1/XSLoader.pm
constant.pm => /Users/nicolas/perl5/perlbrew/perls/perl-5.22.1/lib/5.22.1/constant.pm
lib.pm => /Users/nicolas/perl5/perlbrew/perls/perl-5.22.1/lib/5.22.1/darwin-2level/lib.pm
strict.pm => /Users/nicolas/perl5/perlbrew/perls/perl-5.22.1/lib/5.22.1/strict.pm
vars.pm => /Users/nicolas/perl5/perlbrew/perls/perl-5.22.1/lib/5.22.1/vars.pm
warnings.pm => /Users/nicolas/perl5/perlbrew/perls/perl-5.22.1/lib/5.22.1/warnings.pm
warnings/register.pm => /Users/nicolas/perl5/perlbrew/perls/perl-5.22.1/lib/5.22.1/warnings/register.pm
```

List dependencies

List dependencies without updating the script

hello-world.pl

```
1  #!/usr/bin/env perl
2
3  use strict;
4  use warnings;
5
6  use v5.014;
7
8  say q{hello world};
```

use-modules.pl

```
1  #!/usr/bin/env perl
2
3  use strict;
4  use warnings;
5
6  use v5.014;
7
8  use Carp      ();
9  use Config    ();
10 use Data::Dumper ();
11 use Digest    ();
12 use Encode    ();
13 use FindBin   ();
14
15 use MyPackage ();
16 use MultiplePackages ();
17
18 say q{Use some CORE modules};
19
20
```

List dependencies

```
1    package Devel::ListDeps;
2
3 ▼  CHECK {
4 ▼      foreach my $module ( sort keys %INC ) {
5            next if $module eq 'Devel/ListDeps.pm'; # ...
6            print qq{$module\n};
7        }
8    }
9
10   1;
```

List dependencies


```
> perl samples/hello-world.pl
```

```
> perl -Ilib -c -d:ListDeps samples/hello-world.pl
```

```
> perl -Ilib -c -d:ListDeps samples/use-modules.pl
```

```
10  analyze(@ARGV) unless caller;
11
12  sub analyze {
13      my (@args) = @_;
14
15      die unless scalar @args;
16
17      my $lib = "$FindBin::Bin/../lib";
18
19      foreach my $script_or_module (@args) {
20
21          my $file = -e $script_or_module ? $script_or_module : undef;
22          if ( !$file ) {
23              # $^xdoc :-) perldoc call
24              $file = qx{$^X -S perldoc -l $script_or_module};
25              die if $?;
26              chomp $file;
27          }
28
29          map { print } sort { lc($a) cmp lc($b) }
30           qx{$^X -I$lib -d:ListDeps -c $file 2>/dev/null};
31      }
32
33      return 0;
34  }
```

```perl
analyze(@ARGV) unless caller;

sub analyze {
    my (@args) = @_;

    # lazy getopt
    my $list_files = grep { $_ eq '--files' } @args;
    @args = grep { $_ ne '--files' } @args if $list_files;

    die unless scalar @args;

    my $lib = "$FindBin::Bin/../lib";

    foreach my $script_or_module (@args) {

        my $file = -e $script_or_module ? $script_or_module : undef;
        if ( !$file ) {
            $file = qx{$^X -S perldoc -l $script_or_module}; # perldoc call
            die if $?;
            chomp $file;
        }

        my @list = qx{$^X -I$lib -d:ListDeps -c $file 2>/dev/null};

        print join '', sort { lc($a) cmp lc($b) } map {
            if ( !$list_files ) {
                $_ =~ s{\.pm$}{};
                $_ =~ s{/+}{::}g;
            }
            $_;
        } @list;
    }

    return 0;
}
```

Memory profiling 101

Process memory 101

```
> perl -e 'print qx{grep VmRSS /proc/$$/status}'
VmRSS:     1836 kB
```

Process memory 101

```
> for i in $(seq 1 10); do perl -e 'print qx{grep VmRSS /proc/$$/status}'; done
VmRSS:        1836 kB
VmRSS:        1836 kB
VmRSS:        1832 kB
VmRSS:        1836 kB
VmRSS:        1832 kB
VmRSS:        1836 kB
VmRSS:        1836 kB
VmRSS:        1836 kB
VmRSS:        1832 kB
VmRSS:        1836 kB
```

Process memory 101

```
> perl -e 'print qx{grep VmRSS /proc/$$/status}'
VmRSS:     1836 kB
```

```
> perl -MCarp -e 'print qx{grep VmRSS /proc/$$/status}'
VmRSS:     2872 kB
```

Process memory 101

