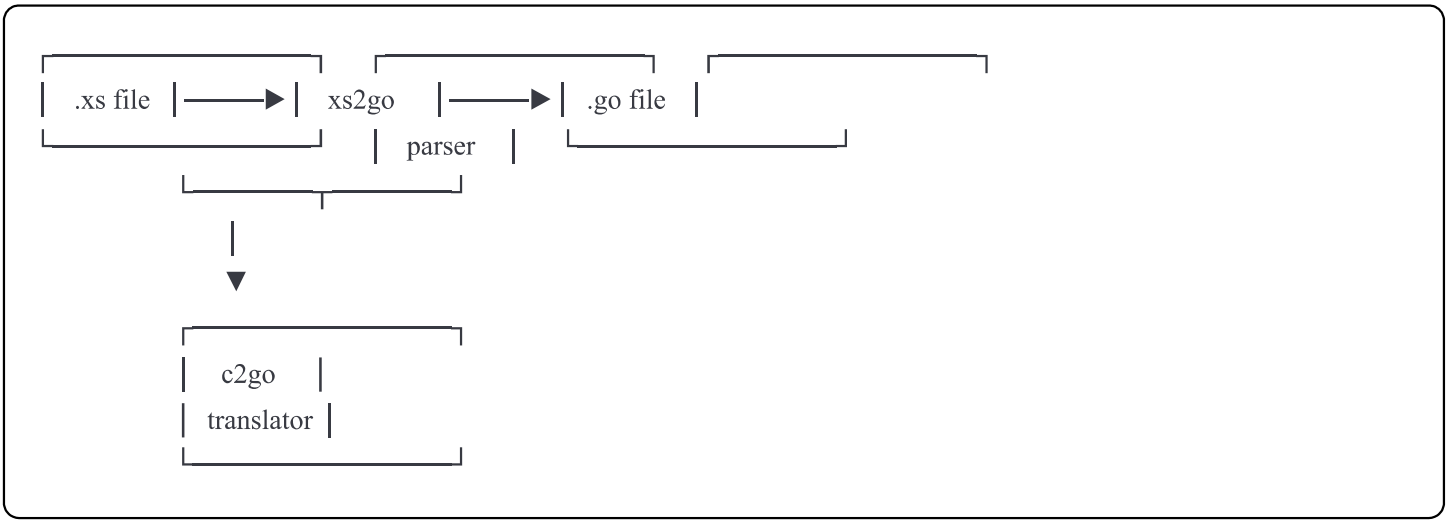
XS → Go Translation System

Статус: В разработке (MVP готов)

Обзор

Система для автоматической трансляции Perl XS модулей в Go код. XS модули — это C расширения для Perl, которые используют Perl C API для создания высокопроизводительных функций.

Архитектура



Компоненты

1. XS Parser (`pkg/xs2go/`)

Парсит структуру XS файлов:

- MODULE/PACKAGE декларации
- XS функции (PPCODE: и CODE:/OUTPUT: секции)
- Аргументы с типами
- Поддержка простых и сложных XS файлов

Файлы:

- `pkg/xs2go/translator.go` — основной транслятор

2. C → Go Translator (`pkg/c2go/`)

Транслирует C код внутри XS функций в Go:

- Объявления переменных

- Присваивания
- Условия (if/else)
- Циклы (for/while)
- Switch/case
- Вызовы функций
- Perl C API → Go runtime маппинг

Файлы:

- `pkg/c2go/translator.go` — транслятор C в Go

3. CLI утилита (`cmd/xs2go/`)

```
bash

go run ./cmd/xs2go path/to/module.xs > output.go
```

Что работает

Простые XS модули

```
c

// test_xs/Test.xs
int add(a, b)
    int a
    int b
    CODE:
        RETVAL = a + b;
    OUTPUT:
        RETVAL

SV *
hello(name)
    SV *name
    CODE:
        char *str = SvPV_nolen(name);
        RETVAL = newSVpvf("Hello, %s!", str);
    OUTPUT:
        RETVAL
```

Транслируется в:

```go
func perl_Test_XS_add(args ...*SV) *SV {
    a := args[0].AsInt()
    b := args[1].AsInt()
    var RETVAL *SV
    RETVAL = svInt(int64(a + b))
    return RETVAL
}

func perl_Test_XS_hello(args ...*SV) *SV {
    name := args[0]
    var RETVAL *SV
    str := name.AsString()
    RETVAL = svStr(fmt.Sprintf("Hello, %s!", str))
    return RETVAL
}
```

Сложные XS модули (частично)

JSON::XS — функции извлекаются, но внутренний C код требует доработки.

