Hello Embed Perl!

Hideaki Ohno

YAPC::Asia Tokyo 2011

About me

- Hideaki Ohno
  - Github: hideo55
  - Twitter: @hide_o_55
  - PAUSE: HIDEAKIO
  - Blog: http://d.hatena.ne.jp/hide_o_55/
  - Perl/C/C++/JavaScript

Agenda

- What's embed perl?
- Why use embed perl
- Common sense of embed perl

Hello embed perl!

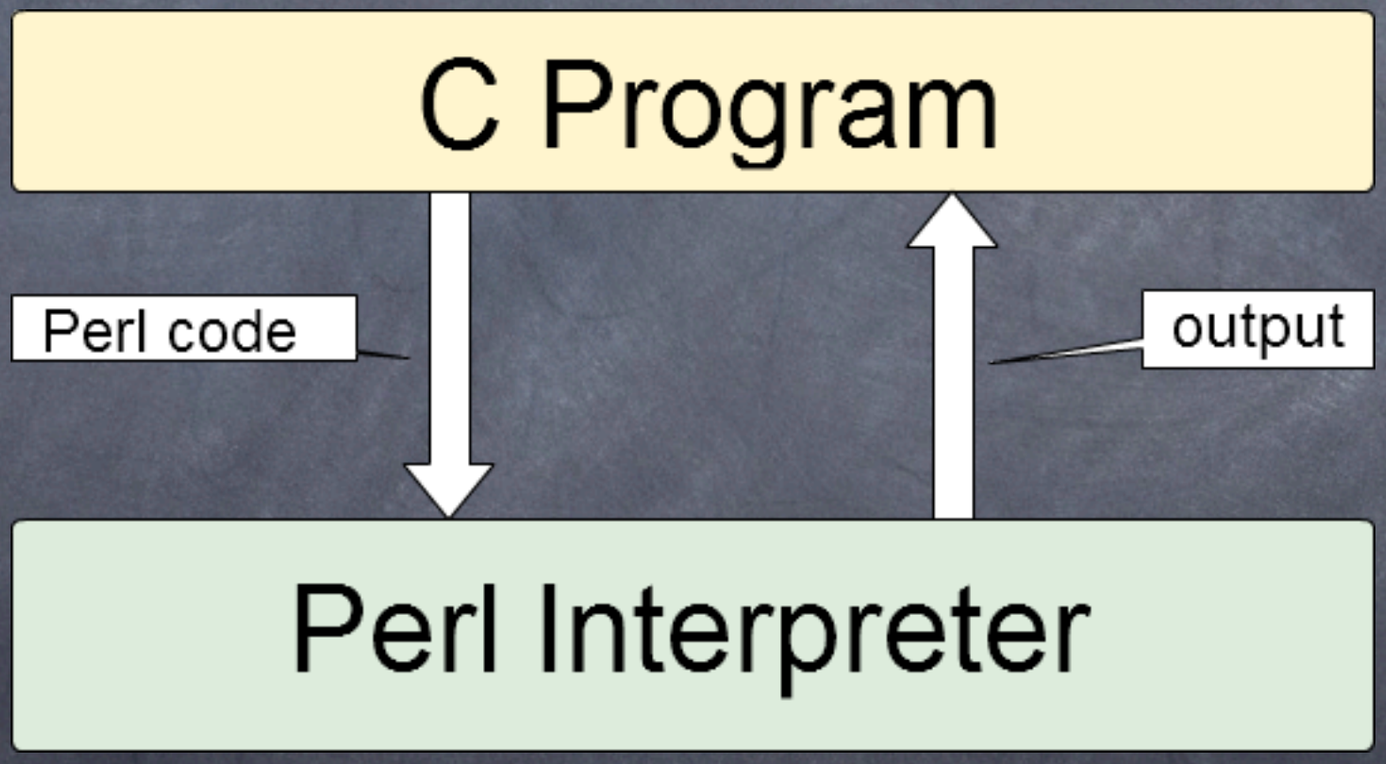
embed perl?

What's embed perl?

What's embed perl?

- Using Perl From C
  - Adding a Perl interpreter to C program
  - Calling a Perl subroutine from C program
  - Evaluting a Perl statement from C program
  - Performing Perl pattern matches and substitutions from C program
  - Fidding with the Perl stack from C program

Adding a Perl interpreter to C program



Example of embed perl

- mod_perl (Apache)
- http_perl_module (Nginx)
- PL/Perl (PostgreSQL)
- MyPerl (MySQL)

Example of embed perl

- mod_perl (Apache)
- http_perl_module (Nginx)
- PL/Perl (PostgreSQL)
- MyPerl (MySQL)
- node-perl (Node.js)

Why use embed perl?

