




---

---

---

---

---

---

---

## Introducción

- Una forma de separar funcionalidad
- Proporciona ciertas ventajas para gestionar
- Muy semejante a un thread convencional
- Tiene su propia pila, variables locales, etc.
- **No se ejecutan en paralelo**
  - Sólo se ejecuta una corutina a la vez
- Sirven para colaborar entre ellas

© Diego Garcés 25/06/2014

---

---

---

---

---

---

---

## Creación

- Coroutine
- Librería con operaciones para gestión de corutinas
- `coroutine.create(funcion)`
  - Crea una corutina para la función que se pasa como parámetro
- Devuelve un valor de tipo thread

© Diego Garcés 25/06/2014

---

---

---

---

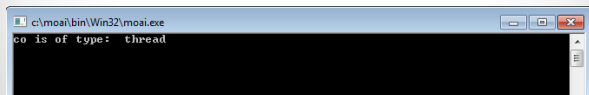
---

---

---

## Creación

```
function saludar()  
  print("Hello world")  
end  
  
co = coroutine.create(saludar)  
print("co is of type:", type(co))
```



```
c:\moai\bin\Win32\moai.exe  
co is of type: thread
```

© Diego Garcés

25/06/2014

---

---

---

---

---

---

---

## Estados

- Una corutina puede estar en 4 estados
  - Suspendida (suspended)
  - Ejecutando (running)
  - Muerta (dead)
  - Normal (normal)
- Al crearlas están en estado suspendido
  - No empiezan a ejecutar nada más crearlas
- Para obtener el estado: `coroutine.status(corutina)`

© Diego Garcés

25/06/2014

---

---

---

---

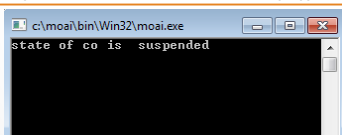
---

---

---

## Estados

```
function saludar()  
  print("Hello world")  
end  
  
co = coroutine.create(saludar)  
print("state of co is", coroutine.status(co))
```



```
c:\moai\bin\Win32\moai.exe  
state of co is suspended
```

© Diego Garcés

25/06/2014

---

---

---

---

---

---

---

## Ejecución

- `Coroutine.resume(corutina)`
  - Pone en ejecución la corutina
  - Cambia el estado a `running`
  - Bloquea la ejecución de quién la llama hasta que termine la corutina

© Diego Garcés

25/06/2014

---

---

---

---

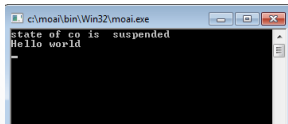
---

---

---

## Ejecución

```
function saludar()  
    print("Hello world")  
end  
  
co = coroutine.create(saludar)  
print("state of co is", coroutine.status(co))  
coroutine.resume(co)
```



© Diego Garcés

25/06/2014

---

---

---

---

---

---

---

## Estados: Dead

- Al terminar la ejecución de la función de la corutina → estado **dead**
- Si se intenta hacer `resume` en ese estado → Error

© Diego Garcés

25/06/2014

---

---

---

---

---

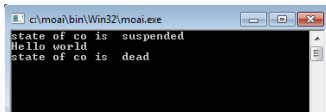
---

---

## Estados: Dead

```
function saludar()
  print("Hello world")
end

co = coroutine.create(saludar)
print("state of co is", coroutine.status(co))
coroutine.resume(co)
print("state of co is", coroutine.status(co))
```



```
c:\moai\bin\Win32\moai.exe
state of co is suspended
Hello world
state of co is dead
```

© Diego Garcés

25/06/2014

---

---

---

---

---

---

---

---

## Yield

- `coroutine.yield()`
- Para la ejecución de una corutina
- Llamado desde dentro de la corutina
  - Función que se usó para crear la corutina
- Devuelve el control a quién llamó a `coroutine.resume`
- Semejante a un `return` de una función
  - Con diferencias

