

Описание `turетар` для передачи структур в XS

Ступницкий Иван
Инженер, YADRO

Perl-Conf.Ru/25

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

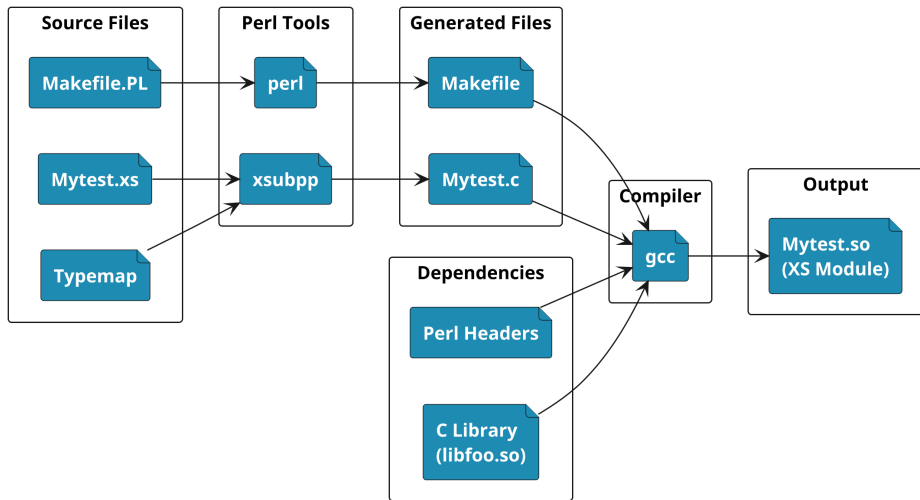
Инженер, YADRO



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

Как переводится название доклада

Perl-Conf.Ru/25



Подопытная библиотека libfoo

```
1 // gcc -shared -fPIC -o libfoo.so foo.c
2 #include <stdio.h>
3 #include "foo.h"
4 void foo() {
5     fprintf(stdout, "japh,\n");
6 }
```

```
1 #ifndef F00
2 #define F00
3 extern void foo();
4 #endif /* F00 */
```

```
1 // gcc -I../foo -o bar bar.c -L../foo -lfoo
2 #include "foo.h"
3 int main() {
4     foo();
5     return 0;
6 }
```

```
1 $ LD_LIBRARY_PATH=$PWD/../foo ./bar
2 japh,
```

Что почитать про XS

- ▶ `perlxsut`
- ▶ `perlxs`
- ▶ `perlguts`
- ▶ `perlapi`
- ▶ `perlxsmap`

```
1 $ h2xs -A -n Mytest
2 Defaulting to backwards compatibility with perl 5.36.3
3 If you intend this module to be compatible with earlier perl
   versions, please
4 specify a minimum perl version with the -b option.
5
6 Writing Mytest/ppport.h
7 Writing Mytest/lib/Mytest.pm
8 Writing Mytest/Mytest.xs
9 Writing Mytest/Makefile.PL
10 Writing Mytest/README
11 Writing Mytest/t/Mytest.t
12 Writing Mytest/Changes
13 Writing Mytest/MANIFEST
```



```
1 $ h2xs -A -n Mytest
2 Defaulting to backwards compatibility with perl 5.36.3
3 If you intend this module to be compatible with earlier perl
   versions, please
4 specify a minimum perl version with the -b option.
5
6 Writing Mytest/ppport.h      # XS compatibility for old Perls
7 Writing Mytest/lib/Mytest.pm # Main module with public interface
8 Writing Mytest/Mytest.xs     # XS code implementing C functions
9 Writing Mytest/Makefile.PL   # Build configuration for the module
10 Writing Mytest/README        # Installation and usage instructions
11 Writing Mytest/t/Mytest.t    # Test suite for module functionality
12 Writing Mytest/Changes       # Module version changelog
13 Writing Mytest/MANIFEST     # List of files in distribution
```

```
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__
```

```
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();
```

```
1  use 5.036003;
2  use ExtUtils::MakeMaker;
3  WriteMakefile(
4      NAME                => 'Mytest',
5      VERSION_FROM        => 'lib/Mytest.pm', # finds $VERSION
6      PREREQ_PM            => {}, # e.g., Module::Name => 1.1
7      ABSTRACT_FROM        => 'lib/Mytest.pm', # finds abstract
8      AUTHOR              => 'i.stup@yadro.com',
9      #LICENSE              => 'perl',
10     #Value must be from legacy list of licenses here
11     #https://metacpan.org/pod/Module::Build::API
12     LIBS                  => [''], # e.g., '-lm'
13     DEFINE                => '', # e.g., '-DHAVE_SOMETHING'
14     INC                   => '-I.', # e.g., '-I. -I/usr/include/other'
15     # Un-comment this if you add C files to link with later:
16     # OBJECT               => '$(0_FILES)', # link all the C files
17 );
```

```
1 use 5.036003;
2 use ExtUtils::MakeMaker;
3 WriteMakefile(
4     NAME                => 'Mytest',
5     VERSION_FROM        => 'lib/Mytest.pm', # finds $VERSION
6     PREREQ_PM           => {}, # e.g., Module::Name => 1.1
7     ABSTRACT_FROM       => 'lib/Mytest.pm', # finds abstract
8     AUTHOR              => 'i.stup@yadro.com',
9     #LICENSE             => 'perl',
10    #Value must be from legacy list of licenses here
11    #https://metacpan.org/pod/Module::Build::API
12    LIBS                 => ['-L../foo -lfoo'],
13    DEFINE               => '', # e.g., '-DHAVE_SOMETHING'
14    INC                  => '-I. -I../foo',
15    # Un-comment this if you add C files to link with later:
16    # OBJECT              => '$(0_FILES)', # link all the C files
17 );
```

