Abstracts of the Eleventh Asilomar FORML Conference

November 24 – 26, 1989
Asilomar Conference Center
Pacific Grove, California

Forth Poetry or Forth Program? Forth Coding Style and Naming Convention

C. H. Ting
San Mateo, CA

Compound names can be used to help documenting Forth words in an easy and natural way, making Forth programs easy to compose and to be understood. Rules and examples are given to illustrate this coding style.

A Trail of Bread Crumbs

Peter Midnight
San Leandro, CA

I perceive a need to be able to SEE or VIEW any word I may care to use. This capability is my favorite feature of F83. The unfortunate side affect of some of the improvements in F-PC is that many words are made available for use through menu selections, function keys, and even mouse action without any way for the user to SEE or VIEW them because there is no practical way for him to find out what words those are. I wish to present my work on this problem.

Seeing Forth

Wil Baden
Costa Mesa, CA

Comparison of Forth with elementary C reveals essential differences in the two languages.

Kernighan and Ritchie, *The C Programming Language*, Second Edition, (henceforth called "the book") has an excellent tutorial introduction. It shows "the essential elements of the language in real programs, but without getting bogged down in details, rules, and exceptions." Translating these programs into another language will not necessarily show the virtues of the new language, but only how the new language handles the tasks performed by the examples. Translation into Forth is very instructive. I have used the same algorithms and data structures unless "you wouldn't do it that way in Forth." The Forth dialect used is that of BASIS9 with whatever else is necessary. The names of the functions have been given a Forth accent.

From Pascal to Forth

Leonard Morgenstern
304 Rheem Blvd.,
Moraga, CA 94556
GEnie: LMORGENSTERN

Fundamental differences in viewpoint between Pascal and Forth are more important than certain obvious dissimilarities in syntax. Pascal is conventional, with a mode of thinking that derives from algebraic notation. Forth is "organic," growing out of the processor and operating system, and permits the stepwise development of complex programming features.

Binary Radix Sort on 80286, 68010, and RTX2000

C. H. Ting
San Mateo, CA

Binary radix sort algorithm is implemented and compared on three different processors: 80286, 68010, and RTX 2000. It is very easy to implement and perform quite well using specialized move instructions in the processors.

An Extensible Optimizer for Compiling Forth

Andrew Scott
IDACOM Electronics Ltd.
4211 - 95 St • Edmonton, AB
CANADA T6E 5R6

Forth implementations on some processors suffer speed limitations. One solution to this problem has involved implementing a subroutine threaded system to eliminate inner interpreter overhead. Short primitives are usually compiled as inline code. This paper describes a method by which Forth can be compiled to produce more optimized code by combining sequences of Forth words into equivalent native instructions. The optimizer described does not require Forth primitives to be "smart" words nor does it require extensive changes to the normal Forth outer interpreter. Also, the optimizer is extensible, permitting new optimization rules to be added at compile time.

Four Different Programmers, Forths, and Computers

Guy M. Kelly
Kelly Enterprises
2507 Caminito La Paz
La Jolla, CA 90237

A group of four programmers attempt to create a new program on four different personal computers, using four different versions of Forth.

Multitasking or Multiple State Machines?

Guy M. Kelly
Kelly Enterprises
2507 Caminito La Paz
La Jolla, CA 90237

An approach to handling asynchronous inputs from the keyboard, a mouse, and a serial port using multiple state machines instead of a multitasker.

Graphics Based Smart Windows

Guy M. Kelly
Kelly Enterprises
2507 Caminito La Paz
La Jolla, CA 90237

A mouse-driven graphics-based windows package is described. Each defined window contains information about its class, job, current state, and associated mouse pointer type. Facilities are included for creating radio, toggle, momentary, and repeat buttons. A generalized requester structure is also provided.

