LISP ENVIRONMENTS

John Foderaro

In this issue we survey the Lisp programming environment provided on the family of Lisp machines built by Xerox. These machines, which once ran only Interlisp-D, are now said to run 'Xerox Lisp' which is a combination of Interlisp-D and Common Lisp.

I asked Pavel Curtis, a member of the Xerox AI Systems Development group and an expert in the Xerox Lisp environment to fill out my questionnaire. His answers are below.

In future issues I'll survey other implementations. I would also like to print your thoughts on programming environments (either short papers, or a 'letter to the editor' on a particular topic). Please send your contributions to the Programming Environments section at the address printed elsewhere in this newsletter.

Xerox Lisp programming environment survey:

Disclaimer: the answers below are the opinion of Pavel Curtis and do not necessarily reflect the opinions of his co-workers nor the official policy of the Xerox corporation.

Background:
Company: Xerox Corporation
Product name: Xerox Lisp
Version of product described: Lyric Release
This version available when: June 1987
Hardware available on: Xerox 1108 (Dandelion), 1186 (Daybreak), and 1132 (Dorado).

Influence:
Derived from the Interlisp-D programming environment which was in turn derived from Interlisp-10, a Lisp implementation for the DEC-10 computer. Interlisp-10 originally called BBN Lisp and was inspired in large part by Lisp 1.5. Xerox Lisp is the name for the new version of this environment.

Primarily residential or file based:
Xerox Lisp originated the term ***residential programming" and could be said to be the quintessential example of it. We are now, however, less pleased with the term because of its vagueness and ambiguity. It is the case that programs are edited almost exclusively as structure and ***source files" are maintained entirely by software and rarely, if ever, directly edited by the user.

Components of the programming environment:
editor:
SEdit, a general structure editor. SEdit is a completely new kind of structure editor, having a very strong orientation toward typing code in the same way as in text editors while simultaneously pretty-printing the structure continuously. While SEdit is heavily optimized for the creation and editing of Lisp programs, the algorithms and interfaces used are not specific to that task. Thus, it is possible to customize SEdit for the editing of non-list structures, such as compiler program-trees. SEdit is entirely window-based and cleanly and thoroughly integrated with the rest of the environment. For example, SEdit is an offered alternative when inspecting list-structures and is invoked by the debugger whenever interpreted code is to be examined or edited. SEdit commands can be invoked either by menu, typing or mouse. SEdit is written entirely in Lisp and runs in the same address space as the rest of the system.

debugger:
The main debugger is window-based and resident in the system. It supports a very easy-to-use menu-based backtrace facility and pops up an attached inspector window on selected frames. Such windows can then be used to examine, inspect, and change the values of variables bound in the frame and to generate an editor on the function associated with the frame. In addition, a window holding the results of disassembling the code can be popped up from the frame inspector; the high-level instruction set of the Xerox Lisp machines makes it easy even for relative novices to understand the code generated by the compiler. Commands to the debugger can be typed or, for the most common ones, selected from a pop-up menu. New commands can be defined by the user. Xerox Lisp implements the proposed Common Lisp standard for error-handling and the debugger provides both menu-based and typing-based interfaces to choosing methods of proceeding from errors and other conditions. There is an extensive facility, called Wrappers, for allowing users to set breakpoints in functions, produce tracing information on calls and returns, and add general ⋙advising″ code before, after or around any other function, compiled or interpreted. This very general mechanism can be used to cause breakpoints to fire when an arbitrary predicate is satisfied or to modularly change the behavior of any function without knowing its internal workings and conventions. Xerox Lisp also includes a network-based remote debugger for those situations in which the user has managed to get the environment into a state in which the normal debugger cannot do its work (such as incorrectly redefining the function that opens windows. This debugger-of-last-resort is rarely used in practice.

large program management:
Xerox Lisp includes the Masterscope tool for analyzing source code and giving the user access to the information it derives. Commands exist, for example, to edit every piece of code using a particular variable or to present a graphical tree showing which functions call which other functions. This tool is considered very important for understanding large, complex programs. There are no tools in the Lyric release for managing sets of files in the manner of the ⋙defsystem″ facility of Zetalisp; a more wide-reaching facility along these lines is planned for a future release.

inspector:
The inspector is a window-based facility allowing the user to view and modify any Lisp structure. Inspectors can either portray the structure in a way closely mirroring its underlying physical representation or can be tailored to specific types, presenting a more abstract view. For example, lists can be displayed and edited with SEdit, shown as numbered sequences, or, if in the form of a property-list or association-list, shown as a structure with fields and values. New kinds of inspectors are easily created for specific data types. Inspectors can be generated from the debugger whenever the user wishes to examine an item on the stack.

documentation:
The Lyric release does not, unfortunately, include up-to-date online documentation. The D-Info/HelpSys system, however, provides a powerful documentation-network browsing facility and, as an example of its use, a complete copy of the Interlisp-D Reference Manual from the previous release. D-Info/HelpSys allows the creation and modification of such online networks of documentation by the user. performance analysis: A number of powerful performance analysis tools are provided by Xerox Lisp. The most commonly-used tool is the Spy, a facility for noticing in which functions time is being spent and for graphically conveying that information to the user. The Spy does not require code to be specially compiled or prepared in any way. The Time facility allows the user to specify a form and receive accurate statistics on the evaluation of that form. For example, Time will describe the CPU usage, elapsed time, paging behavior, and storage allocation and reclamation statistics.

Other tools:
The File Manager runs at intervals and notices any new functions, variables, structures, etc. that have been defined by the user in the environment. It will, if asked, present the user with a summary of what has changed and offer to add the new definitions to given files and to save, compile new versions of any files whose contents have changed. This allows the user to write and modify code without concern for which files contain what entities; new functions can be defined simply by typing DEFUN forms to the Exec or editor. This leads to a very dynamic and immediate style of programming in which there is no artificial barrier between the code being written and that currently running on the system.

The Exec is a command processor accepting several syntaxes for specifying Lisp forms to evaluate and a facility for the definition and invocation of new commands with flexible syntax. The Exec provides a complete history system for remembering the commands and forms that have been executed and for redoing or modifying and evaluating previous events. In addition, a general, user-extensible undoing facility is in place, allowing most events to be undone, including such major changes as the loading of a file.

The Process Status Window allows monitoring of the current states of all running processes in the system. Menu items exist for seeing a current stack backtrace for any process, controlling which process is to receive keyboard input, and suspending, waking, killing and restarting a selected process.

There is a rich library of other facilities available, many supported by Xerox and others contributed by users of the system, far too many to go into here. Talk to a Xerox Lisp sales office for real details.

Non-window environment:
Xerox Lisp provides for only one active environment on a given machine at a single time. While there are facilities for accepting network connections and providing an Exec, the debugger and a non-display-oriented editing environment, it cannot be recommended for serious program development. It is, however, useful for monitoring activities on a remote machine and for some kinds of remote debugging.

The following is a Common Lisp program that computes a relatively familiar function in a relatively unfamiliar way. It has been written with uninformative variable names. The idea is to figure out what the program does without running it.

This one uses one of the fundamental principles of algorithm design to produce a version of a known algorithm that happens to be better parallelized (if that's what you want to do). A surprising fact is that in some implementations of Common Lisp, this version runs faster than the original algorithm.

```
(defun f (n)
 (labels ((f (n m)
           (if (= n m)
           n
           (let ((h (truncate (+ m n) 2)))
          (* (f n h) (f (+ h 1) m))))))
   (f 1 n)))
```

Dick Gabriel