Описание typemap для передачи структур в XS

Ступницкий Иван
Инженер, YADRO

Perl-Conf.Ru/25

О чём расскажу

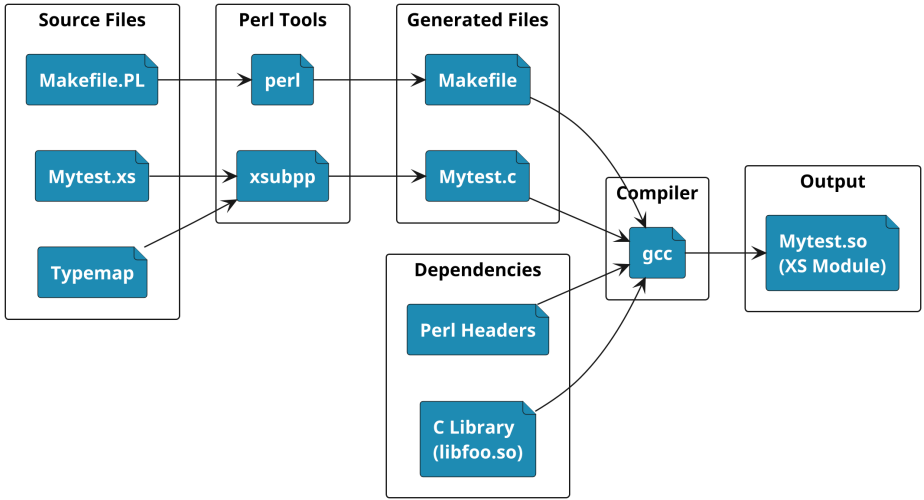
1. Кто я
2. Как переводится название доклада
3. Подопытная библиотека
4. XS в двух словах
5. В чём вообще проблема
6. Способы передачи структур в XS
7. Передача структур из XS в Perl
8. Как и что в итоге использовать

Ступницкий Иван

Инженер, YADRO

- Два года программирую на Perl за деньги
- Увлекаюсь информационной безопасностью и сложными системами

Как переводится название доклада



Подопытная библиотека libfoo


```c
// gcc -shared -fPIC -o libfoo.so foo.c
#include <stdio.h>
#include "foo.h"
void foo() {
    fprintf(stdout, "japh,\n");
}
```

```c
#ifndef FOO
#define FOO
extern void foo();
#endif /* FOO */
```

Вызов libfoo из Си


```c
// gcc -I../foo -o bar bar.c -L../foo -lfoo
#include "foo.h"
int main() {
   foo();
   return 0;
}
```

```
$ LD_LIBRARY_PATH=$PWD/../foo ./bar
japh,
```

Что почитать про XS

- perlxstut
- perlxs
- perlguts
- perlapi
- perlxstypemap

XS в двух словах


```
$ h2xs -A -n Mytest
Defaulting to backwards compatibility with perl 5.36.3
If you intend this module to be compatible with earlier perl
   versions, please
specify a minimum perl version with the -b option.

Writing Mytest/ppport.h
Writing Mytest/lib/Mytest.pm
Writing Mytest/Mytest.xs
Writing Mytest/Makefile.PL
Writing Mytest/README
Writing Mytest/t/Mytest.t
Writing Mytest/Changes
Writing Mytest/MANIFEST
```

XS в двух словах

Perl-Conf.Ru/25

```
$ h2xs -A -n Mytest
Defaulting to backwards compatibility with perl 5.36.3
If you intend this module to be compatible with earlier perl
   versions, please
specify a minimum perl version with the -b option.

Writing Mytest/ppport.h      # XS compatibility for old Perls
Writing Mytest/lib/Mytest.pm # Main module with public interface
Writing Mytest/Mytest.xs     # XS code implementing C functions
Writing Mytest/Makefile.PL   # Build configuration for the module
Writing Mytest/README        # Installation and usage instructions
Writing Mytest/t/Mytest.t    # Test suite for module functionality
Writing Mytest/Changes       # Module version changelog
Writing Mytest/MANIFEST      # List of files in distribution
```