```
> perl -MCarp -e 'print qx{grep VmRSS /proc/$$/status}'
VmRSS:      2872 kB

> for i in $(seq 1 10); do perl -MCarp -e 'print qx{grep VmRSS /proc/$$/status}'; done
VmRSS:      2872 kB
VmRSS:      2872 kB
VmRSS:      2872 kB
VmRSS:      2868 kB
VmRSS:      2868 kB
VmRSS:      2872 kB
VmRSS:      2872 kB
VmRSS:      2872 kB
VmRSS:      2872 kB
VmRSS:      2872 kB
VmRSS:      2868 kB
```

Process memory 101

```
> perl -e 'print qx{grep VmRSS /proc/$$/status}'
VmRSS:     1836 kB
```

```
> perl -MCarp -e 'print qx{grep VmRSS /proc/$$/status}'
VmRSS:     2872 kB
```

```
> perl -MData::Dumper -e 'print qx{grep VmRSS /proc/$$/status}'
VmRSS:     3728 kB
```

Process memory 101

```
> perl -MConfig -e 'print qx{grep VmRSS /proc/$$/status}'
VmRSS:      2608 kB

> perl -MDigest -e 'print qx{grep VmRSS /proc/$$/status}'
VmRSS:      2584 kB

> perl -MEncode -e 'print qx{grep VmRSS /proc/$$/status}'
VmRSS:      3352 kB
```

```
> perl -e 'print qx{grep VmRSS /proc/$$/status}'
VmRSS:      1832 kB
```

```
> perl -e 'print qx{grep VmRSS /proc/$$/status}'
VmRSS:       1832 kB

> perl -MConfig -e 'print qx{grep VmRSS /proc/$$/status}'
VmRSS:       2608 kB
```

```
> perl -e 'print qx{grep VmRSS /proc/$$/status}'
VmRSS:      1832 kB

> perl -MConfig -e 'print qx{grep VmRSS /proc/$$/status}'
VmRSS:      2608 kB

> perl -MEncode -MConfig -e 'print qx{grep VmRSS /proc/$$/status}'
VmRSS:      3420 kB
```

```
> perl -e 'print qx{grep VmRSS /proc/$$/status}'
VmRSS:       1832 kB

> perl -MConfig -e 'print qx{grep VmRSS /proc/$$/status}'
VmRSS:       2608 kB

> perl -MEncode -MConfig -e 'print qx{grep VmRSS /proc/$$/status}'
VmRSS:       3420 kB

> perl -MEncode -MConfig -MPPI -e 'print qx{grep VmRSS /proc/$$/status}'
VmRSS:       9356 kB
```

```
> perl -e 'print qx{grep VmRSS /proc/$$/status}'
VmRSS:       1832 kB

> perl -MConfig -e 'print qx{grep VmRSS /proc/$$/status}'
VmRSS:       2608 kB

> perl -MEncode -MConfig -e 'print qx{grep VmRSS /proc/$$/status}'
VmRSS:       3420 kB

> perl -MEncode -MConfig -MPPI -e 'print qx{grep VmRSS /proc/$$/status}'
VmRSS:       9356 kB

> perl -MEncode -MConfig -MPPI -MTest::More -e 'print qx{grep VmRSS /proc/$$/status}'
VmRSS:      10740 kB
```

```
> perl -e 'print qx{grep VmRSS /proc/$$/status}'
VmRSS:       1832 kB

> perl -MConfig -e 'print qx{grep VmRSS /proc/$$/status}'
VmRSS:       2608 kB

> perl -MEncode -MConfig -e 'print qx{grep VmRSS /proc/$$/status}'
VmRSS:       3420 kB

> perl -MEncode -MConfig -MPPI -e 'print qx{grep VmRSS /proc/$$/status}'
VmRSS:       9356 kB

> perl -MEncode -MConfig -MPPI -MTest::More -e 'print qx{grep VmRSS /proc/$$/status}'
VmRSS:      10740 kB

> perl -MEncode -MConfig -MPPI -MTest::More -MMoose -e 'print qx{grep VmRSS /proc/$$/status}'
VmRSS:      23712 kB
```

use features or not ?

```
> perl -e 'print qx{grep VmRSS /proc/$$/status}'
VmRSS:     1832 kB
```

use features or not ?

```
> perl -e 'print qx{grep VmRSS /proc/$$/status}'
VmRSS:     1832 kB
```

```
> perl -E 'print qx{grep VmRSS /proc/$$/status}'
VmRSS:     2136 kB
```

Import or not ?

