

# Описание `turgetar` для передачи структур в XS

Ступницкий Иван  
Инженер, YADRO

Perl-Conf.Ru/25

# О чём расскажу

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

# Ступницкий Иван

Инженер, YADRO



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



ХСВ



```
struct xcb_randr_mode_info_t {  
    uint32_t id;  
    uint16_t width;  
    uint16_t height;  
    uint32_t dot_clock;  
    uint16_t hsync_start;  
    uint16_t hsync_end;  
    uint16_t htotal;  
    uint16_t hskew;  
    uint16_t vsync_start;  
    uint16_t vsync_end;  
    uint16_t vtotal;  
    uint16_t name_len;  
    uint32_t mode_flags;  
};
```

```
xcb_randr_create_mode_cookie_t xcb_randr_create_mode(  
    xcb_connection_t *conn,  
    xcb_window_t window,  
    struct xcb_randr_mode_info_t mode_info, // <----- PROBLEM  
    uint32_t name_len,  
    const char *name  
);
```

```
# TODO RandR typemap of
# # mode_info_t,
# # transform_t,
# # monitor_info_t not implemented
if (index $path, "randr") {
    my $randr_exclude = join "|", qw(
        randr_create_mode
        randr_set_monitor
        randr_set_crtc_transform );
    @request = grep ! /^HV \*\s+$randr_exclude\b/, @request;
}
```

# Что почитать про XS

- ▶ `perlxstut`
- ▶ `perlxs`
- ▶ `perlguts`
- ▶ `perlapi`
- ▶ `perlxstypemap`

```
$ 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
```

```
$ 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

```
package Mytest;
use 5.036003;
use strict;
use warnings;
require Exporter;

our @ISA = qw(Exporter);
our %EXPORT_TAGS = ( 'all' => [ qw() ] );
our @EXPORT_OK = ( @{ $EXPORT_TAGS{'all'} } );
our @EXPORT = qw();
our $VERSION = '0.01';

require XSLoader;
XSLoader::load('Mytest', $VERSION);

1;
__END__
```

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

#include "ppport.h"
```

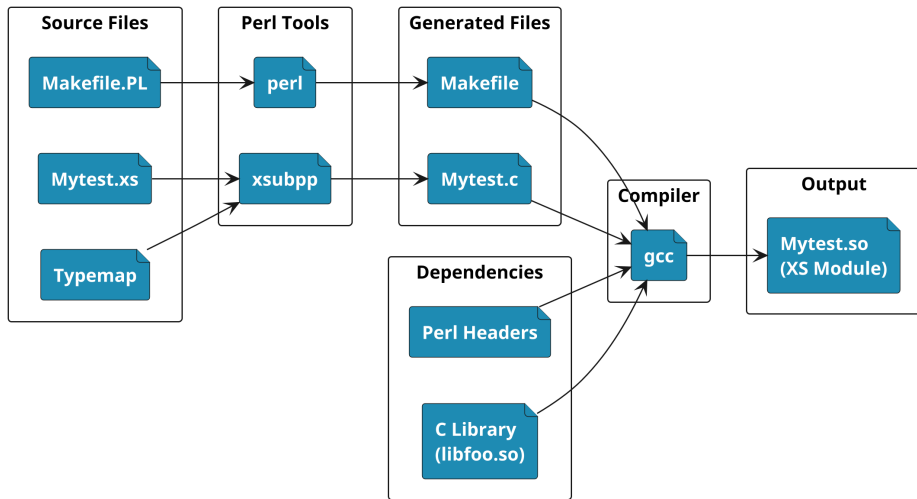
```
MODULE = Mytest      PACKAGE = Mytest
```

# Makefile.PL

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

# Как переводится название доклада

Perl-Conf.Ru/25



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

#ifdef F00
#define F00
extern void foo();
#endif /* F00 */
```

```
// 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,
```

# Вызов libfoo из XS

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

#include "ppport.h"

#include "foo.h"
```

```
MODULE = Mytest      PACKAGE = Mytest
```

```
void
xs()
    CODE:
        foo();
```

# Makefile.PL

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

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

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

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

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

# Структура в libfoo

```
// 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);
}
```

# Структура в libfoo

```
// 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);
}
```

```
#ifndef F00
#define F00
struct foo {
    int number;
    char* string;
};
extern void foo(struct foo);
#endif /* F00 */
```

# Вот и всё!

```
MODULE = Mytest      PACKAGE = Mytest
```

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

# 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");
```

# 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

**MODULE** = Mytest      **PACKAGE** = Mytest

```
void  
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

**MODULE** = Mytest      **PACKAGE** = Mytest

```
void  
foo(struct foo var)
```

# А теперь со структурой

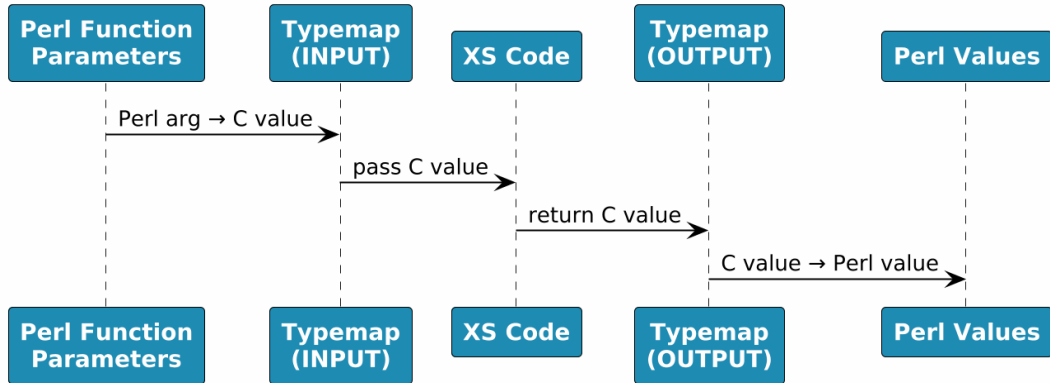
**MODULE** = Mytest      **PACKAGE** = Mytest

```
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



# Что такое typemap

# Что такое typemap

**INPUT**

T\_IV