(What are merits of embed perl)

Why use embed perl?

Regexp:::

Why use embed perl?

Regexp:::

Don't needs PCRE because perl's regexp engine is 100% perl compatible!

Why use embed perl?

CPAN!!!

Common sense of embed perl

Perl data type

SV	SCALAR value
AV	ARRAY value
HV	HASH value
GV	Glob value
RV	Reference value

Basic usage

```
#include <EXTERN.h>
#include <perl.h>

static PerlInterpreter *my_perl;

int main(int argc, char **argv, char **env)
{
    PERL_SYS_INIT3(&argc,&argv,&env);
    my_perl = perl_alloc();
    perl_construct(my_perl);
    PL_exit_flags |= PERL_EXIT_DESTRUCT_END;
    perl_parse(my_perl, NULL, argc, argv, (char **)NULL);
    perl_run(my_perl);
    perl_destruct(my_perl);
    perl_free(my_perl);
    PERL_SYS_TERM();
}
```

Include

#include<EXTERN.h>
Define symbols for specific OSs such as Windows.

#include<perl.h>
Define symbols and include header files in accordance with compile options

Include

- From C++

```
extern “C” {
#include<EXTERN.h>
#include<perl.h>
}
```

Create interpreter

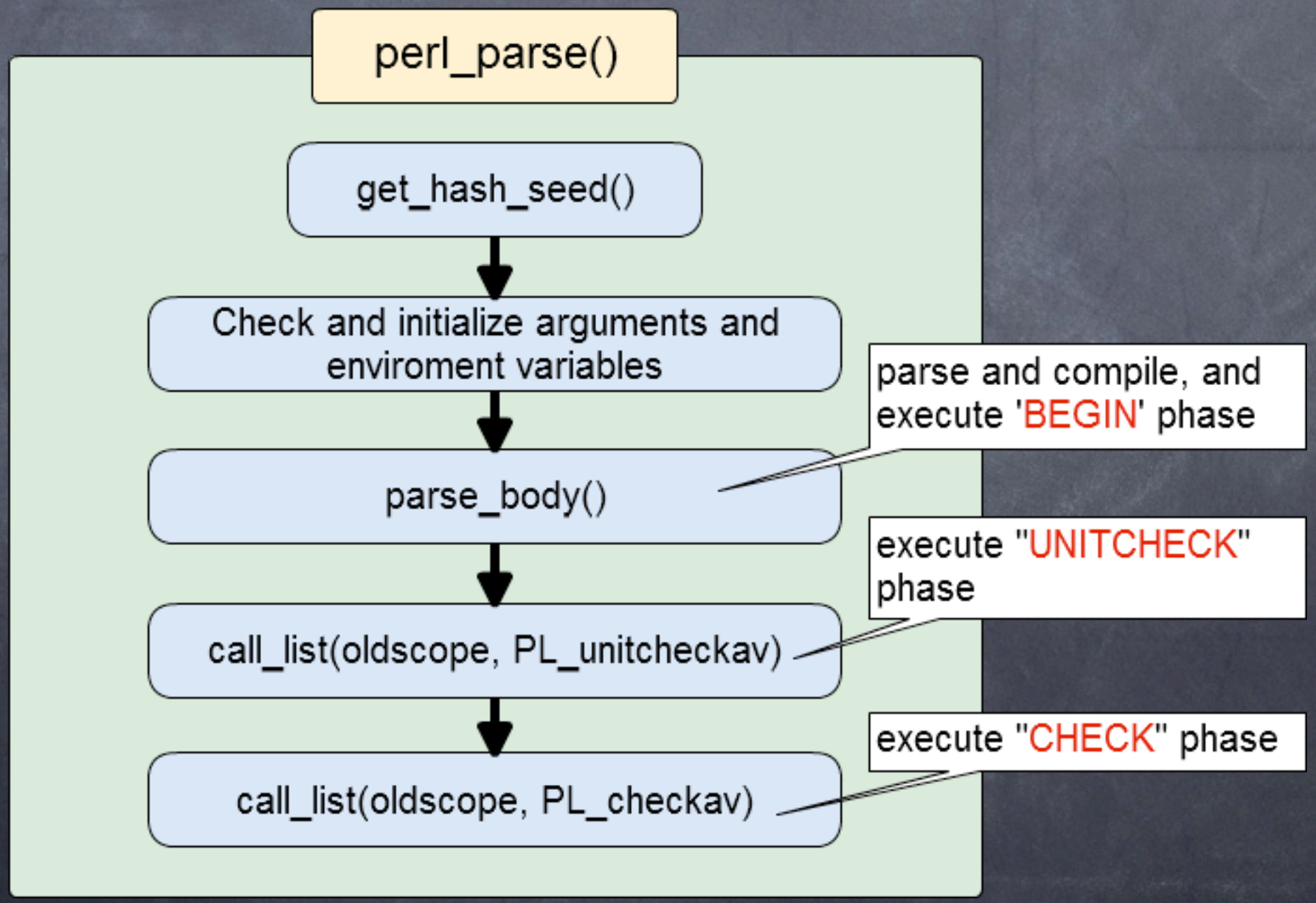
```
PerlInterpreter *perl = perl_alloc();
perl_construct(perl);
```