Основной модуль


```
1  package Mytest;
2  use 5.036003;
3  use strict;
4  use warnings;
5  require Exporter;
6
7  our @ISA = qw(Exporter);
8  our %EXPORT_TAGS = ( 'all' => [ qw() ] );
9  our @EXPORT_OK = ( @{ $EXPORT_TAGS{'all'} } );
10 our @EXPORT = qw();
11 our $VERSION = '0.01';
12
13 require XSLoader;
14 XSLoader::load('Mytest', $VERSION);
15
16 1;
17 __END__
```

Mytest.xs


```
1  #define PERL_NO_GET_CONTEXT
2  #include "EXTERN.h"
3  #include "perl.h"
4  #include "XSUB.h"
5
6  #include "ppport.h"
7
8
9  MODULE = Mytest     PACKAGE = Mytest
10
```

Вызов libfoo из XS


```
1  #define PERL_NO_GET_CONTEXT
2  #include "EXTERN.h"
3  #include "perl.h"
4  #include "XSUB.h"
5
6  #include "ppport.h"
7
8  #include "foo.h"
9
10
11 MODULE = Mytest     PACKAGE = Mytest
12
13 void
14 xs()
15     CODE:
16         foo();
```

Makefile.PL


```
use 5.036003;
use ExtUtils::MakeMaker;
WriteMakefile(
    NAME              => 'Mytest',
    VERSION_FROM      => 'lib/Mytest.pm', # finds $VERSION
    PREREQ_PM         => {}, # e.g., Module::Name => 1.1
    ABSTRACT_FROM     => 'lib/Mytest.pm', # finds abstract
    AUTHOR            => 'i.stup@yadro.com',
    #LICENSE           => 'perl',
    #Value must be from legacy list of licenses here
    #https://metacpan.org/pod/Module::Build::API
    LIBS              => [''], # e.g., '-lm'
    DEFINE            => '', # e.g., '-DHAVE_SOMETHING'
    INC               => '-I.', # e.g., '-I. -I/usr/include/other'
    # Un-comment this if you add C files to link with later:
    # OBJECT            => '$(O_FILES)', # link all the C files
);
```

Makefile.PL


```
use 5.036003;
use ExtUtils::MakeMaker;
WriteMakefile(
    NAME              => 'Mytest',
    VERSION_FROM      => 'lib/Mytest.pm', # finds $VERSION
    PREREQ_PM         => {}, # e.g., Module::Name => 1.1
    ABSTRACT_FROM     => 'lib/Mytest.pm', # finds abstract
    AUTHOR            => 'i.stup@yadro.com',
    #LICENSE           => 'perl',
    #Value must be from legacy list of licenses here
    #https://metacpan.org/pod/Module::Build::API
    LIBS              => ['-L../foo -lfoo'],
    DEFINE            => '', # e.g., '-DHAVE_SOMETHING'
    INC               => '-I. -I../foo',
    # Un-comment this if you add C files to link with later:
    # OBJECT            => '$(O_FILES)', # link all the C files
);
```

Тестирование


```
1 #!/usr/bin/perl
2 use ExtUtils::testlib;
3 use Mytest;
4 Mytest::xs();
```

Тестирование


```
#!/usr/bin/perl
use ExtUtils::testlib;
use Mytest;
Mytest::xs();
```

```
$ perl Makefile.PL
...
$ make
...
$ perl test.pl
japh,
```

Структура в libfoo


```
1 // gcc -shared -fPIC -o libfoo.so foo.c
2 #include <stdio.h>
3 #include "foo.h"
4 void foo(struct foo variable) {
5     fprintf(stdout, "%d %s", variable.number, variable.string);
6 }
```

Структура в libfoo


```c
// gcc -shared -fPIC -o libfoo.so foo.c
#include <stdio.h>
#include "foo.h"
void foo(struct foo variable) {
    fprintf(stdout, "%d %s", variable.number, variable.string);
}
```

```c
#ifndef FOO
#define FOO
struct foo {
    int number;
    char* string;
};
extern void foo(struct foo);
#endif /* FOO */
```

Вот и всё!

Perl-Conf.Ru/25

```
1  MODULE = Mytest     PACKAGE = Mytest
2
3  void
4  xs(int number, char* string)
5      CODE:
6          struct foo var = { number, string };
7          foo(var);
```

Вот и всё!

