Předehra: Nastavujeme překladač Céčka (ExtUtils::Embed)

```
perl -MExtUtils::Embed -e ccopts
    -D_REENTRANT -D_GNU_SOURCE -DDEBIAN
    -D_LARGEFILE_SOURCE -D_FILE_OFFSET_BITS=64
    -fstack-protector -fno-strict-aliasing
    -pipe
    -I/usr/local/include -I/usr/lib/perl/5.14/CORE

perl -MExtUtils::Embed -e ldopts
    -Wl,-E
    -fstack-protector
    -L/usr/local/lib -L/usr/lib/perl/5.14/CORE
    -lperl -ldl -lm -lpthread -lc -lcrypt
```

Viz též:

```
perl -MConfig 'print $Config{ccflags}'
perl -V::ccflags:
perl '-V:.*'
```

Minimalistický /usr/bin/perl (perlembed)

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

int main(int argc, char **argv, char **env)
{
  PERL_SYS_INIT3(&argc, &argv, &env);
  PerlInterpreter *my_perl = perl_alloc();
  perl_construct(my_perl);
  PL_exit_flags |= PERL_EXIT_DESTRUCT_END;

  perl_parse(my_perl, NULL, argc, argv, NULL);
  perl_run(my_perl);

  perl_destruct(my_perl);
  perl_free(my_perl);
  PERL_SYS_TERM();
}
```

Volání funkce (perlembed)

```
int main(int argc, char **argv, char **env)
{
  PERL_SYS_INIT3(&argc, &argv, &env);
  PerlInterpreter *my_perl = perl_alloc();
  perl_construct(my_perl);
  PL_exit_flags |= PERL_EXIT_DESTRUCT_END;

  char *av[] = { argv[0], "script.pl", NULL };
  perl_parse(my_perl, NULL, 2, av, NULL);
  perl_run(my_perl);

  char *par[] = { "42", NULL };
  call_argv("answer", G_DISCARD, par);

  perl_destruct(my_perl);
  perl_free(my_perl);
  PERL_SYS_TERM();
}
```

Zvěřinec typů (perlguts)

Perlové typy:

`SV`	skalár
`AV`	pole
`HV`	heš
`GV`	typeglob

Pomocné typy:

`IV`	integer
`UV`	unsigned integer
`NV`	number (double)
`PV`	pointer (typicky na string)
`RV`	reference na perlový typ

Práce se skaláry 1/2 (perlguts, perlapi)

```
// Výroba
sv = newSViv(42);
sv = newSVpv("Hello!", 0);  // 0 = délka se spočítá automaticky
sv = newSVpvf("%d", 42);
sv = newSVsv(sv);
sv = newSV(len);            // undef s bufferem dané délky

// Čtení
SvIV(sv)
SvNV(sv)
SvPV(sv, len)               // makro modifikující len
SvPV_nolen(sv)
SvTRUE(sv)

// Změna hodnoty
sv_setiv(sv, 1);
sv_setpv(sv, "Goodbye!");
```

Práce se skaláry 2/2 (perlguts, perlapi)

```
// Vlastnosti
SvOK(sv)                    // defined(...) ?
SvIOK(sv)                   // má integerovou hodnotu (veřejnou) ?
SvPOK(sv)                   // má stringovou hodnotu (veřejnou) ?

// Výroba polymorfního skaláru
sv_setiv(sv, 1);                    // SvIOK
sv_setpv(sv, "Goodbye!");           // SvPOK, !SvIOK (jen privátně)
SvIOK_on(sv);                       // SvPOK, SvIOK

// Standardní konstanty (read-only!)
PL_sv_undef
PL_sv_yes
PL_sv_no

// Nalezení skaláru podle jména
sv = get_sv("pkg::var", GV_ADD);
```

Reference a smrtelnost (perlguts, perlapi)

```
// Počítání odkazů
cnt = SvREFCNT(sv)
SvREFCNT_inc(sv);        // vrací tutéž sv
SvREFCNT_dec(sv);

// Perlové reference (speciální případ SV)
rv = newRV_inc(sv)       // může být i přetypovaná AV/HV/...
rv = newRV_noinc(sv)
sv = SvRV(rv)            // výsledek lze přetypovat
SvROK(sv)
SvTYPE(svRV(rv))         // SVt_PVAV pro pole atd.