If you want use multiple interpreter, use PERL_SET_CONTEXT().

It initialize global state for trace "current" interpreter.

Parse

perl_perse()
This function parse and compile perl code.



Parse

```
perl_parse(
  register PerlInterpreter *my_perl,
  XSINIT_t xsinit,
  int argc,
  char** argv,
  char** env
);
```

xsinit is function pointer of initialize XS module.

```
perl -MExtUtils::Embed -e xsinit -- -o perlxsi.c
```

Run

perl_run()

- Execute "INIT" Block
- Execute perl code
- mod_perl execute this function once at initialize time.
- nginx perl module don't execute this function.

Why nginx don't execute perl_run()?

```
PL_exit_flags |= PERL_EXIT_DESTRUCT_END;
```

If set PL_exit_flags
PERL_EXIT_DESTRUCT_END, "END" Block is executed in perl_destruct().

Destructor

perl_destruct()

- Execute "END" block.
- Objects destruction

perl_free()

- Free allocated memory

Q. How to get result from C program?

A. TIMTOWDI

Call subroutine and get return value

```
dSP;
ENTER;
SAVETMPS;
PUSHMARK(sp);
XPUSHs(...); //push some variables to stack
PUTBACK;
count = call_sv(sub, G_EVAL);
SPAGAIN;
if (count != 1) croak(“”);
x = SvPVx(POPs, n_a);
```

Override PerlIO layers

Changing the destination for STDOUT/STDERR to scalar variables.

Override PerlIO layers

```
extern "C" {
#define PERLIO_NOT_STDIO 0
#define USE_PERLIO
#include <EXTERN.h>
#include <perl.h>
}
```

See also perlio.h

Override PerlIO layers

```
void override_stdhandle (pTHX_ SV *sv,const char *name ) {
    int status;
    GV *handle = gv_fetchpv(name,TRUE,SVt_PVIO);
    SV *svref = newRV_inc(sv);
    save_gp(handle, 1);
    status = Perl_do_open9(aTHX_ handle, ">:scalar", 8 , FALSE, O_WRONLY,
      0, Nullfp, svref, 1);
    if(status == 0) {
        Perl_croak(aTHX_ "Failed to open %s: %" SVf,name, get_sv("!",TRUE));
    }
}
```

Override PerlIO layers

```
void restore_stdhandle (pTHX_ const char *name) {
    int status;
    GV *handle = gv_fetchpv(name,FALSE,SVt_PVIO);

    if( GvIOn(handle) && IoOFP(GvIOn(handle)) &&
(PerlIO_flush(IoOFP(GvIOn(handle))) == -1 ) ) {
        Perl_croak(aTHX_ "Failed to flush %s: "
          SVf,name,get_sv("!",TRUE) );
    }
}
```

Override PerlIO layers

```
{
  local *STDOUT;
  my $stdout;
  open STDOUT, ‘>’, ¥$stdout or die $!;
  ...
}
```

Compile

```
use ExtUtils::Embed
```

```
% cc -o myperl myperl.c `perl -MExtUtils::Embed
-e ccopts -e ldopts`
```

See ‘perldoc perlembed’ if you want to know other way of to compile embed perl.

Others

- eval
- eval_pv()
- eval_sv()

Reference

- perldoc perlembed
- perldoc perlcall
- perldoc perlapi
- perldoc perlxs
- perldoc perlguts
- perldoc perlapio
- Book - Extending and Embedding Perl

Summary

- Embed perl is not difficult
- If you need the power of Perl in your C program, then hesitate to jump into the world of embedded Perl
- By challenging the embedded Perl, even deeper understanding of Perl

Thanks for your attention.