Паттерны проектирования в языке Julia

Клюев А.С.

Можно писать быстрый код

```
julia> function func(x)
           return x^10
       end
func (generic function with 1 method)

julia> @time func(x) setup=(x=rand())
0.7559434573817069: 0.000013 seconds (2 allocations: 32 bytes)
0.9724089138351235

julia> function func_c(x::Float64)::Float64
           return x^10
       end
func_c (generic function with 1 method)

julia> @time func_c(x) setup=(x=rand())
2.7644656780139533e-9: 0.000002 seconds (2 allocations: 32 bytes)
0.139367375032855
```

Встроенные инструменты профилирования

```
julia> using BenchmarkTools

julia> function func(x)
           return x^10
       end
func (generic function with 1 method)

julia> @btime func(x) setup=(x=rand())
  6.035 ns (0 allocations: 0 bytes)
2.193286329771389 4e-14

julia> using BenchmarkTools

julia> @benchmark sort(data) setup=(data=rand(10))
BenchmarkTools.Trial: 10000 samples with 985 evaluations.
 Range (min … max):  51.597 ns … 775.909 ns  ┊ GC (min … max): 0.00% … 0.00%
 Time  (median):     61.203 ns               ┊ GC (median):    0.00%
 Time  (mean ± σ):   68.556 ns ±  30.600 ns  ┊ GC (mean ± σ):  2.02% ± 4.78%

  51.6 ns         Histogram: frequency by time          132 ns <

 Memory estimate: 144 bytes, allocs estimate: 1.
```

Особенности синтаксиса

```
julia> var = input < 0 ? -10 : 10
10

julia> input < 0 && 10
false

julia> input < 0 || -10
-10

julia> @generated function bar(x)
           if x <: Integer
               return :(x ^ 2)
           else
               return :(x)
           end
       end
bar (generic function with 1 method)

julia> @assert 1==1 "string"
```

```
julia> a = [1 2 3 4]
1×4 Matrix{Int64}:
 1  2  3  4

julia> a |> first |> typeof |> string |> last |> Int64
52

julia> a |> first |> typeof |> string |> last
'4': ASCII/Unicode U+0034 (category Nd: Number, decimal digit)
```

Возможность заглянуть под капот

```
julia>        is_cond(flags) = return flags & COND != 0
is_cond (generic function with 1 method)

julia> macro IS_COND(flags)
           return :($(esc(flags)) & COND != 0)
       end
@IS_COND (macro with 1 method)

julia> test_func(x) = is_cond(x); test_macro(x) = @IS_COND(x)
test_macro (generic function with 1 method)

julia> @code_lowered test_func(UInt(1))
CodeInfo(
1 — %1 = Main.is_cond(x)
└──      return %1
)

julia> @code_lowered test_macro(UInt(1))
CodeInfo(
1 — %1 = x & Main.COND
│   %2 = %1 != 0
└──      return %2
)
```

```
julia> @code_typed test_func(UInt(1))
CodeInfo(
1 — %1 = Main.COND::UInt64
│   %2 = Base.and_int(x, %1)::UInt64
│   %3 = (%2 === 0x0000000000000000)::Bool
│   %4 = Base.and_int(true, %3)::Bool
│   %5 = Base.not_int(%4)::Bool
└──      return %5
) => Bool

julia> @code_typed test_macro(UInt(1))
CodeInfo(
1 — %1 = Main.COND::UInt64
│   %2 = Base.and_int(x, %1)::UInt64
│   %3 = (%2 === 0x0000000000000000)::Bool
│   %4 = Base.and_int(true, %3)::Bool
│   %5 = Base.not_int(%4)::Bool
└──      return %5
) => Bool

julia> @code_llvm test_func(UInt(1))
;  @ REPL[18]:1 within `test_func`
define i8 @julia_test_func_1156(i64 zeroext %0) #0 {
top:
; ┌ @ REPL[16]:1 within `is_cond`
; │┌ @ operators.jl:282 within `!=`
; ││┌ @ int.jl:485 within `==` @ promotion.jl:477
    %1 = trunc i64 %0 to i8
; ││└
; ││┌ @ bool.jl:35 within `!`
    %2 = lshr i8 %1, 7
; └└└
  ret i8 %2
}

julia> @code_llvm test_macro(UInt(1))
;  @ REPL[18]:1 within `test_macro`
define i8 @julia_test_macro_1167(i64 zeroext %0) #0 {
top:
; ┌ @ operators.jl:282 within `!=`
; │┌ @ int.jl:485 within `==` @ promotion.jl:477
    %1 = trunc i64 %0 to i8
; │└
; │┌ @ bool.jl:35 within `!`
    %2 = lshr i8 %1, 7
; └└
  ret i8 %2
}
```