// Smrtelné skaláry
sv_2mortal(sv);          // objedná pozdější SvREFCNT_dec
sv = sv_newmortal()
sv2 = sv_mortalcopy(sv) // smrtelná kopie skaláru
```

Magie (perlguts)

Každý skalár může mít jednu nebo více `struct magic`. Obsahují:

- typ
- soukromá data
- odkaz na tabulku virtuálních funkcí: pre-get, post-set, free, ...

Příklady druhů magie:

- `pos`
- `tie`
- hodnota v `%ENV`
- hodnota v `@ISA`
- délka pole (`$#array`)

Některé funkce magii vyvolávají samy, jinak je nutno volat:

```
SvGETMAGIC  (před čtením)
SvSETMAGIC  (po zápisu)
```

Předávání argumentů a výsledků (perlcall, perlapi)

```
dSP;
ENTER;
SAVETMPS;

PUSHMARK(SP);
XPUSHs(sv_2mortal(newSViv(1))); // nebo mXPUSHi(1)
XPUSHs(sv_2mortal(newSViv(2)));
PUTBACK;

int nres = call_pv("add", G_SCALAR);    // kontext
if (nres != 1)
  croak("Nevermore!");

SPAGAIN;
printf("Result = %d\n", POPi);
PUTBACK;

FREETMPS;
LEAVE;
```

```
...

PUSHMARK(SP);
mXPUSHi(1);
mXPUSHi(2);
PUTBACK;

int cnt = call_pv("subtract", G_SCALAR | G_EVAL);

SPAGAIN;
if (SvTRUE(ERRSV))                                  // $@
  printf("Error: %s", SvPV_nolen(ERRSV));
else if (cnt == 1)
  printf("Result = %d\n", POPi);
else
  printf("Unexpected result\n");
PUTBACK;

...
```

Vyrábíme Céčkovou knihovnu (perlxstut, perlxs)

Necháme si poradit:

```
h2xs -A -n Oracle::Delphi --skip-ppport
```

```
Oracle-Delphi/
   Makefile.PL
   Delphi.xs
   lib/Oracle/Delphi.pm
   t/Oracle-Delphi.t
   MANIFEST
   README
   Changes
```

Céčková knihovna: Makefile.PL (ExtUtils::MakeMaker)

```
use ExtUtils::MakeMaker;

WriteMakefile(
    NAME           => 'Oracle::Delphi',
    VERSION_FROM   => 'lib/Oracle/Delphi.pm',
    AUTHOR         => 'Martin Mares <mj@ucw.cz>',
    LIBS           => ['-lm'],
);
```

```
perl Makefile.PL INSTALL_BASE=~/perl
make             -> blib/
make test        -> spustí t/*.t a případně test.pl
make install
```

Céčková knihovna: MANIFEST (ExtUtils::Manifest)

```
Changes
Delphi.xs
Makefile.PL
MANIFEST
README
t/Oracle-Delphi.t
lib/Oracle/Delphi.pm
```

Céčková knihovna: Delphi.pm (XSLoader)

```
package Oracle::Delphi;

use 5.014002;
use strict;
use warnings;

require Exporter;
our @ISA = qw(Exporter);
our @EXPORT = qw(prophecy);
our $VERSION = '0.01';

require XSLoader;
XSLoader::load('Oracle::Delphi', $VERSION);

# Zde může být kus modulu napsaný v Perlu

1;

__END__
Zde jednou bude dokumentace...
```

```
#include "EXTERN.h"
#include "perl.h"
#include "XSUB.h"

MODULE = Oracle::Delphi   PACKAGE = Oracle::Delphi

char *
prophecy(question)
INPUT:
        char *question;
CODE:
        RETVAL = "Great empire will be destroyed.";
OUTPUT:
        RETVAL
```

INPUT: lze vynechat a parametry deklarovat jako v ANSI C
CODE: lze vynechat, pak se zavolá Céčková funkce téhož jména

XS: Definice vlastního typu (perlxs)