```
> perl -e 'use Net::IP (); print qx{grep VmRSS /proc/$$/status}'
VmRSS:      7068 kB
```

Import or not ?

```
> perl -e 'use Net::IP (); print qx{grep VmRSS /proc/$$/status}'
VmRSS:      7068 kB

> perl -e 'require Net::IP; print qx{grep VmRSS /proc/$$/status}'
VmRSS:      7012 kB
```

Import or not ?

```
> perl -e 'use Net::IP (); print qx{grep VmRSS /proc/$$/status}'
VmRSS:      7068 kB

> perl -e 'require Net::IP; print qx{grep VmRSS /proc/$$/status}'
VmRSS:      7012 kB
```

```
> perl -e 'use Net::IP; print qx{grep VmRSS /proc/$$/status}'
VmRSS:      7284 kB
```

Import or not ?

```
> perl -e 'use Net::IP (); print qx{grep VmRSS /proc/$$/status}'
VmRSS:      7068 kB

> perl -e 'require Net::IP; print qx{grep VmRSS /proc/$$/status}'
VmRSS:      7012 kB

> perl -e 'use Net::IP; print qx{grep VmRSS /proc/$$/status}'
VmRSS:      7284 kB

> perl -e 'use Net::IP qw/:PROC/; print qx{grep VmRSS /proc/$$/status}'
VmRSS:      7312 kB
```

perldoc Exporter

```
How to Import
  In other files which wish to use your module there are three basic ways
  for them to load your module and import its symbols:

  "use YourModule;"
      This imports all the symbols from YourModule's @EXPORT into the
      namespace of the "use" statement.

  "use YourModule ();"
      This causes perl to load your module but does not import any symbols.

  "use YourModule qw(...);"
      This imports only the symbols listed by the caller into their
      namespace. All listed symbols must be in your @EXPORT or @EXPORT_OK,
      else an error occurs. The advanced export features of Exporter are
      accessed like this, but with list entries that are syntactically
      distinct from symbol names.

  Unless you want to use its advanced features, this is probably all you
  need to know to use Exporter.
```

vi $(perldoc -l Net::IP)

```
# Global Variables definition
use vars qw($VERSION @ISA @EXPORT @EXPORT_OK %EXPORT_TAGS $ERROR $ERRNO
  %IPv4ranges %IPv6ranges $useBigInt
  $IP_NO_OVERLAP $IP_PARTIAL_OVERLAP $IP_A_IN_B_OVERLAP $IP_B_IN_A_OVERLAP $IP_IDENTICAL);

$VERSION = '1.26';

require Exporter;

@ISA = qw(Exporter);

# Functions and variables exported in all cases
@EXPORT = qw(&Error &Errno
  $IP_NO_OVERLAP $IP_PARTIAL_OVERLAP $IP_A_IN_B_OVERLAP $IP_B_IN_A_OVERLAP $IP_IDENTICAL
);

# Functions exported on demand (with :PROC)
@EXPORT_OK = qw(&Error &Errno &ip_iptobin &ip_bintoip &ip_bintoint &ip_inttobin
  &ip_get_version &ip_is_ipv4 &ip_is_ipv6 &ip_expand_address &ip_get_mask
  &ip_last_address_bin &ip_splitprefix &ip_prefix_to_range
  &ip_is_valid_mask &ip_bincomp &ip_binadd &ip_get_prefix_length
  &ip_range_to_prefix &ip_compress_address &ip_is_overlap
  &ip_get_embedded_ipv4 &ip_aggregate &ip_iptype &ip_check_prefix
  &ip_reverse &ip_normalize &ip_normal_range &ip_iplengths
  $IP_NO_OVERLAP $IP_PARTIAL_OVERLAP $IP_A_IN_B_OVERLAP $IP_B_IN_A_OVERLAP $IP_IDENTICAL
);
```

Coding style ?

```
use Code::Monkey  ();
use Busy::Camel   ();
use Banana        ();
```

vs

```
use A;
use B;
use C;
```

pros

> git grep **check**
T/P.pm:14: if (Code::Monkey::**check**(1..4)) {
E/R.pm:42: Busy::Camel::**check**(\&something)}
L/C.pm:51: $x = \&Banana::**check**;
…

> git grep **Something::check**
Y/A.pm:14: **Something::check**(‘a’..’c’) or die;
P/C.pm:42: if ($f && **Something::check**($f)) {

Mmmm I’ve got a problem

```
macOS> perl -e 'print qx{grep VmRSS /proc/$$/status}'
grep: /proc/63813/status: No such file or directory
```

Simple tweak

