Forth System Hooks — Metaobject Protocol in Forth Style

Ulrich Hoffmann
uh@fh-wedel.de

September 18, 2008

Abstract

The ANS Forth Standard and its follow up standard effort, Forth 200x, allow for writing Forth applications portable to a wide range of commercial and open source Forth systems. One area of Forth which has been notoriously hard to standardize is the area of meta/target compilers as well as advanced compiler and interpreter extensions.

Many Forth systems implement system specific extension mechanisms in order to support just their special meta/target compiler but there is no common practice how to do so. One extension mechanism is to place execution vectors — or *hooks* — at key positions in the Forth system. In the un–extended case, the vectors have no or a default behavior and in the extended case the can get sophisticated and elaborated behavior.

One example of a very simple hook would be a vector *notfound* in the typical outer interpreter, which would be located right at the end when both, token lookup in the dictionary and the conversion of the token to a number, failed. In the un–extended case *notfound* would issue an error message (*complaint*) about an unknown token. In the extended case, say hexadecimal number input with `$`–prefix, *notfound* could try a hexadecimal number conversion, and leave the appropriate number on the stack, or issue an error message as before.

We could write a hexadecimal number input with `$`–prefix as a portable Forth system extension, if we would agree on the hook `notfound` and its default behavior.

Other extensions that come to mind are object oriented systems with advanced search order requirements such as Manfred Mahlows CSP (Context Switching Prelude) system. It also can benefit from well defined system hooks.

The idea of hooks is quite common, not only with Forth system but also with operating systems in general. The EMACS editor provides a vast collection of hooks for all kinds of extensions.

In the context of Common Lisp's Object System CLOS, a technique called the Meta Object Protocol has been developed by Gregor Kiczales in the early 1990s which allowed to implement an early form of aspect oriented programming. The advantage of their unique approach was that the CLOS community widely accepted the functionality of their hook pendants as defined in the book *The Art of Meta Object Programming* (AMOP). As a result AMOP based CLOS extensions were portable over a wide variety of CLOS implementations.

The talk will give a short overview of hooks and the CLOS meta object protocol and will also propose possible Forth system hooks in traditional Forth implementations with hope to come closer to a Forth system hook standardization.