Perl C API → Go Mapping

Создание значений

Perl C API	Go Runtime	Статус
`newSViv(i)`	`svInt(int64(i))`	✅
`newSVuv(u)`	`svInt(int64(u))`	✅
`newSVnv(n)`	`svFloat(n)`	✅
`newSVpv(s, len)`	`svStr(s)`	✅
`newSVpvn(s, len)`	`svStr(s)`	✅
`newSVpvf(fmt, ...)`	`svStr(fmt.Sprintf(...))`	✅
`newSVsv(sv)`	`svCopy(sv)`	✅
`newRV_inc(sv)`	`svRef(sv)`	✅
`newHV()`	`svHash()`	✅

Perl C API	Go Runtime	Статус
`newAV()`	`svArray()`	✅

Доступ к значениям

Perl C API	Go Runtime	Статус
`SvIV(sv)`	`sv.AsInt()`	✅
`SvUV(sv)`	`uint64(sv.AsInt())`	✅
`SvNV(sv)`	`sv.AsFloat()`	✅
`SvPV_nolen(sv)`	`sv.AsString()`	✅
`SvPV(sv, len)`	`sv.AsString()`	✅
`SvTRUE(sv)`	`sv.IsTrue()`	✅
`SvOK(sv)`	`!sv.IsUndef()`	✅
`SvROK(sv)`	`sv.IsRef()`	✅
`SvRV(sv)`	`sv.Deref()`	✅
`SvCUR(sv)`	`len(sv.AsString())`	✅

Хеш операции

Perl C API	Go Runtime	Статус
`hv_store(hv, k, l, v, h)`	`svHSet(hv, k, v)`	✅
`hv_fetch(hv, k, l, lv)`	`svHGet(hv, k)`	✅
`hv_exists(hv, k, l)`	`svHExists(hv, k)`	✅
`hv_delete(hv, k, l, f)`	`svHDelete(hv, k)`	✅

Массив операции

Perl C API	Go Runtime	Статус
`av_push(av, sv)`	`svPush(av, sv)`	✅
`av_pop(av)`	`svPop(av)`	✅
`av_shift(av)`	`svShift(av)`	✅
`av_len(av)`	`len(av.av)-1`	✅
`av_fetch(av, i, lv)`	`svAGet(av, i)`	✅

Прочее

Perl C API	Go Runtime	Статус
`croak(msg)`	`panic(msg)`	✅
`warn(msg)`	`fmt.Fprintf(os.Stderr, msg)`	✅
`NULL`	`nil`	✅
`&PL_sv_undef`	`svUndef()`	✅

TODO

Высокий приоритет

- [ ] Улучшить обработку арифметических выражений в RETVAL
- [ ] Поддержка структур (self->field)
- [ ] Интеграция с ccgo для сложного C кода

Средний приоритет

- [ ] XS ALIAS поддержка
- [ ] Memory management (SvREFCNT_inc/dec)
- [ ] Обработка указателей
- [ ] Вложенные структуры

Низкий приоритет

- [ ] Callback функции

- [ ] XS OVERLOAD
- [ ] Threads (CLONE)
- [ ] Полная JSON::XS трансляция

Альтернативный подход: ccgo + наш маппинг

Для очень сложных XS модулей можно использовать ccgo (C → Go транслятор от modernc.org):

```bash
go install modernc.org/ccgo/v4@latest
```

Workflow:

1. ccgo транслирует чистый C код в Go
2. Наш c2go заменяет Perl API вызовы на Go runtime
3. Результат компилируется с нашим SV runtime

Проблема: ccgo не знает Perl API (SV*, newSViv и т.д.), поэтому нужен наш маппинг поверх.

Рекомендации по использованию

Тип модуля	Рекомендация
Простые XS (арифметика, строки)	xs2go напрямую
Средние XS (хеши, массивы)	xs2go + ручная доработка
Сложные XS (JSON::XS, DBI)	Реализация на Go с stdlib

Замены сложных модулей на Go stdlib

XS модуль	Go альтернатива
JSON::XS	`encoding/json`
DBI	`database/sql`
Compress::Zlib	`compress/gzip`
Digest::MD5	`crypto/md5`
Digest::SHA	`crypto/sha256`

XS модуль	Go альтернатива
Socket	`net`
MIME::Base64	`encoding/base64`

Тестирование

```bash
# Простой тест
go run ./cmd/xs2go test_xs/Test.xs

# Сохранить в файл
go run ./cmd/xs2go test_xs/Test.xs > tmp/output.go

# Проверить компиляцию
go build tmp/output.go

# Сложный модуль
go run ./cmd/xs2go JSON-XS-4.03/XS.xs > tmp/json_xs.go
```

Примеры вывода

test_xs/Test.xs

```bash
$ go run ./cmd/xs2go test_xs/Test.xs
```

```go
```

```go
// Auto-generated from XS
// Module: Test::XS, Package: Test::XS

package main

import (
    "fmt"
    "strings"
)

var _ = fmt.Sprint
var _ = strings.Cut

func perl_Test_XS_add(args ...*SV) *SV {
    a := args[0].AsInt()
    b := args[1].AsInt()
    var RETVAL *SV
    RETVAL = svInt(int64(a + b))
    return RETVAL
}

func perl_Test_XS_hello(args ...*SV) *SV {
    name := args[0]
    var RETVAL *SV
    str := name.AsString()
    RETVAL = svStr(fmt.Sprintf("Hello, %s!", str))
    return RETVAL
}

func init() {
    perl_register_sub("Test::XS::add", perl_Test_XS_add)
    perl_register_sub("Test::XS::hello", perl_Test_XS_hello)
}
```

История изменений

- **v0.1** — Базовый XS парсер, простая C → Go трансляция
- **v0.2** — Добавлен модуль c2go, расширен Perl API маппинг
- **v0.3** — Поддержка PPCODE и CODE/OUTPUT секций