Coroutines in 8.6

Miguel Sofer

Tcl2008
Manassas, VA

New possibilities

NRE opens a world of new possibilities:

- Once TEBC knows how to freeze an execution ... put it aside and save it for later ⇶ coroutines
- Edit the Callback stack, rearrange the order of computations ⇶ proper tailcalls
- ??? ⇶ ???

Coroutines - what?

- A computation that can be suspended and resumed on demand or killed while suspended
- A tool to create *generators*
- A tool to simplify the coding of event-based programs
- A tool to enable cooperative multitasking within a single interp, without OS support

Coroutines: how?

- Tip #328: *New Commands and Subcommands*
- [yield] returns *twice*:
  - when yielding: caller receives a return value from the yielding coroutine
  - when resuming: coroutine sees the [yield] command returning
- [cmdName] is ”garbage collected”
- Example at `http://msofer.com:8080/wiki?name=Coroutines`

Coroutines: why?

- GUI programming ”made easier” (NEM)
- ”nestable vwait”
- Generators
- Servers (wub)
- Cooperative multitasking (event loop)
- Cooperative multitasking (dispatcher)

Coroutines: tech?

- [coroutine]
  - creates command and a new execEnv for it
  - callback in the old execEnv to swap back to it
  - Swaps in the new execEnv, callback to clean up
  - launches script
- [yield]
  - Freezes TEBC
  - Swaps out coro's execEnv
  - Returns value

Coroutines: tech? (2)

- [coroCmdName]
  - Swaps in execEnv
  - Resumes TEBC execution

Deleting a suspended coroutine winds-down the suspended execution (as if interp's limit had been reached)

Coroutines: special

- Command runs at [uplevel #0]
- You can [yield] from a nested call: these are *asymmetric stackful coroutines* in terms of Lua's paper
- If the command pushes a CallFrame (proc or lambda!)
  - Local variables retain their state when suspended
  - Local variables are at level #1