Perl-Conf.Ru/25

```
MODULE = Mytest     PACKAGE = Mytest

void
xs(int number, char* string)
    CODE:
        struct foo var = { number, string };
        foo(var);
```

```
#!/usr/bin/perl
use ExtUtils::testlib;
use Mytest;
Mytest::xs(13 => "japh,\n");
```

Вот и всё!


```
MODULE = Mytest     PACKAGE = Mytest

void
xs(int number, char* string)
    CODE:
        struct foo var = { number, string };
        foo(var);
```

```
#!/usr/bin/perl
use ExtUtils::testlib;
use Mytest;
Mytest::xs(13 => "japh,\n");
```

```
$ perl test.pl
13 japh,
```

Вот и всё! (нет)


```
MODULE = Mytest     PACKAGE = Mytest

void
xs(int number, char* string)
    CODE:
        struct foo var = { number, string };
        foo(var);
```

```
#!/usr/bin/perl
use ExtUtils::testlib;
use Mytest;
Mytest::xs(13 => "japh,\n");
```

```
$ perl test.pl
13 japh,
```

В XS нужно проще

Perl-Conf.Ru/25

```
1  MODULE = Mytest     PACKAGE = Mytest
2
3  void
4  foo()
```

В XS нужно проще


```
MODULE = Mytest     PACKAGE = Mytest

void
foo()
```

```
#!/usr/bin/perl
use ExtUtils::testlib;
use Mytest;
Mytest::foo();
```

В XS нужно проще


```
MODULE = Mytest     PACKAGE = Mytest

void
foo()
```

```
#!/usr/bin/perl
use ExtUtils::testlib;
use Mytest;
Mytest::foo();
```

```
$ perl Makefile.PL
...
$ make
...
$ perl test.pl
japh,
```

А теперь со структурой

Perl-Conf.Ru/25

```
1 MODULE = Mytest     PACKAGE = Mytest
2
3 void
4 foo(struct foo var)
```

А теперь со структурой


```
MODULE = Mytest     PACKAGE = Mytest

void
foo(struct foo var)
```

```
$ make
...
"/usr/bin/perl" "/usr/lib/perl5/5.36.3/ExtUtils/xsubpp" \
    -typemap '/usr/lib/perl5/5.36.3/ExtUtils/typemap'   \
    Mytest.xs > Mytest.xsc
Could not find a typemap for C type 'struct foo'.
The following C types are mapped by the current typemap:
'AV *', 'Boolean', 'CV *', 'FILE *', 'FileHandle', 'HV *',
'const char *', 'double', 'float', 'int', 'long', 'short',
...
 in Mytest.xs, line 14
make: *** [Makefile:359: Mytest.c] Error 1
```

Что такое typemap

TODO

T_PACKED


```
1 struct foo           T_PACKED
```

T_PACKED


```
struct foo          T_PACKED
```

```
struct foo XS_unpack_struct_foo(SV *var) {
    return (struct foo){666, "it works\n"};
}

MODULE = Mytest     PACKAGE = Mytest

void
foo(struct foo var)
```

T_PACKED fuckup


```
Mytest.c: In function 'XS_xs_foo':
Mytest.c:176:31: error: 'XS_unpack_struct' undeclared (first
use in this function); did you mean 'XS_unpack_struct_foo'?
  176 |     struct foo      var = XS_unpack_struct foo(ST(0))
      |                           ^~~~~~~~~~~~~~~~
      |                           XS_unpack_struct_foo
Mytest.c:176:31: note: each undeclared identifier is reported
only once for each function it appears in
Mytest.c:176:48: error: expected ',' or ';' before 'foo'
  176 |     struct foo      var = XS_unpack_struct foo(ST(0))
      |                                            ^~~
make: *** [Makefile:341: Mytest.o] Error 1
```

Фикс для T_PACKED


```
1  struct foo           T_PACKED_PATCHED
2
3  INPUT
4  T_PACKED_PATCHED
5      # $var = XS_unpack_$ntype($arg)
6      $var = XS_unpack_${(my $nt = $ntype) =~ s/\s/_/g; \$nt}($arg)
```

Фикс для T_PACKED


```
struct foo          T_PACKED_PATCHED

INPUT
T_PACKED_PATCHED
    # $var = XS_unpack_$ntype($arg)
    $var = XS_unpack_${(my $nt = $ntype) =~ s/\s/_/g; \$nt}($arg)
```

```
$ perl test.pl
666 it works
```

