

# Особенности создания XS-модулей на языке C++

Владимир Тимофеев

2013

- 1 Введение
- 2 Пример
- 3 Как найти компилятор
- 4 Конфликты имен
- 5 Исключения
- 6 Заклучение

- 1 Введение
  - Почему XS
  - Почему C++

2 Пример

3 Как найти компилятор

4 Конфликты имен

5 Исключения

6 Заключение

# Почему XS?

Потому что мы хотим:

- Увеличить производительность

# Почему XS?

Потому что мы хотим:

- Увеличить производительность
- Уменьшить потребление памяти

# Почему XS?

Потому что мы хотим:

- Увеличить производительность
- Уменьшить потребление памяти
- Использовать сторонние компоненты

# Почему C++ ?

XS – кодогенератор для C. Но C++ ...

- Похож на C

# Почему C++ ?

XS – кодогенератор для C. Но C++ ...

- Похож на C
- Сравним по скорости и потреблению памяти



# Почему C++ ?

XS – кодогенератор для C. Но C++ ...

- Похож на C
- Сравним по скорости и потреблению памяти
- + Инкапсуляция

# Почему C++ ?

XS – кодогенератор для C. Но C++ ...

- Похож на C
- Сравним по скорости и потреблению памяти
- + Инкапсуляция
- + STL, Boost, ...

## 1 Введение

## 2 Пример

- Обзор файлов

## 3 Как найти компилятор

## 4 Конфликты имен

## 5 Исключения

## 6 Заключение

# Пример

☛ Makefile.PL | CppStat.pm | CppStat.xs | typemap | perlobject.map

```
1 use ExtUtils::MakeMaker;
3 my $CC = 'g++';
5 WriteMakefile(
      NAME          => 'CppStat',
7     VERSION_FROM  => 'CppStat.pm',
      TYPEMAPS      => ['perlobject.map'],
9     CC => $CC,
      LD => '$(CC)',
11 );
```

# Пример

Makefile.PL | 🖱️ CppStat.pm | CppStat.xs | typemap | perlobject.map

```
package CppStat;  
2 use strict;  
  use warnings;  
4  
  our $VERSION = '0.01';  
6  
  require XSLoader;  
8 XSLoader::load('CppStat', $VERSION);  
10 1;
```

# Пример

Makefile.PL

CppStat.pm

🖱️ CppStat.xs

typemap

perlobject.map

```
#ifdef __cplusplus
2 extern "C" {
#endif
4
6 #include "EXTERN.h"
7 #include "perl.h"
8
8 #ifdef __cplusplus
9 }
10 #endif
12 #include "XSUB.h"
14 #include <numeric>
15 #include <vector>
```

# Пример

Makefile.PL | CppStat.pm | 🖱️ CppStat.xs | typemap | perlobject.map

```
class CppStat {  
18 public:  
    CppStat() : values() { }  
20    ~CppStat() { }  
  
22    void add(double val) {  
        values.push_back(val);  
24    }  
    double avg() const {  
26        return std::accumulate(  
            values.begin(), values.end(), 0.0  
28            ) / values.size();  
    }  
30 private:  
    std::vector<double> values;  
32 };
```

# Пример

Makefile.PL | CppStat.pm | 🖱️ CppStat.xs | typemap | perlobject.map

```
34 MODULE = CppStat          PACKAGE = CppStat
36 CppStat*
  CppStat::new();
38
  void
40 CppStat::DESTROY();
42
  void
  CppStat::add(double val);
44
  double
46 CppStat::avg();
```



# Пример

Makefile.PL

CppStat.pm

CppStat.xs

● typemap

perlobject.map

TYPEMAP

2 CppStat \* O\_OBJECT

# Пример

Makefile.PL | CppStat.pm | CppStat.xs | typemap | 🖱️ perlobject.map

- Можно поискать в интернете
- или сгенерировать с помощью модуля ExtUtils::Typemap::ObjectMap на CPAN
- или почитать perldoc perlxs

