X shell

Spencer Tipping

February 21, 2014

Contents

Part I

Bootstrap implementation

Chapter 1

Self-replication

Listing 1.1 boot/xh-header

```
#!/usr/bin/env perl
BEGIN {eval(our $xh_bootstrap = q{
# xh: the X shell | https://github.com/spencertipping/xh
# Copyright (C) 2014, Spencer Tipping
# Licensed under the terms of the MIT source code license

# For the benefit of HTML viewers (long story):
# <body style='display:none'>
# <script src='http://spencertipping.com/xh/page.js'></script>
use 5.014;
package xh;
our %modules;
our @module_ordering;

our %compilers = (pl => sub {
  my $package = $_[0] =~ s/\./::/gr;
  eval "{package ::$package;\n$_[1]\n}";
  die "error compiling module $_[0]: $@" if $@;
});

sub defmodule {
  my ($name, $code, @args) = @_;
  chomp($modules{$name} = $code);
  push @module_ordering, $name;
  my ($base, $extension) = split /\.(\w+$)/, $name;
  die "undefined module extension '$extension' for $name"
    unless exists $compilers{$extension};
  $compilers{$extension}->($base, $code, @args);
}
```

```

chomp($modules{bootstrap} = $::xh_bootstrap);
undef $::xh_bootstrap;
```

At this point we need a way to reproduce the image. Since the bootstrap code is already stored, we can just wrap it and each defined module into an appropriate `BEGIN` block.

Listing 1.2 boot/xh-header (continued)

```
sub image {
  my @pieces = "#!/usr/bin/env perl";
  push @pieces, "BEGIN {eval(our \$xh_bootstrap = <<'_')}",
                $modules{bootstrap},
                '_';
  push @pieces, "BEGIN {xh::defmodule('$_', <<'_')}",
                $modules{$_},
                '_' for @module_ordering;
  push @pieces, "xh::main::main;\n__DATA__";
  join "\n", @pieces;
}
})}
```

Chapter 2

Data structures

All values in xh have the same type, which provides a bunch of operations suited to different purposes. This implementation is based on strings and, as a result, has egregious performance appropriate only for bootstrapping the self-hosting compiler.

Listing 2.1 modules/v.pl

```perl
BEGIN {xh::defmodule('xh::v.pl', <<'_')}
sub parse_with_brackets {
  my ($regexp, $filler, $x) = @_;
  $regexp = qr/$regexp/;
  my @initial_split = split /$regexp/, $x;

  @initial_split = grep length, @initial_split if $regexp =~ /\(/;
  my $item;
  my @result;
  my $bracket_count = 0;

  for my $data (@initial_split) {
    $bracket_count += length($data =~ s/\\.|[^\[({]//gr);
    $bracket_count -= length($data =~ s/\\.|[^\])}]//gr);
    $item = length($item) ? "$item$filler$data" : $data;
    unless ($bracket_count) {
      push @result, $item;
      $item = '';
    }
  }

  push @result, $item if $item;
  @result;
}

```

```
sub parse_lines {parse_with_brackets '\v',                 "\n", @_}
sub parse_words {parse_with_brackets '\s',                 " ",  @_}
sub parse_path  {parse_with_brackets '(/[^\[\](){}\s/]*)', "",   @_}
_
```

Chapter 3

Perl compiler

This is the dumbest thing we can possibly do to make xh runnable.

Listing 3.1 `modules/compile.pl`

```
BEGIN {xh::defmodule('xh::compile.pl', <<'_')}
sub xh_to_perl {
  # TODO
}
_
```