Linguagem Lua

Francisco Sant'Anna

Introdução

- Multi-paradigma
  - procedural, OO, funcional
- Multi-plataforma (ANSI-C)
  - PC (Windows, Mac, Linux), Sis
- Dinâmica
  - *eval*, tipagem dinâmica, tabelas,
- Foco em *scripting*
  - configuração, macros, extensão
  - nicho em video games

```
// procedural
for i=1, 10 do
  if i % 2 == 0 then
    local v = i*i
    print(i,v)
  end
end
```

```
// OO
function f (self, v)
  return self.v + v
end
o = { v=10, m=f }
print(o:m(10)) -- 20
print(o.m(o,10))
```

```
// funcional
t = { 10,1,5 }
table.sort(t,
  function (v1,v2)
    return v1 > v2
  end
) -- {10,5,1}
```

Lua em Video Games

Origens e Influências

- Modula (sintaxe)
- Scheme (semântica)
- CLU (atribuição e retorno múltiplo)
- SNOBOL e AWK (array associativo)

Expressividade

Exemplo 1: Closures

- `code/counter-0[0-3].lua`
- Funções
  - puras vs impuras
- Estado
  - global vs encapsulado (composição?)
- Colsure como "cidadão de primeira classe"
  - Programação funcional
  - Atribuição, passagem, retorno, **criação dinâmica**
- Com o que se parecem `c1` e `c2`?

Exemplo 2: Iteradores

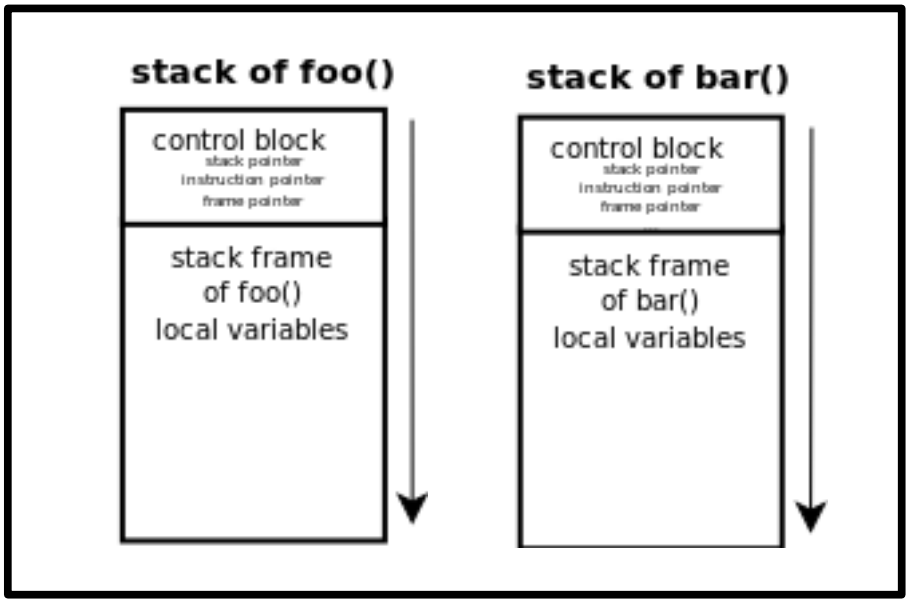
```
for i=1, 10 do
    local v = i*i
    print(i,v)
end
```

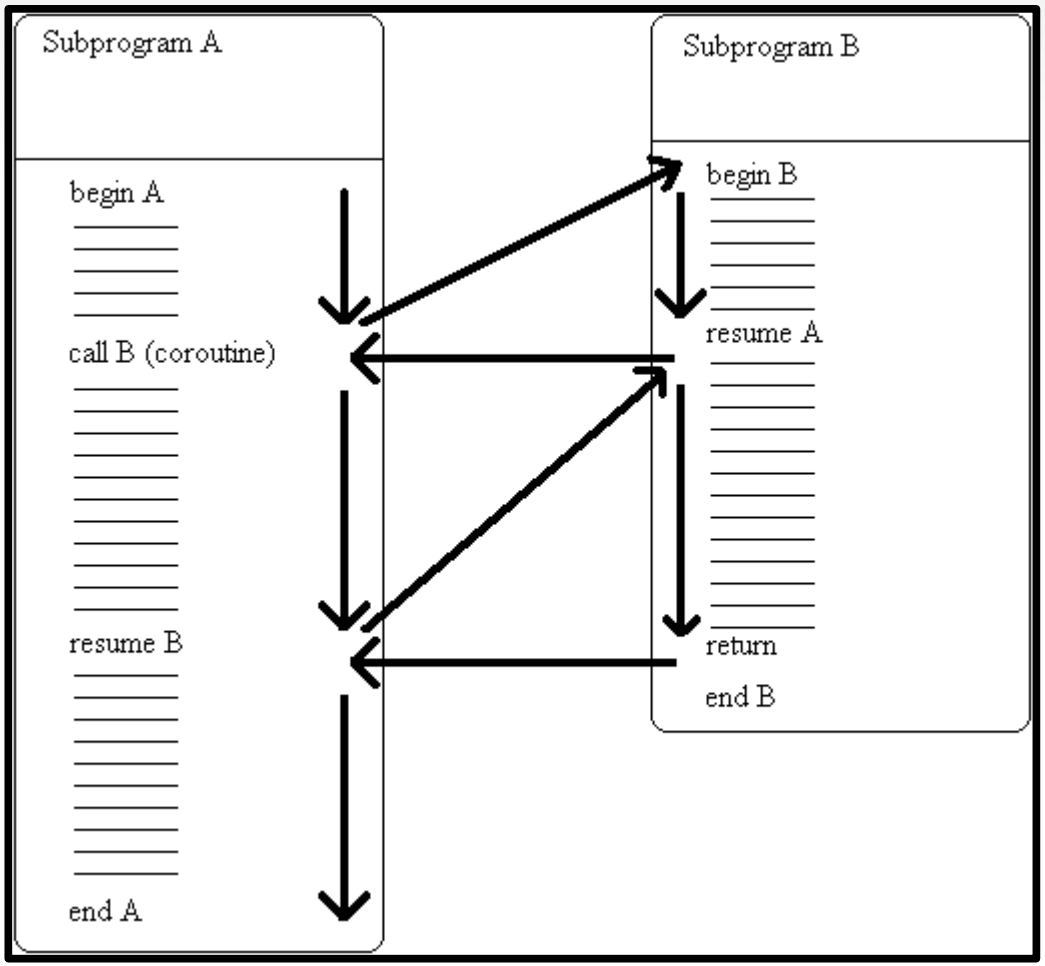
```
for i,v in <f_iter> do
    print(i,v)
end
```

- `code/iterator-0[1-3].lua`
- Estado global e encapsulado

Exemplo 2: Co-rotinas

- `code/iterator-0[4-5].lua`
- Contexto = PC, SP, pilha separada
  - estado implícito





Exemplo 2: Co-rotinas

- Controle/Pilha como “cidadão de primeira classe”
- Iteradores, Multi-Tarefa cooperativa

Comparison with subroutines [edit]

"Subroutines are special cases of ... coroutines." –Donald Knuth.[3]

Exemplo 3: Co-rotinas

- Corrida entre dois jogadores
- `code/game.lua`
- API: `player1(), player2()`
- Retorno: `'move' ou 'stand'`