## 1 Введение

## 2 Пример

## 3 Как найти компилятор

- CC = g++
- -x c++
- ExtUtils::CppGuess
- Module::Build::WithXSp
- Статистика

## 4 Конфликты имен

## 5 Исключения

## 6 Заключение

# CC = g++

```
2 use ExtUtils::MakeMaker;
4 my $CC = 'g++';
6 WriteMakefile(
8     NAME          => 'CppStat',
9     VERSION_FROM  => 'CppStat.pm',
10    TYPEMAPS       => ['perlobject.map'],
11    CC => $CC,
12    LD => '$(CC)',
13 );
```

- Только GCC
- Все файлы как C++

# -X C++

```
1 use ExtUtils::MakeMaker;  
2 use Config;  
3  
4 my $CCFLAGS = join(' ', $Config{ccflags}, '-x c++');  
5  
6 WriteMakefile(  
7     NAME          => 'CppStat',  
8     VERSION_FROM  => 'CppStat.pm',  
9     TYPEMAPS      => ['perlobject.map'],  
10    LIBS          => ['-lstdc++'],  
11    CCFLAGS        => $CCFLAGS,  
12 );
```

- Компилятор тот же, что для perl
- Компилятор должен понимать опцию -x (GCC, Clang)
- Все файлы как C++

# ExtUtils::CppGuess

```
use ExtUtils::MakeMaker;  
2 use ExtUtils::CppGuess;  
  
4 my $guess = ExtUtils::CppGuess->new;  
  
6 WriteMakefile(  
    NAME          => 'CppStat',  
8    VERSION_FROM => 'CppStat.pm',  
    TYPEMAPS      => ['perlobject.map'],  
10   CONFIGURE_REQUIRES => {  
        'ExtUtils::CppGuess' => 0,  
12   },  
    $guess->makemaker_options,  
14 );
```

- Компилятор тот же, что для perl
- GCC, Clang, MSVC
- Все файлы как C++

# Module::Build::WithXSp

Использует ExtUtils::CppGuess, но...

- XS++
- Module::Build

# Статистика

- Всего на CPAN:  $\sim 120$
- CC = g++:  $\sim 50$
- ExtUtils::CppGuess: 13
- Module::Build:  $\sim 15$
- Module::Build::WithXSpp: 6



- 1 Введение
- 2 Пример
- 3 Как найти компилятор
- 4 Конфликты имен**
  - `iostream`
- 5 Исключения
- 6 Заключение

# Конфликты имен

```
1  #ifdef  __cplusplus
2  extern  "C" {
3  #endif
4
5  #include  "EXTERN.h"
6  #include  "perl.h"
7
8  #ifdef  __cplusplus
9  }
10 #endif
11
12 #include  "XSUB.h"
13
14 #undef  do_open
15 #undef  do_close
16
17 #include  <iostream>
```

## 6 Заключение

# Просто и неправильно

MyClass.cc

```
1 void MyClass::method(int arg) {  
    std::vector<int> tmp;  
3     if (arg < 0) croak("MyClass::method: arg should  
        be positive");  
    // ...  
5 }
```

# Просто и неправильно

MyClass.cc

```
1 void MyClass::method(int arg) {  
    std::vector<int> tmp;  
3     if (arg < 0) croak("MyClass::method: arg should  
        be positive");  
    // ...  
5 }
```

- Утечка памяти

# Просто и неправильно

MyClass.cc

```
1 void MyClass::method(int arg) {  
    std::vector<int> tmp;  
3     if (arg < 0) croak("MyClass::method: arg should  
        be positive");  
    // ...  
5 }
```

- Утечка памяти
- Деструктор для tmp вызван не будет

# Просто и неправильно

MyClass.cc

```
1 void MyClass::method(int arg) {  
    std::vector<int> tmp;  
3     if (arg < 0) croak("MyClass::method: arg should  
        be positive");  
    // ...  
5 }
```