Типы

- Система типов динамическая, но есть возможность указать тип
- Добавление аннотации типов позволяет использовать множественную диспетчеризацию
- Нет наследования от конкретного типа
- Типы имеют значения, не переменные
- Абстрактные и конкретные типы могут быть параметризованны типами

```
help?> DataType
search: DataType

  DataType <: Type{T}

  DataType represents explicitly declared types that have
  names, explicitly declared supertypes, and, optionally,
  parameters. Every concrete value in the system is an instance
  of some DataType.

  Examples
  ==========

  julia> typeof(Real)
  DataType

  julia> typeof(Int)
  DataType

  julia> struct Point
             x::Int
             y
         end

  julia> typeof(Point)
  DataType

help?> Any
search: Any any any! code_warntype ReadOnlyMemoryError

  Any::DataType

  Any is the union of all types. It has the defining property
  isa(x, Any) == true for any x. Any therefore describes the
  entire universe of possible values. For example Integer is a
  subset of Any that includes Int, Int8, and other integer
  types.
```

Паттерны переиспользования (Reusability patterns)

- Делегация (Delegation)
- Holy traits
- Параметрический тип

```
julia> abstract type MyAbType end

julia> struct MyNewStruct{T} <: MyAbType
           field::Float64
           function MyStruct(x::T) where {T}
               new{T}(x)
           end
       end
```

```
julia> struct DelegationStruct
           my_struct::DataType
           DelegationStruct(
               ms::DataType
           ) = new(ms)
       end

julia> struct MyStruct
           field::Float64
           function MyStruct(x::Float64)
               new(x)
           end
       end

julia> MyStruct |> typeof
DataType

julia> struct DelegationStruct
           my_struct::DataType
           DelegationStruct(
               ms::DataType
           ) = new(ms)
       end
```

Паттерны переиспользования (Reusability patterns)

- Делегация (Delegation)
- Holy traits
- Параметрический тип

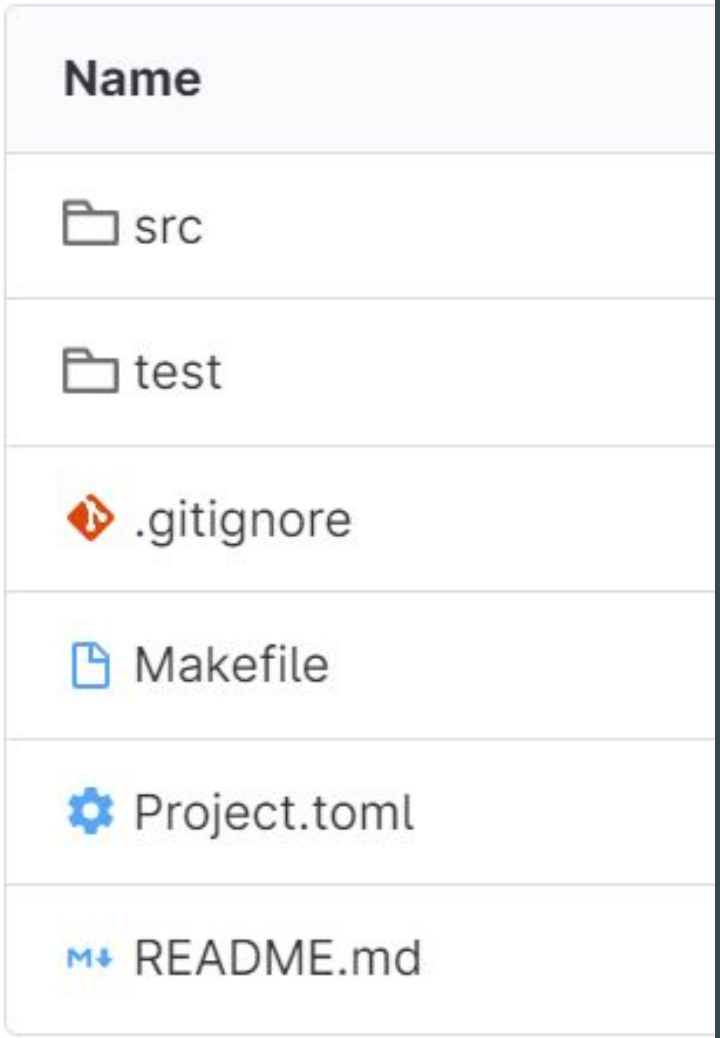
```
1  abstract type IsSweetTooth end
2
3  struct SweetTooth <: IsSweetTooth end
4  struct NotSweetTooth <: IsSweetTooth end
5
6  IsSweetTooth(::Type) == SweetTooth()
7  IsSweetTooth(::Type{<:ParentOne}) == NotSweetTooth()
8  IsSweetTooth(::Type{ChildThree}) == NotSweetTooth()
9
10 want_some_candy(child::T) where {T} = want_some_candy(IsSweetTooth(T), child)
11 want_some_candy(::NotSweetTooth, child) = false
12 want_some_candy(::SweetTooth, child) = true
```