```
> perl -e 'print qx{ps -o rss -p $$ | tail -1}'
  1632
```

Yeah… It’s working

```
> perl -MDancer2 -e 'print qx{ps -o rss -p $$ | tail -1}'
22328
```

but this is bad…

Let’s wrap this

```
if ( -e q{/proc} ) { # unix
    print qx{grep VmRSS /proc/$$/status};
 } else { # mac os x
    print qx{ps -o rss -p $$ | tail -1};
}
```

```
> perl -c -d:ProcessMemory -MTest::More -e1
  4852
-e syntax OK
```

```
> perl -c -d:ProcessMemory -MTest::More -e1 2>/dev/null
 4868
```

there is a space

```
> perl -c -d:ProcessMemory -MTest::More -e1 2>/dev/null
4868
```

```perl
package Devel::ProcessMemory;

CHECK {

    if ( -e q{/proc} ) { # unix
        print int qx{grep VmRSS /proc/$$/status};
     } else { # mac os x
        print int qx{ps -o rss -p $$ | tail -1};
     }

    exit
}

1;
```

```
> perl -c -d:ProcessMemory -MTest::More -e1 2>/dev/null
4868

> perl -c -d:ProcessMemory ./samples/use-modules.pl 2>x
5460
```

```perl
#!/usr/bin/env perl

use strict;
use warnings;

use v5.014;

profile(@ARGV) unless caller;

sub profile {
    my (@args) = @_;

    foreach my $script_or_module (@args) {

        my $file = -e $script_or_module ? $script_or_module : undef;
        die "Invalid name: $script_or_module" if !$file && $script_or_module !~ qr{^[-:a-zA-z0-9_/\.]+$};

        if ( !$file ) {
            $file = qx{$^X -S perldoc -l $script_or_module}; # $^Xdoc :-)
            $? == 0 or die "Cannot find file for $script_or_module";
            chomp $file;
        }

        say qx[$^X -c -d:ProcessMemory $file 2>/dev/null];
    }

    return;
}
```

```
> ./perlprofile Test::More
3876
> ./perlprofile Data::Dumper
4276
> ./perlprofile samples/hello-world.pl
2436
> ./perlprofile samples/use-modules.pl
5836
```

```
> ./perlprofile lib/*.pm
lib/MultiplePackages.pm: 2380
lib/MyPackage.pm: 2300
lib/Tools.pm: 4936
```

very simple tools

1. list dependencies
2. check memory required by a module

very simple tools

1. list dependencies
2. check memory required by a module

simple, but we can build on top of them

very simple tools

1. list dependencies
2. check memory required by a module

simple, but we can build on top of them

- generate list of most used modules
- generate list of big memory consumers modules
  - => .csv
    - where should you be focused on

analyze evolution of memory required by a module:

- over versions
- over perl versions

List of modules

click on the column titles for sorting the table

Module	v5.22	v5.14	delta	% inc.
	14556	11208	3348	29 %
	13772	10616	3156	29 %
	11904	9152	2752	30 %
	11856	9112	2744	30 %
	11720	9004	2716	30 %
	10284	7916	2368	29 %
	9412	7320	2092	28 %
	9048	7040	2008	28 %
	8144	6384	1760	27 %
	7632	6000	1632	27 %
	7520	5888	1632	27 %
	7164	5668	1496	26 %
	6592	5204	1388	26 %
	6284	4992	1292	25 %
	6308	5020	1288	25 %
	6260	4972	1288	25 %
	5840	4636	1204	25 %
	5516	4404	1112	25 %
	5512	4400	1112	25 %
	5508	4396	1112	25 %
	5236	4188	1048	25 %
	5196	4152	1044	25 %
	5020	3992	1028	25 %
	3280	2260	1020	45 %
	4844	3844	1000	26 %
	4996	4012	984	24 %
	4812	3856	956	24 %
	4700	3764	936	24 %
	4568	3644	924	25 %
	4548	3632	916	25 %
	4524	3616	908	25 %
	4536	3660	876	23 %
	4500	3640	860	23 %
	4252	3404	848	24 %
	4216	3376	840	24 %
	4216	3380	836	24 %
	4364	3528	836	23 %
	4360	3524	836	23 %
	4340	3508	832	23 %
	4356	3532	824	23 %
	4284	3480	804	23 %
	4128	3344	784	23 %

Memory details ?


