Creole Forth In-Depth

Joseph M. O'Connor

What is Creole Forth?

- A scripting language
- Originally developed in Delphi
- Is now ported to Lazarus
- Exists as a drop-in component inside the Delphi or Lazarus environment.

What is Delphi?

- A RAD IDE
- RAD = Rapid Application Development
- IDE = Integrated Development Environment
- Build apps primarily by dropping components on a form.
- Lives (mostly) on Windows
- Is not open-source

What is Lazarus?

- A RAD IDE like Delphi
- Was inspired by Delphi
- Build apps primarily by dropping components onto a form.
- Available on a number of environments besides Windows.
- Write once, compile anywhere.
- Open source

Delphi vs Lazarus

- Delphi is a more mature environment
- Networking more stable / better-documented in Delphi
- Lazarus is free as in free beer
- Lazarus is less Windows-centric than Delphi
- Lazarus as of 2012 is “good enough”

Example apps using Creole

- Simple demo app. Perl interface, DLL, web server.
- Two-way client-server reporting app.
- Multitier spreadsheet handler.
- Safecrosser. For data hiding by travelers.

How to use Creole

- Drop the TCreole component onto an application
- Define any primitives needed.
- Call the corresponding BuildPrimitive method
- Define high-level defs and load as needed.
- Call RebuildDefs method in FormCreate method. (Important!)

Creole extension mechanisms

- Defining new primitives
- Colon compiler
- Defining words
- Compiling words
- Creating a new compiler – i.e. help compiler
- Prefilter stack

Prefilter stack?

- Before code in the input stream is submitted to Creole,
  it is 'filtered' through Creole by any words on the prefilter stack.
- All prefilter words are in the prefilter vocabulary
- Currently is used for stripping out comments.
- An entire language could be embedded within.
- Is thus a valid extension mechanism of Creole

Postfilter stack

- Can be used to enforce integers-only or floating-point only.
- Can therefore enforce a more conventional Forth rule-set.
- Current default value on post-filter stack – NONE.
- NONE lets anything on the stack that isn't in the dictionary.
- Defined in the post-filter vocabulary

Creole oddities, Part 1

- Due to its working within a Delphi / Lazarus environment, apps built will be more “massive” than conventional Forths.
- Possible to get down to about 1 meg on Linux, 600k on Windows.
- No STATE variable. Thanks to Chuck Moore and Jeff Fox for inspiring this feature.
- Colon compiler starts its process by pushing the IMMEDIATE vocabulary onto the vocabulary stack.
- Compiling words such as Compile_Do are always searched for first.
- Semicolon terminator halts compilation by popping IMMEDIATE off the vocab stack.

Creole oddities, Part 2

- Namespacing is enforced via encryption.

- Each word when compiled is encrypted based on which vocabulary it exists in.

- Outer interpreter lookup mechanism encrypts each word before lookup based on the vocabularies on the vocabulary stack.

- Each vocabulary on the stack is searched.

Example of vocabulary search

- Searching for the word “Hello”

ONLY
FORTH

- Creole first searches the ONLY vocab.
- Since it fails, it searches the FORTH vocab.
- The second search is successful and “Hello” in the FORTH vocabulary is executed.
- Once execution is complete, the search does not proceed further.

Creole oddities, Part 3

- Outer interpreter searches by hashing
- Inner interpreter searches by indexing
- “Addresses” are really indexes. Definition 1 is index 0 in the Dictionary.
- Parameter field is an array of indexes

More Creole Internals

- Dictionary is a TStringList.
-
- TStringlist is an Object Pascal / Free Pascal data type.

- It's a container that can hold anything.

- In this case it holds an (encrypted) name, and a value of type TCreoleWord.

More Creole Internals, Part 2

- Creole words have procedures that can be inserted dynamically.
- A primitive would have its own code defined in Pascal in its Code Field.
- A colon def would have DoColon in its Code Field.

How I've used Creole (usually)

- Define and test primitive procedures.

- These are procedures with two interfaces :
- TExtInterface and TDictInterface.

- Add them to the BuildPrimitive list.

- Create any high-level defs needed.

- High-level defs can be loaded from a text file or embedded in a Tmemo component.

- Resulting app is composed of a “lexicon” of perhaps 40-50 words on top of the Creole built-in word set.

Guidelines / Lessons learned

- Have the primitives defined in their own file.
- Building of primitives must be done separately and should be handled in application's FormCreate method.
- Networking primitives should be built into the core of the language (unfortunately, they aren't yet).

Example apps

- 5-minute app (Linux)
- Sample app (Linux) – has one user-defined primitive
- Safecrosser (developed for Windows, recompiled in Linux)