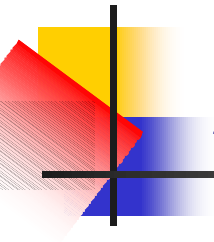
Tutorial do OiL – ORB in Lua

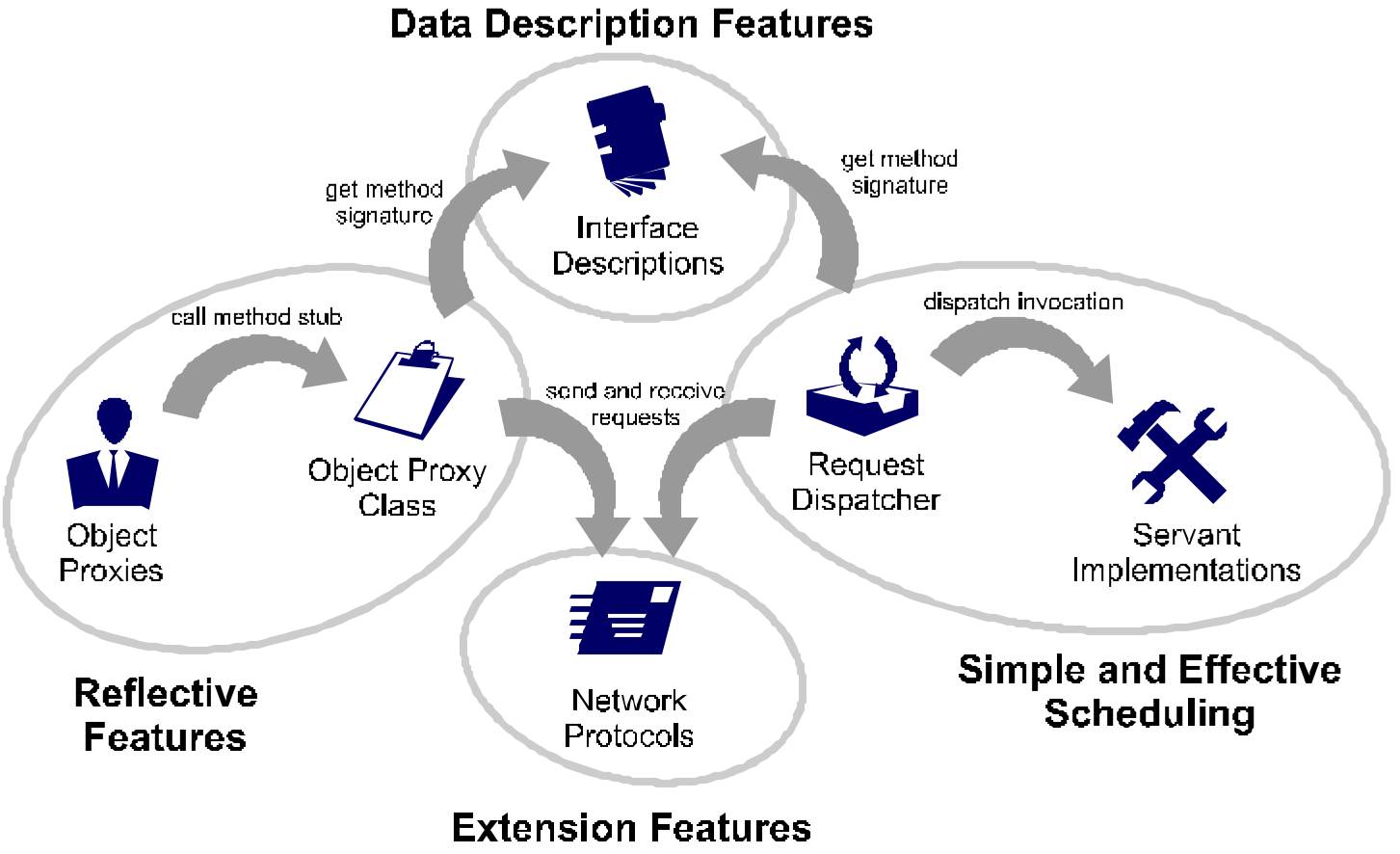
Renato Maia - maia@inf.puc-rio.br
Departamento de Informática
PUC-Rio