```
> perl -Ilib -d:ListDepsDetails ./samples/use-modules.pl
> perl -Ilib -d:ListDepsDetails -e 'require "./samples/use-modules.pl"'
> perl -Ilib -d:ListDepsDetails -e 'require Data::Dumper'
```

Let’s wrap this

> ./perlprofile ./samples/hello-world.pl

> ./perlprofile ./samples/hello-world.pl —details

> ./perlprofile Data::Dumper

```
profile(@ARGV);

sub profile {
    my (@args) = @_;

    foreach my $script_or_module (@args) {

        say "# Profiling module ", $script_or_module if $verbose;
        my $file = -e $script_or_module ? $script_or_module : undef;
        die "Invalid name: $script_or_module" if !$file && $script_or_module !~ qr{^[-:a-zA-z0-9_/\.]+$};

        my $esc = $file ? qq{"$file"} : $script_or_module;

        my $cmd;

        if ( !$details ) {    # default mode
            my $rss;
            if ( -e q{/proc} ) { # unix
                $rss = q{grep VmRSS /proc/$$/status};
            } else { # mac os x
                $rss = q{ps -o rss -p $$ | tail -1};
            }

            $cmd = qq{$^X -e 'require $esc; print int(qx{$rss}) . "\n";'};
        }
        else {
            $cmd = qq{$^X -I$FindBin::Bin/../ -d:ListDepsDetails -e 'require $esc'};
        }

        say "# Running: ", $cmd if $verbose;
        print "$script_or_module: ";
        print qx{$cmd};
    }

    return;
}
```

```
BEGIN {
    sub get_memory {
        my $m;
        if ( -e q{/proc} ) { # unix
            $m = int qx{grep VmRSS /proc/$$/status};
        } else { # mac os x
            $m = int qx{ps -o rss -p $$ | tail -1};
        }
        return $m;
    }

    my @inc = sort { $b cmp $a } @INC;
    sub short  {
        my $s = shift;

        foreach my $in ( @inc ) {
            $s =~ s{$in/}{} and return $s; # $poor editor
        }

        return $s;
    }

    my %seen;
    my $total_mem = 0;
    sub DB::DB {
        my ( $package, $file, $line ) = caller;

        return if $file eq '-e' || $file eq '-E';
        return if $seen{$file}++;

        my $code;
        {
            no strict 'refs';    ## no critic (ProhibitNoStrict)
            $code = \@{"::_<$file"};
        }
        $file ||= '';

        my $mem   = get_memory();
        my $delta = $mem - $total_mem;
        $total_mem = $mem;
        if ( keys %seen == 1 ) {
            print "# [delta => total RSS in kB] module name (or eval)\n";
        }

        # try to guess where it comes from (manual longmess :-)
        my ( $frompkg, $fromfile, $fromline ) = caller();
        my $max = 1_000;
        foreach my $level ( 0 .. $max ) {
            my ( $package, $filename, $line ) = caller($level);
            last unless defined $filename;
            if ( $fromfile ne $filename ) {    # when the filename differs, we know where it comes from
                ( $frompkg, $fromfile, $fromline ) = ( $package, $filename, $line );
                last;
            }
            if ( $level == $max ) {
                ( $frompkg, $fromfile, $fromline ) = ( '????', '????', '?' );
            }
        }

        print sprintf( "[%5s => %8d] %-50s from %-30s at line %d: %s\n", ( $delta > 0 ? '+' : '' ) . $delta, $mem, ( short($file) || 'undef' ), short($fromfile), $fromline, '' );

        return;
    }

}

CHECK { exit }
```

```
● [nicolas@MacBook:inc-talk] (master) [Push] > ./perlprofile --details Data::Dumper
# [delta => total RSS in kB] module name (or eval)
[+2184 =>     2184] Data/Dumper.pm                          from -e                       at line 1:
[  +52 =>     2236] Carp.pm                                 from Data/Dumper.pm           at line 22:
[   +4 =>     2240] strict.pm                               from Carp.pm                  at line 4:
[ +328 =>     2568] warnings.pm                             from Carp.pm                  at line 5:
[  +44 =>     2612] (eval 1)[Carp.pm:21]                    from Carp.pm                  at line 21:
[ +448 =>     3060] /Users/nicolas/perlperl5/Exporter.pm     from Carp.pm                  at line 99:
[  +40 =>     3100] XSLoader.pm                             from Data/Dumper.pm           at line 33:
[ +228 =>     3328] constant.pm                             from Data/Dumper.pm           at line 277:
[   +8 =>     3336] warnings/register.pm                    from constant.pm              at line 4:
[+1048 =>     4384] bytes.pm                                from Data/Dumper.pm           at line 754:
[ +156 =>     4540] overload.pm                             from Data/Dumper.pm           at line 20:
[    0 =>     4540] overloading.pm                          from overload.pm              at line 83:
```