```
#include "EXTERN.h"
#include "perl.h"
#include "XSUB.h"

typedef enum {
        OM_AUGURY,
        OM_HARUSPICY,
        OM_SPIRITISM,
} oracle_mode;

MODULE = Oracle::Delphi   PACKAGE = Oracle::Delphi

char *
prophecy(oracle_mode mode, char *question)
CODE:
        RETVAL = "Great empire will be destroyed.";
OUTPUT:
        RETVAL
```

XS: Překlad vlastního typu (perlxstypemap)

Soubor typemap:

```
TYPEMAP
oracle_mode_num T_IV
oracle_mode     T_ORACLE_MODE

INPUT
T_ORACLE_MODE
        char *tmp = SvPV_nolen($arg);
        if (!strcmp(tmp, \"augury\"))
                $var = OM_AUGURY;
        ...
        else
                croak(\"Invalid oracle mode\");
```

Podobně OUTPUT ...

XS: Přímá konstrukce SV (perlxs)

```
SV *
prophecy(oracle_mode mode, char *question)
CODE:
        RETVAL = newSVpv("Offer more if you want to ask: ", 0);
        sv_catpv(RETVAL, question);
OUTPUT:
        RETVAL
```

XS: Ruční obsluha argumentů i výsledku (perlxs)

```
void
prophecy(...)
PREINIT:
    int i;
PPCODE:
    if (items < 1) croak("Only barbarians ask no questions");
    for (i=0; i<items; i++) puts(SvPV_nolen(ST(i)));
    if (GIMME_V == G_ARRAY) {
        mXPUSHs(newSVpv("Sacrifice", 0));   // mXPUSHp nechceme
        mXPUSHi(100);
        mXPUSHs(newSVpv("computer bugs", 0));
        XSRETURN(3);
    } else {
        ST(0) = sv_2mortal(newSVpv("Empire will be ...", 0));
        sv_setiv(ST(0), 42);
        SvPOK_on(ST(0));
        XSRETURN(1);
    }
```

XS: Příklad volání (Devel::Peek)

```
#!/usr/bin/perl
use common::sense;
use Oracle::Delphi;
use Devel::Peek;

say prophecy("Will I", "pass the exam?");
say scalar prophecy("Will I?");
say 0 + prophecy("Will I?");
Dump(scalar prophecy("Will I?"));
```

```
SV = PVIV(0x1da35b8) at 0x1d91998
  REFCNT = 1
  FLAGS = (TEMP,IOK,POK,pIOK,pPOK)
  IV = 42
  PV = 0x1db0680 "Great empire will be destroyed"\0
  CUR = 30
  LEN = 32
```

Op-tree: Interní reprezentace programu (perlinterp, B::Terse)

```
perl -MO=Terse -e '$a=$b+$c'
LISTOP (0x1a3a240) leave [1]
    OP (0x1a2cfc0) enter
    COP (0x1a3aae0) nextstate
    BINOP (0x1a39010) sassign
        BINOP (0x1a3a200) add [4]
            UNOP (0x1a3a2c0) null [15]
                PADOP (0x1a3a390) gvsv  GV (0x1a31e68) *b
            UNOP (0x1a38f70) null [15]
                PADOP (0x1a3a280) gvsv  GV (0x1a14a88) *c
        UNOP (0x1a3a550) null [15]
            PADOP (0x1a3a6f0) gvsv  GV (0x1a31e38) *a
```

Op-tree: Pořadí provádění (perlinterp, B::Terse)

```
perl -MO=Terse,-exec -e '$a=$b+$c'
OP (0xc8efc0) enter
COP (0xdc09e0) nextstate
PADOP (0xc9c3b0) gvsv  GV (0xc93e68) *b
PADOP (0xc9c2a0) gvsv  GV (0xc76a88) *c
BINOP (0xc9c220) add [4]
PADOP (0xc9c710) gvsv  GV (0xc93e38) *a
BINOP (0xc9b030) sassign
LISTOP (0xc9c260) leave [1]
```