- Утечка памяти
- Деструктор для tmp вызван не будет
- Потому что croak реализован через longjmp

# Переходим к C++ исключениям

MyClass.cc

```
1 #include <stdexcept>
void MyClass::method(int arg) {
3     std::vector<int> tmp;
    if (arg < 0) throw std::runtime_error(
5         "MyClass::method: arg should be positive"
        );
7     // ...
}
```

- Программа упадет
- Исключение надо еще перехватить



# Переходим к C++ исключениям

MyModule.xs

`void`

2 `MyClass::method(int arg);`

Придется переписать и обернуть в try...catch.

# Переходим к C++ исключениям

MyModule.xs

```
void
```

```
2 MyClass::method(int arg);
```

Придется переписать и обернуть в try...catch.

```
void
```

```
2 MyClass::method(int arg)
```

```
CODE:
```

```
4     try {
        THIS->method(arg);
6     }
    catch (const std::exception& e) {
8         croak("Ops: %s", e.what());
    }
10    catch (...) {
        croak("Ops: unknown c++ exception");
12    }
```

# Переходим к C++ исключениям

```
void
2 MyClass::method(int arg)
  CODE:
4   try {
        THIS->method(arg);
6   }
    catch (const std::exception& e) {
8       croak("Ops: %s", e.what());
    }
10  catch (...) {
        croak("Ops: unknown c++ exception");
12  }
```

# Переходим к C++ исключениям

```
void
2 MyClass::method(int arg)
  CODE:
4   try {
        THIS->method(arg);
6   }
    catch (const std::exception& e) {
8        croak("Ops: %s", e.what());
    }
10   catch (...) {
        croak("Ops: unknown c++ exception");
12   }
```

- И получим опять утечку памяти (хоть и мелкую)

# Переходим к C++ исключениям

```
void
2 MyClass::method(int arg)
  CODE:
4     try {
        THIS->method(arg);
6     }
    catch (const std::exception& e) {
8         croak("Ops: %s", e.what());
    }
10    catch (...) {
        croak("Ops: unknown c++ exception");
12    }
```

- И получим опять утечку памяти (хоть и мелкую)
- Именно в такой код наши вызовы оборачивает ExtUtils::XSpp

# Вот так не течет

```
void
2 MyClass::method(int arg)
  CODE:
4   char errmsg[1024];
   errmsg[0] = '\0';
6   try {
       THIS->method(arg);
8   }
   catch (const std::exception& e) {
10      snprintf(errmsg, 1024, "Ops: %s", e.what())
        ;
   }
12  catch (...) {
      snprintf(errmsg, 1024, "Ops: unknown c++
        exception");
14  }
   if (errmsg[0]) {
16      croak("%s", errmsg);
   }
```

- A set of small navigation icons typically found in Beamer presentations, including symbols for back, forward, search, and other slide controls.

# Хотелось бы писать Makefile.PL так

```
1 use ExtUtils::MakeMaker;  
3 WriteMakefile(  
    NAME          => 'CppStat',  
5    VERSION_FROM => 'CppStat.pm',  
    TYPEMAPS      => ['perlobject.map'],  
7    XS_LANG      => 'c++',  
    );
```



# Может быть XS может выглядеть так?

```
void  
2 MyClass::method(int arg)  
  CPPCATCH: std
```

# Итого

- Писать на C++ можно

# Итого

- Писать на C++ можно
- Есть поле для улучшений

# Итого

- Писать на C++ можно
- Есть поле для улучшений
- Вопросы?

# Итого

- Писать на C++ можно
- Есть поле для улучшений
- Вопросы?
- <mailto:vovkasm@gmail.com>
- Эта презентация доступна на [github.com/vovkasm/using-cpp-in-perlxs-slides](https://github.com/vovkasm/using-cpp-in-perlxs-slides)