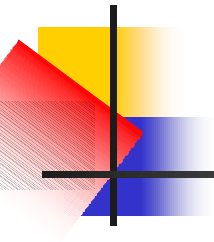


Agenda

- Visão Geral
- Pacote de Implantação
- Definição de Interfaces
- Implementação de Objetos
- Proxies de Objetos
- Concorrência Cooperativa
- Modularização
- Serviços de CORBA
- Trabalhos Futuros

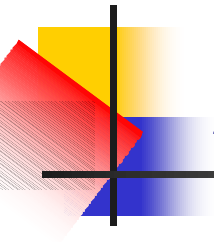
Visão Geral


Data Description Features
get method signature
get method signature
Interface Descriptions
call method stub
dispatch invocation
send and receive requests
Object Proxies
Object Proxy Class
Request Dispatcher
Servant Implementations
Network Protocols
Reflective Features
Simple and Effective Scheduling
Extension Features



Pacote de Implantação

- Software necessário
  - Lua e LuaSocket
- Modelos de Implantação
  - Padrão [make install]
    - Scripts Lua compilados e carregados dinamicamente através do LUA_PATH
  - Pré-Compilado [make installb]
    - Bibliotecas dinâmicas carregadas pelo LUA_CPATH
  - Embutido [make installp]
    - Única biblioteca para ser lincada à aplicação hospedeira



A Linguagem Lua

- Simples, flexível, pequena e eficiente
- Multi-paradigmática (extensível)
  - Procedural (funções e variáveis globais)
  - Funcional (funções são valores de primeira-classe)
  - Orientação a objetos

```
-- Açúcar sintático
obj = { nome = "João" }
function obj:diga_mensagem(texto)
    print(self.nome..":", texto)
end
obj:diga_mensagem("Olá todos!")
```

```
-- Código equivalente
-- obj = { ["nome"] = "João" }
-- obj["diga_mensagem"] = function(self, texto)
--   print(self["nome"]..":", texto)
-- end
-- obj["diga_mensagem"](obj, "Olá todos!")
```

Definição de Interfaces

- Compilador de IDL

```
oil.loadidl("interface IHello { void hello(in string to); };")
oil.loadidlfile("/usr/etc/idl/hello.idl")
```

- Repositório de Interface Remoto

```
oil.setIR(oil.newproxy(oil.readIOR("ir.ior")))
```

- Descritores Lua

```
oil.Manager:putiface(oil.idl.interface{
  name = "IHello",
  members = {
    hello = oil.idl.operation{
      parameters = {{ name="to", type = oil.idl.string }},
}}})
```

Implementação de Objetos

- Objeto de Implementação
  - Valor indexável que mapeie o nome do membro para o valor que o implemente.

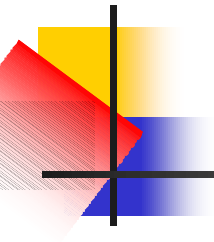
```
local impl = { atributo = "valor do atributo",
               operacao = function(self, parametro)
                 return valorDeRetorno
               end  }
```

- oil.newobject(impl, iface)
  - Recebe o objeto de implementação e a interface que o objeto implementa.

```
oil.newobject(impl, "::MyModule::MyInterface")
oil.newobject(impl, oil.idl.interface{ ... })
```

Implementação de Operações

```
oil.loadidl("interface Valvula {
    boolean abrir(inout long tentativas, in double tempo); };")
local Impl = {}
function Impl:abrir(tentativas, tempo)
    for i=1, tentativas do
        tentaAbrirAValvula()
        sleep(tempo)
        if valvulaAberta() then return true, i end
    end
    return false, tentativas
end
```



Implementação de Atributos

```
oil.loadidl [[
    interface WorkMeter {
        attribute string name;
        readonly double workload; };
]]

local Impl = { name = "MainNode" }
function Impl:_get_workload()
    return getCurrentWorkload()
end
```