A State Machine Based Drawing Package

Guy M. Kelly
Kelly Enterprises
2507 Caminito La Paz
La Jolla, CA 90237

A monochrome drawing package has been written as part of a communications program. The package uses state machines to implement the various drawing functions. The functions include dots and lines (with connected versions of each), boxes, circles, ellipses, and polygons (with filled versions of each), cubic splines, text (horizontal and vertical), and a variable sized eraser (which can be used as a drawing tool when erasing is in the foreground color).

Communications and State Machines

Guy M. Kelly
Kelly Enterprises
2507 Caminito La Paz
La Jolla, CA 90237

A serial communications package was needed for inclusion, with a windowing and drawing package, in an interactive on-line graphics program. The program design required either multitasking or the use of multiple state machines.

The serial communications package produced for use in the application uses a low-level serial interrupt service routine (ISR) for raw serial input and high-level state machines for serial i/o.

The serial input ISR stores incoming bytes in a circular buffer for handling by a high-level serial input state machine. The high-level serial input state machine reads bytes from the low-level circular buffer and assembles them into incoming communications packets and places the assembled packets into an appropriate packet buffer for processing by a packet handler.

The high-level serial output state machine takes outgoing packets from a circular packet buffer and transmits them a byte at a time. Both serial state machine handle ACKing and NAKing and time-outs. The output routine is also responsible for retries.

A 3-D Measurement System Using Object-Oriented Forth

Kenneth B. Butterfield
Los Alamos, NM

Discussed is a system for storing 3-D measurements of points that relates the coordinate system of the measurement device to the global coordinate system. The program described here uses object-oriented Forth to store the measured points as sons of the measuring device location. Conversion of local coordinates to absolute coordinates is performed by passing messages to the point objects. Modifications to the object-oriented Forth system are also described.

User-Defined Systems for Pure Mathematics

John J. Wavrik
Department of Mathematics
University of California - San Diego
La Jolla, California

Some branches of mathematics require the ability to represent unusual types of data and algorithms. Some of the needs of workers in these areas can be met by providing them tools to construct their own special purpose software systems. Forth has properties that make it very useful for this purpose. It is possible to build such systems from reusable parts and to create abstract types like "polynomial" and "matrix" which can, without writing new code, be specialized to polynomials with various kinds of coefficients and monomial parts, matrices with various kinds of entries, etc. — and which can be combined to form compound structures. This paper discusses an approach resulting in systems which manipulate complex mathematical objects using standard Forth syntax and semantics.

CRC Polynomials Made Plain

Wil Baden
Costa Mesa, CA

In data communications a cyclic redundancy code (CRC) polynomial is often used as a method for error detection. It is option in the popular XMODEM protocol and is required by international communications standards. Many programmers have heard of it and even implemented it, but there are few who understand it.

Hierarchical Objects from Flat Vocabularies

Mike Elola
San Jose, CA

Without mentioning message-passing, the author discusses how many of its benefits can be obtained through the use of vocabularies. These benefits are name overloading and object-driven selection of the correct operation once operator names have been overloaded.

Previously I have shown that the selection of operators can be mediated by objects, ultimately through object vocabularies (*Forth Dimensions*, Volume 10, Issue 5). Now, extensions for search-order management are discussed that would support a basic form of multiple inheritance. Management of search-orders through hierarchical object classes is normally a prerequisite. Rather than taking the direct route to constructing these trees, special "search-tree traversal" data structures are proposed to provide object-driven search orders. With these search-tree traversal objects, hierarchies of objects exist only on an ad-hoc, as-needed basis.

PAI Virtuoso — Graphics Interface, Text Interpreter, Memory Management, Relocatable Modules, and Object Orientation Boost Productivity

Lloyd R. Prentice, President
Prentice Associates Incorporated
4 Maple Street, Suite 2
Quincy, MA 02169
(617) 773-2340

PAI Virtuoso™ is a graphics-oriented, Forth-based development environment for creation of highly interactive consumer and educational software, and animated, interactive, desk-top presentations. Virtuoso has contributed more than 10-fold boost to productivity of major software projects.

