Programming Environments

John Foderaro

We continue our survey of Common Lisps with a report on the environment of Extended Common Lisp from Franz Inc. of Berkeley, California. Our previous surveys were for Lisps on dedicated Lisp Machines. Extended Common Lisp is designed for general purpose machines. I wish to thank David Margolies of Franz Inc. for completing the survey.

If you have comments on the surveys or thoughts on programming environments for Lisp, please mail them to me at the address given in the inside cover of this journal (either electronic or paper is fine).

Background:

Company: Franz Inc.
Product name: Extended Common Lisp
Version of product described: 2.0, available now;
3.0 will be available in the Fall.
Hardware available on: Sun 2, Sun 3, Vax (4.2 BSD Unix or DEC Ultrix), Masscomp, Mac II. Earlier release available on Tektronix 440x, ISI, Silicon Graphics, etc.

Extended Common Lisp is designed to run on general purpose hardware and thus runs within many different programming environments. Most people run Extended Common Lisp under the Unix operating system, the premier program development operating system. They communicate with Unix (and thus Lisp) either with a simple terminal or through a window system.

Version 2.0 of Extended Common Lisp was designed to be controlled through a simple terminal interface. This is a universal interface and also allows Lisp to be used easily from within Emacs. Version 3.0 will provide window-based interfaces to the editing, debugging, and performance monitoring features of Lisp.

We waited until version 3.0 to start using windows because we were wanted to develop our tools in a standard window system (the community certainly doesn't need a different window system design by each Lisp vendor). At this point it appears that for general purpose machines the standard window system will be either X (developed primarily by MIT and DEC) or NeWS (developed by Sun Microsystems) or some combination of the two. For the Lisp interface to the window system, many Lisp application developers prefer Intellicorp's Common Windows specification and we are working with Intellicorp and Sun on extending the specification so that it handles color and works well with networked window systems such as NeWS and X.

Influence:

The primary influence of the version 2.0 terminal based interface is the Unix C shell. Extended Common Lisp has a 'history' facility for remembering all user commands. One can search for and execute previous commands based on characters in the command or on the command number. The main read-eval-print loop (which we call the 'toplevel') treats input preceded by a command character (by default the colon) as an instruction to run a toplevel function. For example, typing

:cf foo/bar

has the same effect as typing

(compile-file "foo/bar")

The user can easily add his own commands to the set of top level commands.

We were also influenced by the Smalltalk idea of a modeless system. Our toplevel has an integrated stack inspector, trace package, stepping package and object inspector. All components of the toplevel are available all the time. On some systems when you get into an error break, certain commands are available, and when you are in the inspector other commands are available. Thus what you can type depends on the mode you are in at the time. We wanted to avoid such a system. Thus all toplevel commands (and your history list and your private set of toplevel commands) are available to you whether you are in a error break, a trace break, a step break, or in the toplevel loop itself.

Our version 3.0 window based interface is based on the Interlisp-D interface. This is partially due to the fact that the Common Windows window system is based on the Interlisp-D window system and partially due to the fact that people praise the Interlisp-D environment.

Primarily Residential or File Based: file based.

Components of the Programming Environment:

editor: Users select from the editors available on their machine. Many programmers use a version of Emacs such as Gnumacs, an Emacs editor from the Free Software Foundation or Unipress Emacs. The advantage of these two versions of Emacs is that lisp can be run within an Emacs buffer.

debugger: As mentioned above, in version 2.0, the debugger is terminal based. In 3.0 it will be window based, with a window produced when a break level is entered. For both 2.0 and 3.0, there are a set of stack inspection commands which permit the user to browse the stack, examining and changing local variables, and then return from any part of the stack or restart the computation at any part of the stack.

inspector: In 2.0, a the inspector is terminal based. In 3.0, it is window based, with an inspect window appearing when the inspector is called, with slots and values listed. Mousing on a value allows that value to be inspected, mousing on a slot allows the value of the slot to be modified.

documentation: Many, but not all, functions have online documentation. The user may document user-created functions. For hardcopy documentation, there is a User Guide, describing the implementation and its extensions, and a Reference Manual, listing all functions in alphabetical order.

Other Components:

There is a profiler which can either sample the actual time spent in each function or else count the number of calls to each function.

Extended Common Lisp has a stack group/multiprocessing facility compatible with that found on Lisp Machines. This is important for window system code where the stack is used to hold computational context for one window when another window is active.

Common Windows, based on the specification by Intellicorp, also allows users to customize their environments. Both stack groups and Common Windows are available in version 2.0.

Non-window Environment:

In version 2.0, the complete programming environment is available without the need for a window system. Users typically use Emacs to create a little terminal-based window system. In version 3.0 the window-based environment will be easier to use than the non-window environment but it will still be possible to use Lisp from a simple terminal (i.e. nothing will be removed from the terminal-based environment, things will simply be added to the window based environment).

(Unix is a trademark of AT&T, Interlisp-D is a trademark of Xerox Corp., NeWS is a trademark of Sun Microsystems).