Lançamento de Exceções

```
oil.loadidl [[
  exception LuaError { string script; string message; };
  interface RemoteLua {
    void doscript(in string path) raises (LuaError); };
]]
local Impl = {}
function Impl:doscript()
  local success, errmsg = loadfile(path)
  if success then success, errmsg = pcall(success) end
  if not success then
    error{ "IDL:LuaError:1.0", script = path, message = errmsg }
  end
end
```

Proxies de Objetos

- Representam objetos remotos
- oil.newproxy(ref, [iface])
  - Recebe uma referência do objeto
    - IOR, corbaloc
  - Opcionalmente, pode-se informar a interface oferecida pelo objeto remoto
  - Exemplo

```
oil.loadidlfile("/usr/etc/idl/remotelua.idl")
local remote = oil.newproxy("IOR:0ffffa...", "RemoteLua")
remote:doscript("/usr/etc/lua/script.lua")
```

Captura de Exceção

- Chamada protegida

```
oil.loadidlfile("/usr/etc/idl/remotelua.idl")
local obj = oil.newproxy("IOR:0ffffab...", "RemoteLua")
local ok, ex = pcall(obj.doscript, obj, "/usr/etc/lua/script.lua")
if not ok then print(ex) end
```

- Tratadores de exceção

```
oil.loadidlfile("/usr/etc/idl/remotelua.idl")
local obj = oil.newproxy("IOR:0ffffab...", "RemoteLua")
obj._handlers={ ["IDL:LuaError:1.0"]=function(obj,ex,op,arg)
    print("exceção na chama da operação "..op.name, ex)
end }
obj:doscript("/usr/etc/lua/script.lua")
```

Exemplo

```
require "oil"
oil.loadidl [[
    interface User { void say(in string msg); }
    interface Server { void log( in string name,
                                 in User user);
                       User get(in string name); }
]]
local Srv = { users = {} }
function Srv:log(name, user)
    self.users[name] = user
end
function Srv:get(name)
    return self.users[name]
end
oil.writeIOR(oil.newobject(Srv, "Server"), "srv.ior")
oil.run()
```

```
require "oil"
oil.loadidl [[
    interface User { void say(in string msg); }
    interface Server { void log( in string name,
                                 in User user);
                       User get(in string name); }
]]
local Usr = {}
function Usr:say(msg) print(msg) end
local Srv = oil.newproxy(oil.readIOR("srv.ior"))
Srv:log("World", Usr)
oil.run()
```

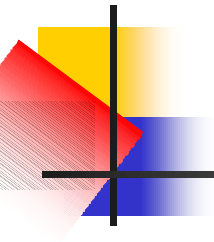
```
require "oil"
local Srv = oil.newproxy(oil.readIOR("srv.ior"))
local Usr = Srv:get("World")
Usr:say("Hello World!")
```

Exemplo 2

```
require "oil"

oil.loadidl [[
    interface User { attribute string name;
                     void say(in string msg); }
    interface Server { void log(in User user);
                       User get(in string name); }
]]

local Srv = { users = {} }
function Srv:log(user)
    self.users[user.name] = user
end
function Srv:get(name)
    return self.users[name]
end

oil.writeIOR(oil.newobject(Srv, "Server"), "srv.ior")
oil.run()
```

```
require "oil"

oil.loadidl [[ // mesmo que antes ]]

local Usr = { name = "World" }
function Usr:say(msg) print(msg) end
local Srv = oil.newproxy(oil.readIOR("srv.ior"))
Srv:log(Usr)

oil.run()
```

```
require "oil"

local Srv = oil.newproxy(oil.readIOR("srv.ior"))
local Usr = Srv:get("World")
Usr:say("Hello World!")
```