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

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

```
1  $ perl Makefile.PL
2  ...
3  $ make
4  ...
5  $ perl test.pl
6  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

```
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 }
```

```
1 #ifndef F00
2 #define F00
3 struct foo {
4     int number;
5     char* string;
6 };
7 extern void foo(struct foo);
8 #endif /* F00 */
```

Вот и всё!

```
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!

```
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);
```

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

Вот и всё!

```
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);
```

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

```
1  $ perl test.pl
2  13 japh,
```

Вот и всё! (нет)

```
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);
```

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

```
1  $ perl test.pl
2  13 japh,
```

В XS нужно проще

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

В XS нужно проще

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

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


В XS нужно проще

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

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

```
1  $ perl Makefile.PL
2  ...
3  $ make
4  ...
5  $ perl test.pl
6  japh,
```

А теперь со структурой

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

А теперь со структурой

```
1  MODULE = xs          PACKAGE = xs
```

```
2
```

```
3  void  
4  foo(struct foo var)
```

```
1  $ make
```

```
2  ...
```

```
3  "/usr/bin/perl" "/usr/lib/perl5/5.36.3/ExtUtils/xsubpp" \  
4      -typemap '/usr/lib/perl5/5.36.3/ExtUtils/typemap' \  
5      Mytest.xs > Mytest.xsc
```

```
6  Could not find a typemap for C type 'struct foo'.
```

```
7  The following C types are mapped by the current typemap:
```

```
8  'AV *', 'Boolean', 'CV *', 'FILE *', 'FileHandle', 'HV *',  
9  'const char *', 'double', 'float', 'int', 'long', 'short',
```

```
10 ...
```

```
11 in Mytest.xs, line 14
```

```
12 make: *** [Makefile:359: Mytest.c] Error 1
```

Что такое typemap

TODO

```
1 struct foo
```

```
T_PACKED
```

```
1 struct foo          T_PACKED

1 struct foo XS_unpack_struct_foo(SV *var) {
2     return (struct foo){666, "it works\n"};
3 }
4
5 MODULE = Mytest      PACKAGE = Mytest
6
7 void
8 foo(struct foo var)
```

```
1 Mytest.c: In function 'XS_xs_foo':
2 Mytest.c:176:31: error: 'XS_unpack_struct' undeclared (first
3 use in this function); did you mean 'XS_unpack_struct_foo'?
4     176 |         struct foo      var = XS_unpack_struct foo(ST(0))
5         |                               ^~~~~~
6         |                               XS_unpack_struct_foo
7 Mytest.c:176:31: note: each undeclared identifier is reported
8 only once for each function it appears in
9 Mytest.c:176:48: error: expected ',' or ';' before 'foo'
10    176 |         struct foo      var = XS_unpack_struct foo(ST(0))
11        |                               ^~~
12 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

```
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)
```

```
1 $ perl test.pl
2 666 it works
```

Как это выглядит в Си

TODO

Избежать вызова функции

```
1 struct foo          T_STRUCT_F00
2
3 INPUT
4 T_STRUCT_F00
5     $var.number = 777;
6     $var.string = \"this too\\n\";
```

Избежать вызова функции

```
1 struct foo          T_STRUCT_F00
2
3 INPUT
4 T_STRUCT_F00
5     $var.number = 777;
6     $var.string = \"this too\\n\";
```

```
1 $ perl test.pl
2 777 this too
```

А теперь без хардкода

```
1 struct foo          T_STRUCT_F00
2
3 INPUT
4 T_STRUCT_F00
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

```
1 struct foo          T_STRUCT_F00
2
3 INPUT
4 T_STRUCT_F00
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));
```

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

А теперь без хардкода

```
1 struct foo          T_STRUCT_F00
2
3 INPUT
4 T_STRUCT_F00
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));
```

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

```
1 $ perl test.pl
2 13 hash
```

TODO

Спасибо за внимание!

Вопросы?

Perl-Conf.Ru/25