```
• [nicolas@MacBook:inc-talk] (master) > ./perlprofile --details samples/use-modules.pl
# [delta => total RSS in kB] module name (or eval)
[+2240 =>     2240] sampleuse-modules.pl                          from -e                          at line 1:
[  +36 =>     2276] strict.pm                                     from sampleuse-modules.pl        at line 3:
[ +312 =>     2588] warnings.pm                                   from sampleuse-modules.pl        at line 4:
[  +84 =>     2672] Carp.pm                                       from sampleuse-modules.pl        at line 8:
[    0 =>     2672] (eval 1)[Carp.pm:21]                          from Carp.pm                     at line 21:
[ +412 =>     3084] /Users/nicolas/perlperl5/Exporter.pm           from Carp.pm                     at line 99:
[   +8 =>     3092] Config.pm                                     from sampleuse-modules.pl        at line 9:
[  +20 =>     3112] vars.pm                                       from Config.pm                   at line 11:
[   +8 =>     3120] warnings/register.pm                          from vars.pm                     at line 7:
[ +132 =>     3252] Data/Dumper.pm                                from sampleuse-modules.pl        at line 10:
[  +56 =>     3308] XSLoader.pm                                   from Data/Dumper.pm              at line 33:
[ +252 =>     3560] constant.pm                                   from Data/Dumper.pm              at line 277:
[+1048 =>     4608] bytes.pm                                      from Data/Dumper.pm              at line 754:
[ +152 =>     4760] overload.pm                                   from Data/Dumper.pm              at line 20:
[  +24 =>     4784] overloading.pm                                from overload.pm                 at line 83:
[  +88 =>     4872] Digest.pm                                     from sampleuse-modules.pl        at line 11:
[    0 =>     4872] Encode.pm                                     from sampleuse-modules.pl        at line 12:
[  +32 =>     4904] Encode/Alias.pm                               from Encode.pm                   at line 47:
[ +164 =>     5068] Encode/Config.pm                              from Encode.pm                   at line 52:
[  +20 =>     5088] Encode/Encoding.pm                            from Encode.pm                   at line 265:
[  +36 =>     5124] FindBin.pm                                    from sampleuse-modules.pl        at line 13:
[   +4 =>     5128] Cwd.pm                                        from FindBin.pm                  at line 83:
[ +424 =>     5552] File/Basename.pm                              from FindBin.pm                  at line 84:
[  +52 =>     5604] File/Spec.pm                                  from FindBin.pm                  at line 85:
[  +36 =>     5640] File/Spec/Unix.pm                             from File/Spec.pm                at line 22:
[ +292 =>     5932] liMyPackage.pm                                from sampleuse-modules.pl        at line 15:
[    0 =>     5932] liMultiplePackages.pm                         from sampleuse-modules.pl        at line 16:
```


```
> perl samples/use-tools.pl
> ./perlprofile samples/use-tools.pl
> ./perlprofile samples/use-tools-patched.pl
> ./perlprofile samples/use-tools-patched.pl —details
```

Blacklist a module