Concorrência Cooperativa

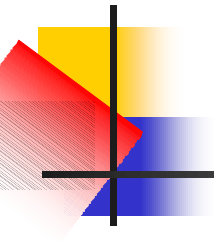
- Comparação (modelo preemptivo)
  - Threads trocam de contexto explicitamente
    - Através da chamada 'coroutine.yield()'
    - Chamadas de rede bloqueantes
  - Mais simples de implementar e depurar
    - Todo código é atômico por default
  - Mais eficiente
    - Evita troca de contextos desnecessárias
  - Desvantagem
    - Garantia de *fairness* e ausência de *starvation* fica por conta do programador

Exemplo com concorrência

```
require "scheduler"
require "oil"
oil.loadidl [[
    interface User { attribute string name;
                     void say(in string msg); }
    interface Server { void log(in User user);
                       User get(in string name); }
]]
local Srv = { users = {} }
function Srv:log(user)
    self.users[user:_get_name()] = user
end
function Srv:get(name)
    return self.users[name]
end
oil.writeIOR(oil.newobject(Srv, "Server"), "srv.ior")
scheduler.new(oil.run)
scheduler.run()
```

```
require "scheduler"
require "oil"
scheduler.new(function()
    oil.loadidl [[ // mesmo que antes ]]
    local Usr = { name = "World" }
    function Usr:say(msg) print(msg) end
    local Srv=oil.newproxy(oil.readIOR("srv.ior"))
    Srv:log(Usr)
end)
scheduler.new(oil.run)
scheduler.run()
```

```
require "oil"
local Srv = oil.newproxy(oil.readIOR("srv.ior"))
local Usr = Srv:get("World")
Usr:say("Hello World!")
```



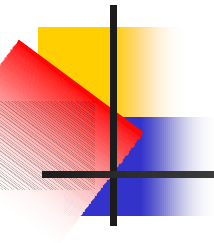
Modularização

- Repositório de Interfaces
  - Nomes absolutos de interfaces
  - Operação 'get_interface()'
- Compilador de IDL
  - Definição de interfaces em IDL
- Proxy de Objetos
  - Açúcar para manipular objetos remotos
- Gerenciador de Interfaces
  - Criação automática de proxies recebidos pela rede
  - Adaptação dinâmica de proxies para novas IDLs
- Servidor
  - Criação explícita e implícita de objetos CORBA
  - Entrega de requisições de objetos

Exemplos Mínimos

```
require "oil.iiop"
require "oil.orb"

local iface = oil.idl.interface{
  name = "Hello",
  members={hello=oil.idl.operation{}},
}

local imp = {}
function imp:hello() print("Hello") end

local orb=oil.orb.init{ port = 8080 }
local obj=orb:object(imp,iface,"Hello")
WriteToFile(obj:_ior(), "hello.ior")
orb:run()
```

```
require "oil.iiop"
require "oil.ior"
require "oil.invoke"

local op = oil.idl.operation{
    name = "hello",
}

local ref = ReadFromFile("hello.ior")
ref = oil.ior.decode(ref)

oil.invoke.call(ref, op)
```



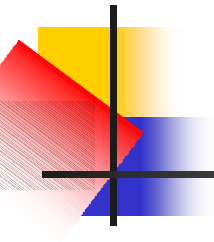
Serviços de CORBA

- Repositório de Interfaces

```
require "oil"
oil.writeIOR(oil.getLIR(), "ir.ior")
oil.run()
```

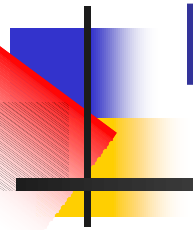
- Serviço de Nomes

```
require "oil"
require "oil.cos.naming"
oil.loadidlfile "CosNaming.idl"
oil.writeIOR(oil.newobject(oil.cos.naming.new()), "ns.ior")
oil.run()
```



Trabalhos Futuros

- Documentar todos recursos
- Evoluir a implementação atual
  - Testar e avaliar recursos atuais
  - Componentização
    - Melhor modularização e personalização
    - Suporte multi-protocolo extensível
- Implementar novos recursos
  - Tipos avançados
  - Novas versões do IIOP
  - Políticas do POA
  - Operações do ORB
  - Interceptores portáteis
  - Componentes CCM



FIM

Dúvidas e discussão