Programmable Controlled Processing and Graphic System Based on FSY63 Forth

Zuoping Chen
G. Brockmueller
CH-8092 Zurich, Switzerland

The Programmable Controlled Processing And Graphic System which is assembled on a printed board with 16 cm x 14 cm could be used to process the metal elements to any figures of two dimensions. There are libraries of Chinese and English alphabets in the Forth software. Small volume, low price and multiple functions are the advantages of this system.

Pattern Matching in Forth

Brad Rodriguez
T-Recursive Technology
55 McCaul Street, #14
Toronto, Ontario M5T 2W7

The problem of matching ambiguous or partially specified text patterns is a recurring one in computer programming. Specialized languages (SNOBOL4, Icon) have been created to address this problems. This paper describes a set of Forth extensions to match patterns in text strings.

The syntax for patterns resembles that of SNOBOL4. The representation of patterns as Forth words allows nesting, recursion, and indirection. The problem of backtracking through sub-patterns is specifically addressed. The code has been designed to operate on binary data, allowing patterns to be found in any one-dimensional arrays, and not just in text strings.

This project is part of a larger string processing package being written for the IBM PC. 8086 source code for the pattern matcher requires only nine screens.

Control Flow Words from Basis9

Wil Baden
Costa Mesa, CA

Warning: This extract is unofficial and subject to change. It ain't over till it's over.

Languages that can define new functions and procedures are extensible in a weak sense. Forth is extensible in a strong sense because it can define new syntax.

In Forth, discretely defined control flow words are associated with each other to implement the logic for flow of control. What control flow words do must be made clear so that they can be used as components of other words.

(FPC) Forth for the PC

Tom Zimmer
Milpitas, California

F-PC is a greatly enhanced version of Forth derived from the F83 model for the IBM PC, XT, or AT developed by Tom Zimmer of Maxtor Corp. Many other people also contributed to it, including Robert L. Smith, Charles Curley, and Jerry Modrow, but the major work was done by Tom Zimmer.

F-PC is trying to achieve a number of conflicting goals. It is intended to be a Forth system that provides all of the following:

- Compatible to F83 and Forth-83 Standard
- Smooth interface to DOS, using sequencial files
- Fast compilation and execution
- Integrated text file editor for easy program development

- Many utilities and tools
- Room for very large application programs

Logic Stack

C. H. Ting
San Mateo, CA

A separated stack is used to store logic flags to control the execution flow in Forth. Significant savings in stack space can be realized because a flag occupies only one bit on the logic stack. It also forces the separation of logic operations from arithmetic operations and reduces the dangers in mixing logic flags with ordinary numbers. Putting different types of data on different stacks could make Forth a strongly typed language.

A Cross-Assembler for a Small Interactive Target

Robert L. Smith
Lockheed Missles and Space Co.
Palo Alto Research Lab.
Palo Alto, California

The cross-assembler described below was conceived while developing an application for a scientific rocket payload using a small EPROM-based Forth system. In the development phase, the target system is interactive, having **KEY** and **EMIT**. The target does not have **LOAD**. Programs may be downloaded from the host by sending each line of a source block through a serial port to the target. One of the problems in development of the code was the absence of an assembler in the target. It was considered undesirable to add a full assembler to the target because of the additional space requirements. The only other alternative seemed to be to do a meta-compilation and make new EPROMs whenever a change in the code routines was required. Instead, an alternative approach was taken in which a relatively small assembler (about one-tenth the normal size) is used for the target, and the major part of the assembler translation occurs in the host.

A Stack Machine Assembler

Glen B. Haydon
Box 429, Star Route 2
La Honda, CA 94020

Stack machines are the wave of the future. How does Forth fit in to the overall picture of the new machines? It would be a shame for the Forth community to get lost in the trees and fail to see the forest.

CSU Forth
An Object-Oriented Forth Implementation

Ayman S. Abu-Mostafa
California State University
Office of the Chancellor

Object-oriented design concepts can be easily implemented in Forth since the language permits the programmer to extend it by writing new defining words. We have written a Forth interpreter, CSU Forth, which has the object-oriented features built in. No modification or compromise of the standard Forth structures, conventions or philosophy was necessary.