```
$INC{‘Do/Not/Need/It.pm’} = '__FAKE__';
```

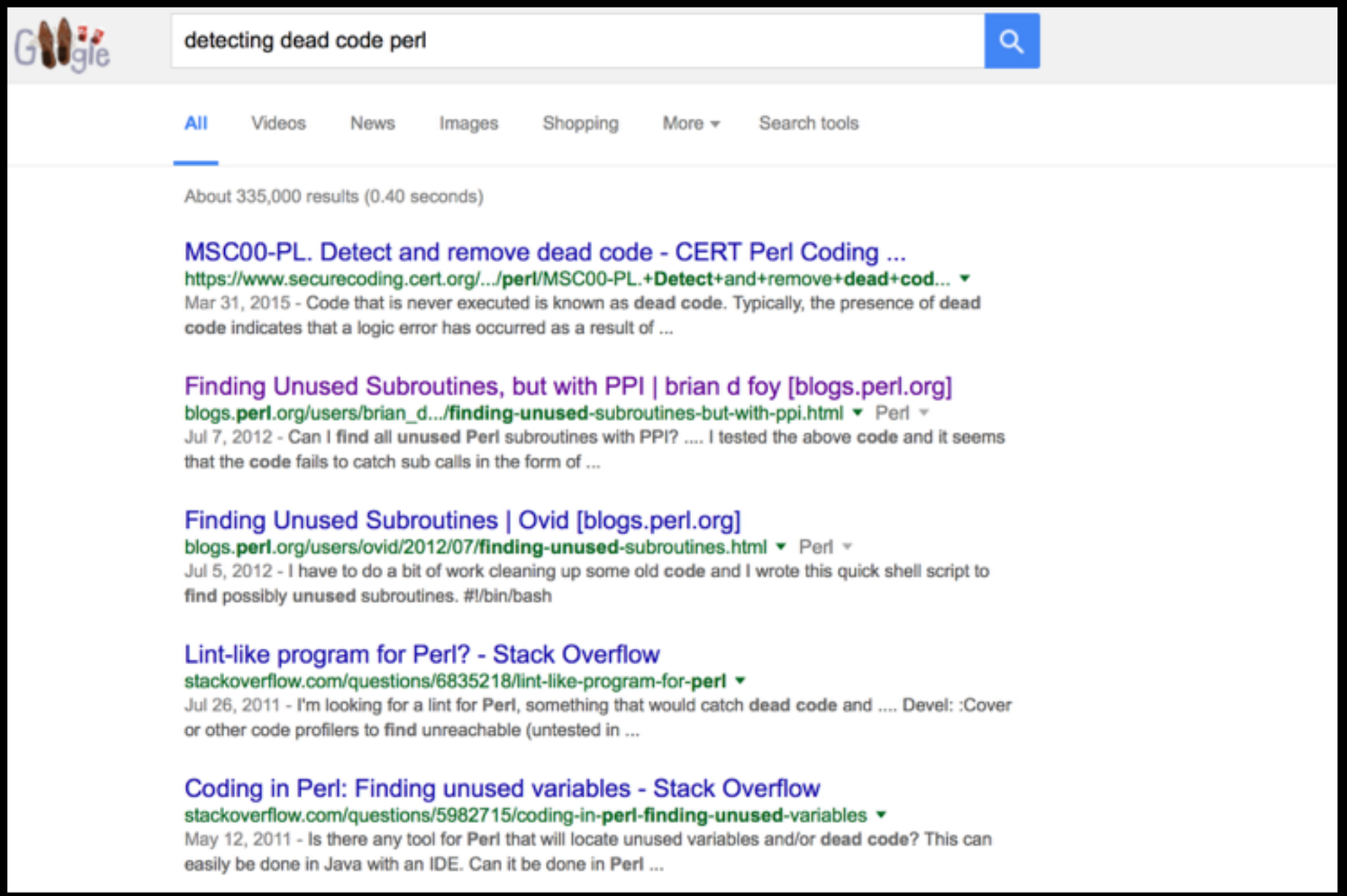
Blacklist a module

```
$INC{‘Do/Not/Need/It.pm’} = '__FAKE__';

*Do::Not::Need::It::do_something = sub {}
```

Going further

- Detecting dead code ?



Finding Unused Subroutines, but with PPI

By **brian d foy** on July 7, 2012 1:52 AM

Can I find all unused Perl subroutines with PPI? Ovid tried it with a quick shell script, which is probably good enough for his purposes, and certainly shorter than my solution.

```
#!/usr/bin/perl
use v5.14;
use strict;
use warnings;

use PPI;
use Scalar::Util qw(blessed);


# # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # #
# Create the PPI document, and add an isa method that takes a list
sub PPI::Element::isa {
        my( $self, @classes ) = @_;
        foreach my $class ( @classes ) {
                return 1 if $self->UNIVERSAL::isa( $class );
                }
        return 0;
        }

my $Document = PPI::Document->new( 'subs.pl' );
die "Could not create PDOM!" unless blessed $Document;

# # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # #
# Get all of the subroutine definitions
my @subs =
        map { $_->name }
        @{ $Document->find(
                sub {
                        $_[1]->isa( 'PPI::Statement::Sub' )
                        }
        ) };
say "All sub definitions: @subs";


# # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # #
# find the sub calls that use &
#       &foo
#       &foo()
#       \&foo
my @symbols =
        map  { $_->content =~ s/\A&//r; }
        @{ $Document->find(
```

About brian d foy

I'm the author of Mastering Perl, and the co-author of Learning Perl (6th Edition), Intermediate Perl, Programming Perl (4th Edition) and Effective Perl Programming (2nd Edition).

More info »

Finding Unused Subroutines

By **Ovid** on July 5, 2012 4:02 PM

Posting this here to help me remember this.

I have to do a bit of work cleaning up some old code and I wrote this quick shell script to find *possibly* unused subroutines.