Паттерны производительности

- Использование глобальных констант
- Структуры с массивами
- Общий массив
- Мемоизаторы (Memoization)
- Барьерная синхронизация

Паттерны для гибкости

- Singleton type dispatch
- Stub/Mock
- Functional pipes pattern



Анти-паттерны

- Пиратство (Piracy)
- Ограниченные аргументы (Narrow argument types anti-pattern)
- Nonconcrete field types anti-pattern

```
julia> "a" + 1
ERROR: MethodError: no method matching +(::String, ::Int64)
Closest candidates are:
  +(::Any, ::Any, ::Any, ::Any...) at operators.jl:591
  +(::T, ::T) where T<:Union{Int128, Int16, Int32, Int64, Int8, UInt128, UInt16, UInt32, U
  +(::LinearAlgebra.UniformScaling, ::Number) at /opt/julia-1.8.1/share/julia/stdlib/v1.8/
  ...
Stacktrace:
 [1] top-level scope
   @ REPL[1]:1

julia> module MySum
       import Base.+
       (+)(str::AbstractString, num::Number) = "$str-$num"
       end # end of MySum
Main.MySum

julia> "a" + 1
"a-1"
```

```
julia> Base.:(+)(x::Int64, y::Int64) = "string"

fatal: error thrown and no exception handler available.
MethodError(f=Int64, args=("string",), world=0x0000000000007eb4)
jl_method_error_bare at /cache/build/default-amdci5-0/julialang/j
ulia-release-1-dot-8/src/gf.c:1879
jl_method_error at /cache/build/default-amdci5-0/julialang/julia-
release-1-dot-8/src/gf.c:1897
jl_lookup_generic_ at /cache/build/default-amdci5-0/julialang/jul
ia-release-1-dot-8/src/gf.c:2530 [inlined]
ijl_apply_generic at /cache/build/default-amdci5-0/julialang/juli
a-release-1-dot-8/src/gf.c:2545
threadid at ./task.jl:253
enq_work at ./task.jl:719
#schedule#613 at ./task.jl:800
schedule##kw at ./task.jl:789
notify at ./condition.jl:148 [inlined]
#notify#586 at ./condition.jl:142 [inlined]
notify at ./condition.jl:142 [inlined]
notify at ./condition.jl:142 [inlined]
task_done_hook at ./task.jl:617
unknown function (ip: 0x7efdc31f5122)
_jl_invoke at /cache/build/default-amdci5-0/julialang/julia-relea
se-1-dot-8/src/gf.c:2367 [inlined]
ijl_apply_generic at /cache/build/default-amdci5-0/julialang/juli
a-release-1-dot-8/src/gf.c:2549
jl_apply at /cache/build/default-amdci5-0/julialang/julia-release
-1-dot-8/src/julia.h:1838 [inlined]
jl_finish_task at /cache/build/default-amdci5-0/julialang/julia-r
elease-1-dot-8/src/task.c:254
start_task at /cache/build/default-amdci5-0/julialang/julia-relea
se-1-dot-8/src/task.c:942

atexit hook threw an error: MethodError(f=Integer, args=("string"
,), world=0x0000000000007eb4)
jl_method_error_bare at /cache/build/default-amdci5-0/julialang/j
ulia-release-1-dot-8/src/gf.c:1879
jl_method_error at /cache/build/default-amdci5-0/julialang/julia-
release-1-dot-8/src/gf.c:1897
jl_lookup_generic_ at /cache/build/default-amdci5-0/julialang/jul
ia-release-1-dot-8/src/gf.c:2530 [inlined]
ijl_apply_generic at /cache/build/default-amdci5-0/julialang/juli
a-release-1-dot-8/src/gf.c:2545
length at ./range.jl:749 [inlined]
```

Спасибо за внимание!