```
$var = ($type)SvIV($arg)
```

# Что такое typemap

## INPUT

```
T_IV  
    $var = ($type)SvIV($arg)
```

## OUTPUT

```
T_IV  
    sv_setiv($arg, (IV)$var);
```

# Что такое typemap

```
# ExtUtils/typemap
```

## TYPEMAP

```
int      T_IV  
long     T_IV  
short    T_IV
```

## INPUT

```
T_IV  
    $var = ($type)SvIV($arg)
```

## OUTPUT

```
T_IV  
    sv_setiv($arg, (IV)$var);
```

# T\_PACKED

```
struct foo
```

```
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)
```

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

```
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)
```

```
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

```
XS_EUPXS(XS_Mytest_foo); /* prototype to pass -Wmissing-prototypes
    */
XS_EUPXS(XS_Mytest_foo)
{
    dVAR; dXSARGS;
    if (items != 1)
        croak_xs_usage(cv, "var");
    {
        struct foo    var = XS_unpack_struct_foo(ST(0))
;

        foo(var);
    }
    XSRETURN_EMPTY;
}
```

# Избежать вызова функции

```
struct foo          T_STRUCT_F00
```

## INPUT

```
T_STRUCT_F00
    $var.number = 777;
    $var.string = \"this too\\n\";
```

# Избежать вызова функции

```
struct foo          T_STRUCT_F00
```

## INPUT

```
T_STRUCT_F00
    $var.number = 777;
    $var.string = \"this too\\n\";
```

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

# Как это выглядит в Си

```
XS_EUPXS(XS_Mytest_foo); /* prototype to pass -Wmissing-prototypes
*/
XS_EUPXS(XS_Mytest_foo)
{
    dVAR; dXSARGS;
    if (items != 1)
        croak_xs_usage(cv, "var");
    {
        struct foo var;

        var.number = 777;
        var.string = "this too\n"
;

        foo(var);
    }
    XSRETURN_EMPTY;
}
```

# Передача структуры C -> Perl [1]

```
...  
struct foo fooGet(void) {  
    struct foo v;  
    v.number = 42;  
    v.string = "Hi from C";  
    return v;  
}
```

# Передача структуры C -> Perl [1]

```
...  
struct foo fooGet(void) {  
    struct foo v;  
    v.number = 42;  
    v.string = "Hi from C";  
    return v;  
}
```

```
...  
extern struct foo fooGet(void);
```

# Передача структуры C -> Perl [1]

```
...  
struct foo fooGet(void) {  
    struct foo v;  
    v.number = 42;  
    v.string = "Hi from C";  
    return v;  
}
```

```
...  
extern struct foo fooGet(void);
```

```
...  
struct foo  
fooGet();
```

## OUTPUT

T\_STRUCT\_F00

```
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));
```

# Передача структуры C -> Perl [3]

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

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

# Передача структуры C -> Perl [3]

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

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

$perl test
$VAR1 = {
    'string' => 'Hi from C',
    'number' => 42
};
```

```
...
dVAR; dXSARGS;
if (items != 0)
    croak_xs_usage(cv, "");
{
    struct foo RETVAL;
    RETVAL = fooGet();
    {
        SV * RETVALSV;
        RETVALSV = sv_newmortal();
        HV* hash = newHV();
        hv_stores(hash, "number", newSViv(RETVAL.number));
        hv_stores(hash, "string", newSVpv(RETVAL.string, 0));
        RETVALSV = sv_2mortal(newRV_noinc((SV*)hash));
        ST(0) = RETVALSV;
    }
}
XSRETURN(1);
```

# А теперь без хардкода

```
struct foo          T_STRUCT_F00
```

## INPUT

```
T_STRUCT_F00
    $var.number = SvIV(*hv_fetchs((HV*)SvRV($arg),
        \"number\", FALSE));
    $var.string = SvPV_nolen(*hv_fetchs((HV*)SvRV($arg),
        \"string\", FALSE));
```

# А теперь без хардкода

```
struct foo          T_STRUCT_F00
```

## INPUT

```
T_STRUCT_F00
```

```
    $var.number = SvIV(*hv_fetchs((HV*)SvRV($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_F00
```

## INPUT

```
T_STRUCT_F00
```

```
    $var.number = SvIV(*hv_fetchs((HV*)SvRV($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_F00
```

## INPUT

```
T_STRUCT_F00
    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");
    }
}
```

```
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\_F00

```
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]

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

use Data::Dumper;
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' => 42,
    'string' => 'Hi from C'
};
```

В простом случае можно описывать код в функциях на Си и использовать `T_PACKED_PATCHED` => компилятор сделает работу за вас :)

При работе со сложными структурами или для явного контроля трансляции предпочтительнее написать `typedef`

Спасибо за внимание!

Вопросы?

Perl-Conf.Ru/25