Object-oriented design amounts to three things in the most part: Encapsulation, Message passing and Inheritance. Encapsulation in CSU Forth is done using the defining words **:CLASS** and **;CLASS** which begin and end a *class* definition. The new class becomes a defining word for *objects* of that class. Objects thus created become *message* destinations, i.e., they receive the next word in the input stream as a message. The object CFA matches the message to one of the *methods* defined in its class or to one of its *instance variables*, and executes such method as a regular Forth word. Inheritance is achieved by the word **INHERIT** which is used within the **:CLASS ;CLASS** pair to establish any number of parent classes. This saves having to rewrite methods that are common between classes.

CSU Forth also implements information hiding and assertions to help guarantee proper use and execution of objects.

For Think Forthink Forth Ink

Rob Chapman
Edmonton, Alberta, Canada

One of the things that "Non-Forthies" like to gripe about is that Forth is unreadable with all those mysterious stack operations. The stack presents a learning cliff that some wish not to ascend. Another common gripe is that some words lack clear meaning. To present a solution to these problems, I propose that some of Forth's words are misspelled. In this short paper I propose a better spelling for some of the common Forth words which should make Forth code clearer and easier to follow. This paper is a forml submission to the forthcoming Pooh-Forth standard.

Smart RAM

Rob Chapman
Edmonton, Alberta, Canada

RAM is normally thought of as a dumb slave entity used by microprocessors. However, with increasing chip densities and the simplicity of Forth hardware engines, the concept of coupling the two onto one chip becomes feasible. This is Smart RAM. Smart RAM is explored in this paper with a specific application in mind; use the smart RAM to incrementally and interactive breed a Forth tailored to a microprocessor. Smart RAM will play the role of an emulator.

Thermal Meter
An Application of Forth FSY63 Based on Microcomputer

Zuoping Chen
G. Brockmueller
CH-8092 Zurich, Switzerland

The cost for the heating system is directly proportional to used caloriquantity. The reasonable charge for the customer should be calculated on the basis of the used caloriquantity. With this method by using the cheap MCUs (8-bit microcomputer unit) to measure the caloriquantity of the individual heating radiator is an economical and practical method.

The Harris C Cross Compiler

Tom Hand
Harris Semiconductor
Melbourne, Florida 32802

This paper gives an introductory description of the features that have been incorporated in the Harris C Cross Compiler for the RTX family of 16-bit microcontrollers.

The RTX 2000 is a highly integrated, high performance microcontroller that has been designed for embedded real-time systems. The C Cross Compiler is for developers who prefer the C programming language for implementation of their applications.

The C compiler was designed to meet the proposed ANSI Standard for the C Programming Language. It has a state-of-the-art menu/window interface that utilizes a context-sensitive help facility.

Adding Compiler Security to METHODS>

Ulrich Hoffman
DELTA t GmbH
Uhlenhorster Weg 3,
D-2000 Hamburg 76, FRGermany

This paper describes a simple-to-implement compiler security technique for the even simpler METHODS> concepts, a one-screen object oriented extension to Forth.

Cool - Unifying Class and Prototype Inheritance

Antero Taivalsaari
University of Jyväskylä
Ruovedenkatu 13 D 54
SF-33720 Tampere
Finland
tsaari@tukki.jyu.fi (128.214.7.5) taivalsaari@jylk.jyu.fi (128.214.7.1)

Cool (Coherent object-oriented language) is a new Forth-based object-oriented programming language that unifies two models of inheritance – class inheritance and prototype inheritance (delegation) – to a coherent model in a new way. Cool proves that it is possible to combine class and prototype inheritance without having to lose any major advantages of class inheritance. Cool features clear object structure, data abstraction and possibility for type checking. Yet, from the user's point of view, there is only one kind of object without separate class and instance hierarchies as in most object-oriented languages. Therefore, each object in Cool can function both as a class and an instance.