NRE: the non-recursive engine in Tcl8.6

Tcl8.6 has a completely redesigned execution model that allows among others arbitrary deep recursion in constant C-stack space. The design is derived from the ideas already present in mod-8-3-branch, implemented at a different level and much generalized. It is completely transparent to extensions, users of the standard Tcl APIs keep perfect compatibility.

The basic design of NRE will be discussed, with special emphasis on the trampolines that manage the execution flow and the structures that are kept in the C stack and heap during evaluation/execution.

We describe the new APIs to enable extensions to profit from stackless execution.

Tcl8.6's new non-recursive engine permits some interesting and efficient constructs that were difficult or impossible to express in previous versions. We will discuss in some detail the four new experimental commands included: tailcall, atProcExit, coroutine and yield.

The first command provides proper tailcalls. These permit arbitrary recursion in constant space (NRE itself only provides constant C stack), continuation passing style scripting and other techniques usually exploited in functional programming.

The second gives a simple way to insure inconditional code execution when a proc (or lambda) exits. This can be exploited to insure that resources are properly cleaned up and freed without using [catch] around every possible error cause.

The last two work as a pair to enable the creation of coroutines - or should they be called micro-threads? These are the basis on which several interesting features can be built: nestable vwaits, non-blocking after, simple non-blocking file access. They permit much simpler and safer coding for event-based programs, with garbage collection of a sort and avoiding the artificial split of functions that is often needed.