© Diego Garcés

25/06/2014

---

---

---

---

---

---

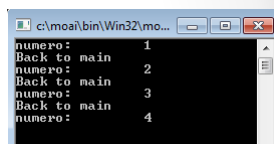
---

---

## Yield

```
function escribe()
  for i=1, 10 do
    print("numero: ", i)
    coroutine.yield()
  end
end

co = coroutine.create(escribe)
coroutine.resume(co)
print("Back to main")
coroutine.resume(co)
print("Back to main")
coroutine.resume(co)
print("Back to main")
coroutine.resume(co)
print("Back to main")
```



```
c:\moai\bin\Win32\moai.exe
numero: 1
Back to main
numero: 2
Back to main
numero: 3
Back to main
numero: 4
```

© Diego Garcés

25/06/2014

---

---

---

---

---

---

---

---

## Estados: Normal

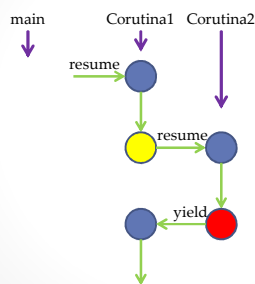
- Estado usado cuando hay varias corutinas
- Una corutina hace resume a otra
- Esta última se pone en ejecución
- La anterior debe pararse
  - No hay ejecución paralela
  - No está suspendida → Podemos hacer resume de ella
  - No está dead → No ha terminado de ejecutarse
  - Está en estado normal

• © Diego Garcés

25/06/2014 •

[illegible]

## Estados: Normal



• © Diego Garcés

25/06/2014 •

---

---

---

---

---

---

## Errores

- Si ocurre un error durante una corutina no salen mensajes de error
- Se devuelve el error como resultado de `coroutine.resume`
- Así se protege la ejecución

• © Diego Garcés

25/06/2014 •

---

---

---

---

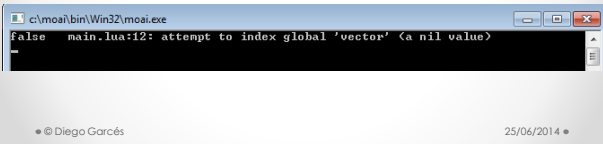
---

---

## Errores

```
function sumar()
    var = vector.X + 3
end

co = coroutine.create(sumar)
ret, msg = coroutine.resume(co)
print(ret, msg)
```



© Diego Garcés

25/06/2014

## Colaboración

- Utilidad de corutinas → Colaboración
- Necesidad de paso de información entre ellas
- `coroutine.resume(co, param, param, ...)`
  - Llamada a función
- `coroutine.yield(param, param, ...)`
  - Return

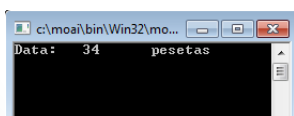
© Diego Garcés

25/06/2014

## Colaboración

```
function printer(v1, v2)
    print("Data: ", v1, v2)
end

co = coroutine.create(printer)
coroutine.resume(co, 34, "pesetas")
```



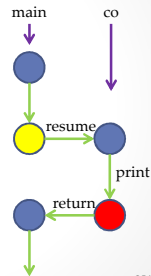
© Diego Garcés

25/06/2014

## Colaboración

```
function printer(v1, v2)
  print("Data: ", v1, v2)
end

co = coroutine.create(printer)
coroutine.resume(co, 34, "pesetas")
```



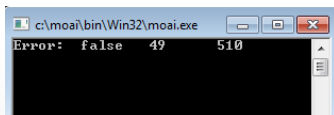
© Diego Garcés

25/06/2014

## Colaboración

```
function printer(v1, v2)
  return v1 + v2, v1 * v2
end

co = coroutine.create(printer)
success, suma, mul = coroutine.resume(co, 34, 15)
print("Error: ", not success, suma, mul)
```



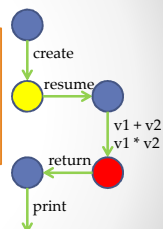
© Diego Garcés

25/06/2014

## Colaboración

```
function printer(v1, v2)
  return v1 + v2, v1 * v2
end

co = coroutine.create(printer)
success, suma, mul = coroutine.resume(co, 34, 15)
print("Error: ", not success, suma, mul)
```



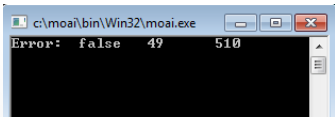
© Diego Garcés

25/06/2014

## Colaboración

```
function printer(v1, v2)
  coroutine.yield(v1 + v2, v1 * v2)
end

co = coroutine.create(printer)
success, suma, mul = coroutine.resume(co, 34, 15)
print("Error: ", not success, suma, mul)
```