Как это выглядит в Си


```
1  XS_EUPXS(XS_Mytest_foo); /* prototype to pass -Wmissing-prototypes
       */
2  XS_EUPXS(XS_Mytest_foo)
3  {
4      dVAR; dXSARGS;
5      if (items != 1)
6         croak_xs_usage(cv,  "var");
7      {
8      struct foo    var = XS_unpack_struct_foo(ST(0))
9  ;
10
11     foo(var);
12     }
13     XSRETURN_EMPTY;
14 }
```

Избежать вызова функции


```
1  struct foo           T_STRUCT_FOO
2
3  INPUT
4  T_STRUCT_FOO
5      $var.number = 777;
6      $var.string = \"this too\\n\";
```

Избежать вызова функции

Perl-Conf.Ru/25

```
struct foo          T_STRUCT_FOO

INPUT
T_STRUCT_FOO
    $var.number = 777;
    $var.string = \"this too\\n\";
```

```
$ perl test.pl
777 this too
```

А теперь без хардкода

Perl-Conf.Ru/25

```
1  struct foo          T_STRUCT_FOO
2
3  INPUT
4  T_STRUCT_FOO
5      $var.number = SvIV(*hv_fetchs((HV*)SvRV($arg),
6          \"number\", FALSE));
7      $var.string = SvPV_nolen(*hv_fetchs((HV*)SvRV($arg),
8          \"string\", FALSE));
```

А теперь без хардкода

Perl-Conf.Ru/25

```
struct foo          T_STRUCT_FOO

INPUT
T_STRUCT_FOO
    $var.number = SvIV(*hv_fetchs((HV*)SvRV($arg),
        \"number\", FALSE));
    $var.string = SvPV_nolen(*hv_fetchs((HV*)SvRV($arg),
        \"string\", FALSE));
```

```
#!/usr/bin/perl
use ExtUtils::testlib;
use Mytest;
Mytest::foo({number => 13, string => "hash\n"});
```

А теперь без хардкода


```
struct foo          T_STRUCT_FOO

INPUT
T_STRUCT_FOO
    $var.number = SvIV(*hv_fetchs((HV*)SvRV($arg),
        \"number\", FALSE));
    $var.string = SvPV_nolen(*hv_fetchs((HV*)SvRV($arg),
        \"string\", FALSE));
```

```
#!/usr/bin/perl
use ExtUtils::testlib;
use Mytest;
Mytest::foo({number => 13, string => "hash\n"});
```

```
$ perl test.pl
13 hash
```

Финальный typemap [1]


```
struct foo          T_STRUCT_FOO

INPUT
T_STRUCT_FOO
    if (!SvROK($arg) || SvTYPE(SvRV($arg)) != SVt_PVHV) {
        croak(\"$var is not a hash reference\");
    }

    HV* hash = (HV*)SvRV($arg);
    SV** sv_number = hv_fetchs(hash, \"number\", FALSE);
    if (sv_number && SvIOK(*sv_number)) {
        $var.number = SvIV(*sv_number);
    } else {
        croak(\"Missing or invalid 'number' field\");
    }
```

Финальный typemap [2]


```
    SV** sv_string = hv_fetchs(hash, \"string\", FALSE);
    if (sv_string && SvPOK(*sv_string)) {
        $var.string = SvPV_nolen(*sv_string);
    } else {
        croak(\"Missing or invalid 'string' field\");
    }

OUTPUT
T_STRUCT_FOO
    HV* hash = newHV();
    hv_stores(hash, \"number\", newSViv($var.number));
    hv_stores(hash, \"string\", newSVpv($var.string, 0));
    $arg = sv_2mortal(newRV_noinc((SV*)hash));
```

Финальный typemap [3]


```
1 #!/usr/bin/perl
2 use ExtUtils::testlib;
3 use Mytest;
4 Mytest::foo({number => 100500, string => "pwnd\n"});
5
6 use Data::Dumper;
7 warn Dumper Mytest::fooGet();
```

Финальный typemap [3]


```
#!/usr/bin/perl
use ExtUtils::testlib;
use Mytest;
Mytest::foo({number => 100500, string => "pwnd\n"});

use Data::Dumper;
warn Dumper Mytest::fooGet();
```

```
$ perl test
100500 pwnd
$VAR1 = {
          'number' => 100500,
          'string' => 'pwnd'
        };
```

Выводы

TODO

Спасибо за внимание!

Вопросы?

Perl-Conf.Ru/25