```
#!/bin/bash

tempfile=/tmp/$$allsubs.txt

ack '^\s*sub\s+(\w+)\b' lib |      \
    awk '/sub (\w*)/ { print $2 }' | \
    cut -d'(' -f1 |                  \
    sort |                           \
    uniq > $tempfile

for sub in $(cat $tempfile); do
    if [[ $(expr `git grep -E "\<$sub\>" |wc -l`) == 1 ]]; then
        echo $sub
    fi
done

rm $tempfile
```

My `bash` skills are awful (see above), but I've already found 25 subroutines that can probably be deleted. I say "probably" because all have to be investigated.

Let’s play with them


```
> cat samples/unused-1.pl
> perl blog.brian.find_unused_subs.pl samples/unused-1.pl
> perl blog.brian.find_unused_subs.pl samples/use-tools-patched.pl
```

Let’s pack it

we need a name ?

Let’s pack it

YAPC ?

Let’s pack it

TPC

The Packaging Camel ?

Let’s pack it

The Packaging Camel ?

Let’s pack it

YAPaCk

Let’s pack it

yapack

Let’s pack it

yapack

Ye[st] another packing (script)?

A quick glance to yapack ?


```
> ./yapack samples/use-tools-patched.pl
> perl blog.brian.find_unused_subs.pl samples/use-tools-patched.pl.fat
```

detecting unused code

What Could Possibly Go Wrong?

- Difficulties / limits:
  - methods
  - **dynamic**
  - base / parent

find unused subs


```
> perl ./find_unused_subs samples/unused-1.pl
> perl ./find_unused_subs samples/unused-1.pl —debug
```

What Could Possibly Go Wrong?

TDD ? find unused subs

- let’s add (/see) some tests
- prove -v t/find_unused_subs.t

find_include_discrepancies

- detect extra use/require statements
- detect missing usr/require statements

get-hostname sample

```
> ./get-hostname.pl
yapc.my.hostname.tld
```

get-hostname sample

```
> wc -l get-hostname.pl
12 get-hostname.pl
> perl-dependencies get-hostname.pl | wc -l
9
# including warnings and strict
```

get-hostname sample

```
> find_unused_subs get-hostname.pl
....
# after 5 parses:
# - removed 16 functions
# - get rid of 3 packages
# - one package was reduced by ~62 %
# - another one package was reduced by ~33 %
```

get-hostname sample

> ls -l get-hostname.pl.fat.*
-rw-r--r-- 1 root root 11873 Jun 20 14:05 get-hostname.pl.fat.cut.pl
-rw-r--r-- 1 root root 19134 Jun 20 14:04 get-hostname.pl.fat.original

> wc -l get-hostname.pl.fat.*
  413 get-hostname.pl.fat.cut.pl
  623 get-hostname.pl.fat

get-hostname sample

```
> perlcc -v2 -O3 -UO -UB::C -ohostname.fat.compiled get-hostname.pl.fat
> perlcc -v2 -O3 -UO -UB::C -ohostname.fat.cut.compiled get-hostname.pl.fat.cut.pl

> ls -lh hostname.fat.compiled hostname.fat.cut.compiled
-rwxr-xr-x 1 root root 1.1M Jun 20 14:14 hostname.fat.compiled
-rwxr-xr-x 1 root root 963K Jun 20 14:14 hostname.fat.cut.compiled
```

get-hostname sample

> time for i in $(seq 1 10000); do ./hostname.fat.compiled >/dev/null; done

real 0m50.769s
user 0m31.905s
sys 0m19.372s

real 0m51.916s
user 0m32.288s
sys 0m20.240s

> time for i in $(seq 1 10000); do ./hostname.fat.cut.compiled >/dev/null; done

real 0m47.547s
user 0m29.158s
sys 0m18.859s

real 0m46.743s
user 0m28.628s
sys 0m18.679s

Is it worth it ?

Is it worth it ?

Yes, it’s fun !

Is it worth it ?

It’s fun !

Saving Memory vs Saving Speed

Should I care about it ?

probably not

Memory:
- how many servers ?
- your servers ?

Speed:
- how many queries / sec ?
- your servers ?

~balance: impact of your savings vs time spent

What’s next ?

- set of simple tools: let’s keep it simple (need some cleanup)
- PPI parsing of eval blocks:
    - PPI::Token::Quote::Single ''
    - PPI::Token::Quote::Literal "q{}"
    - PPI::Token::Quote::Double ""
    - PPI::Token::Quote::Interpolate "qq{}"
- **PPI::Xref**: generate cross-references for Perl code
- combining unused subs + include-discrepancies
- increase tests
- packaging ?
- have also a look at **Devel::QuickCover**
- **AutoLoad, AutoSplit ?**

The End

@atoomic

bit.do/b7N79