© Diego Garcés

25/06/2014

---

---

---

---

---

---

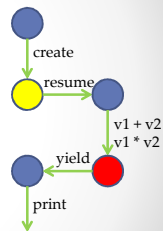
---

---

## Colaboración

```
function printer(v1, v2)
  coroutine.yield(v1 + v2, v1 * v2)
end

co = coroutine.create(printer)
success, suma, mul = coroutine.resume(co, 34, 15)
print("Error: ", not success, suma, mul)
```



© Diego Garcés

25/06/2014

---

---

---

---

---

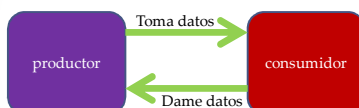
---

---

---

## Ejemplos

- Productor / Consumidor



© Diego Garcés

25/06/2014

---

---

---

---

---

---

---

---





## Ejemplos

- Filtro: Introducir un elemento que filtre los datos antes de procesarlos

© Diego Garcés

25/06/2014

---

---

---

---

---

---

---

---

```
function productor()
  local datos = {1, 3, 5}
  local i = 1
  while i < #datos do
    local x = datos[i]
    coroutine.yield(x)
    i = i + 1
  end
  return datos[#datos]
end

function consumidor()
  local prod = coroutine.create(productor)
  local success = true
  local dato = nil
  while coroutine.status(prod) ~= "dead" do
    success, dato = coroutine.resume(prod)
    -- procesar(dato)
    print("Procesado: ", dato)
  end
end
```

### Filtro

```
function productorC()
  local prod = coroutine.create(productor)
  local status = true
  local dato = nil
  while coroutine.status(prod) ~= "dead" do
    status, dato = coroutine.resume(prod)
    dato = dato - 1
    coroutine.yield(dato)
  end
end
```

consumidor()

```
c:\moa\bin\Win32\...
Procesado: 0
Procesado: 2
Procesado: 4
Procesado: nil
```

© Diego Garcés
25/06/2014

---

---

---

---

---

---

---

---

```
function productor()
  local datos = {1, 3, 5}
  local i = 1
  while i < #datos do
    local x = datos[i]
    coroutine.yield(x)
    i = i + 1
  end
  return datos[#datos]
end

function consumidor()
  local prod = coroutine.create(productor)
  local success = true
  local dato = nil
  while coroutine.status(prod) ~= "dead" do
    success, dato = coroutine.resume(prod)
    -- procesar(dato)
    print("Procesado: ", dato)
  end
end
```

### Filtro

```
function productorC()
  local prod = coroutine.create(productor)
  local status = true
  local dato = nil
  while coroutine.status(prod) ~= "dead" do
    status, dato = coroutine.resume(prod)
    dato = dato - 1
    if coroutine.status(prod) == "dead" then
      return dato
    else
      coroutine.yield(dato)
    end
  end
end
```

consumidor()

```
c:\moa\bin\Win32\...
Procesado: 0
Procesado: 2
Procesado: 4
```

© Diego Garcés
25/06/2014

---

---

---

---

---

---

---

---

## Ejemplos

- Iteradores
  - Genero un dato y espero a que me pidan otro
- Se pueden hacer iteradores simplemente usando corutinas directamente
- El ejemplo de productor es un ejemplo de iterador
- Me va devolviendo datos de un array local

© Diego Garcés

25/06/2014

---

---

---

---

---

---

---

## Wrap

- Lua proporciona cierta encapsulación para hacerlo más sencillo
- `res = coroutine.wrap(funcion)`
- **for val in res do**
- Semejante a
  - `for k, v in pairs(tabla) do`

© Diego Garcés

25/06/2014

---

---

---

---

---

---

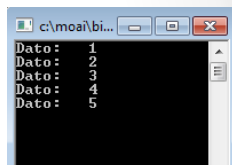
---

## Wrap

```
function getData()
    local datos = {1, 2, 3, 4, 5}
    for i = 1, #datos do
        coroutine.yield(datos[i])
    end
end

function datos()
    return coroutine.wrap(getData)
end

for d in datos() do
    print("Dato: ", d)
end
```



© Diego Garcés

25/06/2014

---

---

---

---

---

---

---