# Creole Forth and Safecrosser

Forth as a DSL in Delphi and Lazarus

By

Joseph M. O’Connor

## Who I am

- A Forth enthusiast of many years
- Do most of my development work on Windows and Unix.
- First intro to Forth – Tom Zimmer’s F-PC and Starting Forth.
- Later on, dabbled in Norman Smith’s UNTIL.
- UNTIL inspired me to write my own version of Forth in Delphi.

# Delphi, the good parts

- A GUI, visually-driven environment.
- Powerful OO language. (Object Pascal).
- Based on components (objects on steroids).
- Extremely rapid compilation.
- Suitable for rapid prototyping.
- Allows user to write new components.

# Delphi, the not so good parts

- Very Windows-centric
- An expensive product.
- Product ownership has changed hands more than once.

## Lazarus – a parallel IDE

- Lazarus is a product inspired by Delphi.
- Is open source.
- Based on the Free Pascal compiler.
- Was started around 1999
- Became practically usable around 2008-2009
- Goal is write-once/compile anywhere
- Is fairly portable to different platforms.

## Background on Creole Forth

- Developed originally as a Delphi component
- Inspired by Norman Smith’s UNTIL.
- Ported to Lazarus with minimal changes (so far).
- Intended as a small DSL or scripting language to run on top of a Delphi/Lazarus application.

## A few Creole Forth projects

- Sample demo app (includes DLL calls, Perl script execution, and simple Web server).
- Two-way client/server. Submits report requests to a web server on Unix and returns the results.
- Multitier file handler. Transparently converts Excel files to comma-delimited format using excel and transmits to a Unix server for processing.

# Build an app in five minutes

- Open Lazarus
- Create a new form-based application
- Drop on the form a memo component, a button component, and a Creole component from the palette.
- In the FormCreate method of the form, call the RebuildDefs method of the Creole component.

## Build an app in five minutes 2

- Double-click the button, set Creole1’s Input property to the Memo’s Lines property.
- Call the submit method of Creole.
- That’s it!

# Real-world applications

- Take a bit more than five minutes to develop (no kidding!)
- Still, it’s feasible to take the core vocabulary of Creole (currently 64 words), add perhaps 40-50 primitives and high-level definitions, and have a fully-functioning application.

## Example: Safecrosser

- First Creole Forth application developed in Lazarus – other apps were in Delphi.
- Compiled executable weighs in at about 600 kb – very lightweight by current standards.
- No registry or ini file dependencies.
- No installation required.
- Has about 30-40 primitives

## Key advantages of using Creole

- Powerful scriptability.
- Minimal overhead.
- Add what you need to your language and no more.
- Underlying code is simple and extensible.
- Can write primitives in Object Pascal.
- Extension mechanism(s) are simple.

# Creole is Forth with a few twists

- Fundamental data type is the string
- Has five built-in stacks (parameter, return, vocabulary, prefilter, and postfilter).
- Dictionary is a hash table built using the TStringList data type.
- Has no “state” variable to enforce compilation or interpretation.
- Vocabularies are enforced through encryption.

## Some links and references

- Latest version of Lazarus
- Creole at Github
- Creole on public Dropbox folder
- Norman Smith's UNTIL
- Wikipedia article on Delphi
- Wikipedia article on Lazarus