Programming Environments

John Foderaro

The Franz Inc. ALLEGRO CL/GNU EMACS Interface

John Foderaro and D. Kevin Layer

1. Introduction

The editor is an important part of a LISP programming environment. In this paper we describe how we have integrated the GNU EMACS editor with ALLEGRO CL (previously known as Extended COMMON LISP). Parts of the interface between ALLEGRO CL and GNU EMACS can be used with any version of LISP, including FRANZ LISP. Other parts require a sophisticated multiprocessing facility, such as the one found in ALLEGRO CL or on LISP-machines. The interface can be obtained at no charge as described at the end of this paper.

2. Choosing the Editor

Our goal was to choose an appropriate editor for ALLEGRO CL running on general-purpose UNIX workstations. Our choices were:

1. write an InterLISP-style structure editor,
2. write an in-LISP EMACS-style text editor,
3. or use an existing editor for general-purpose workstations.

We ruled out an InterLISP-style structure editor because ALLEGRO CL is not a residential LISP in the InterLISP style. A structure editor would still be useful for examining and modifying objects and we may write one in the future, but we do not see ourselves using one as the primary program editor.

We were then faced with choosing between writing an EMACS-like editor in LISP and using an existing editor. We chose GNU EMACS, currently the most widely used EMACS editor on general-purpose UNIX workstations, for these reasons:

- Most people prefer to learn just one editor. Aside from not wanting to learn and remember the conventions of more than one editor, many people expend great effort customizing their editor environment.
- GNU EMACS is a comprehensive, well-designed editor. Duplication of all of its features would be a difficult task and wasted effort.
- On a general-purpose machine, developers want a general-purpose editor since they will be using that editor not only for editing COMMON LISP programs but for reading and processing mail, writing papers using TeX or *troff*, and editing programs in other languages such as C or Fortran. Also, the editor should work on whatever terminal or bit-mapped display is available.
- The editor must be reliable. In the model where LISP and EMACS share the same address space (i.e., EMACS is implemented in LISP), changes made to files or buffers can be lost if LISP dies. (It is not normal for LISP to die, but buggy foreign functions and *unsafe* user-written programs can cause any LISP to fail unexpectedly.)

3. GNU EMACS

GNU EMACS is a portable and extensible text editor written by Richard Stallman, with contributions by many others. It is written in C and EMACS LISP (Elisp) and runs on most machines running the UNIX operating system. Elisp, the extension language, is similar in many ways to COMMON LISP, and includes a LISP-to-portable-byte-code compiler and garbage collector.

ALLEGRO CL is a trademark of Franz Inc., Macintosh is a trademark licensed to Apple Computer, and UNIX is a trademark of AT&T.

EMACS is extensible at a very basic level: there is a table that maps every key on the keyboard (and some key sequences) to a function. When you press a key, a function is executed which defines the behavior of the key. For example, with the initial key bindings, when you press the 'a' key, a function called **self-insert-command** is executed that inserts an 'a' into the buffer at the location of the cursor. Another feature of EMACS is that it permits you to edit more than one file at a time, each file being stored in its own buffer. Each buffer has a local mapping of keys to functions, thus you can alter the behavior of EMACS in each buffer. For example, in a mail-reading buffer pressing the 'a' key may mean 'answer the mail', and the function executed would set up a buffer in which to write a reply to the current letter.

The set of key to function bindings in a buffer is called the *mode*. In our interface we define several modes. One is COMMON LISP mode which makes it easier to write COMMON LISP programs. For example, in COMMON LISP mode there is a function, or key sequence, that displays the argument list for a given symbol by querying the LISP environment. There is also a mode for interacting with a LISP process, and another for having multiple LISP listeners connect to the *same* LISP.

4. Features of the interface

The parts of the interface include:

- A mode for editing LISP programs. Expressions are automatically indented as they are typed. Matching parentheses are highlighted with the cursor. This mode is an enhancement to the standard LISP mode provided by GNU EMACS, and provides features not available in the standard LISP mode.
- A mode for interacting with a LISP process. The LISP process can be running on any machine on the network. Expressions are indented when they are typed and matching parenthesis highlighting is done as in LISP editing mode. Previous typed input can be edited and re-executed. Previous lines of input can be recalled based on a search pattern.
- A mode that combines the first two. While editing a file you can request an argument list or description of a symbol or function (*meta shift a*), the result of a macroexpansion (*meta shift m*), or to see the source associated with a function, defstruct, or variable (*meta dot*). The EMACS part of the interface accomplishes this by sending information to and receiving information from LISP via a *back door* (explained below). When the response is received from LISP, EMACS displays the information requested by the user.
- A mode that simulates a residential or InterLISP-like environment. Instead of having to recompile and load entire files, this interface allows single functions or a group of functions to be compiled and loaded into the LISP environment. This merging of the EMACS and LISP worlds regains what is lost in the file-based approach to LISP development, and makes the EMACS and LISP seem like they inhabit the same world (like an in-LISP EMACS would).
- Other features, such as an improved *Shell* mode (for issuing commands to the operating system) and a function for creating multiple separate LISP listeners onto the same LISP, using the multiprocessing feature of ALLEGRO CL.

5. Implementation

The modes for editing LISP programs and talking with a LISP process are written in entirely in Elisp (EMACS LISP). They are important parts of the interface but there is nothing worth noting here.

For the rest of the interface, which deals with the interaction between a running LISP and EMACS, the global view is that there is a LISP process running in a separate UNIX process under the control of EMACS. In other words, the characters that LISP writes to ***terminal-io*** are sent to EMACS and what EMACS sends to LISP appears as input in ***terminal-io***. LISP runs in a named EMACS buffer, called, for example, **common-lisp**. The user types to EMACS in the *common-lisp* buffer, which then becomes input to the LISP process. When LISP produces output, it is read by EMACS and put in the *common-lisp* buffer.

Consider now the case of a user running EMACS with a connection to LISP. Say, for example, the user wants to describe (in the COMMON LISP sense) the COMMON LISP function **make-package**. The key sequence "M-D make-package RET" (*meta shift d*, followed by *make-package*, followed by a carriage return), in a COMMON LISP moded buffer (as opposed to, say, a mail moded buffer), is the way to request that this information be displayed by EMACS. If the user were just talking with COMMON LISP, the form to evaluate to get this information would be:

    (describe 'make-package)

EMACS could just send the above sequence of characters to the COMMON LISP process, but there would be no guarantee that the COMMON LISP process was in a state where a reply would be possible (the user may be running an application). For this reason, there needs to be a dedicated communication link between EMACS and COMMON LISP. In our interface, this *link* is referred to as the *back-door LISP listener*. It is called *back door* because the user never actually sees it or types to it. *LISP listener* is the conventional name for a read-eval-print loop to which a user of LISP types. Using this terminology, the LISP process connected to a buffer (the one to which the user types) could be called the *front-door LISP listener*, or just *front-door*.

The back-door communication between EMACS and ALLEGRO CL is at the heart of the interface. While the user is interacting with ALLEGRO CL through the front door, EMACS has a communication channel open to ALLEGRO CL through the back-door. This back-door is just another listener which EMACS uses to query the Lisp environment for information pertaining to, for example, argument lists, macro definitions, documentation strings of functions, etc. It uses the networking primitives available on Berkeley UNIX, which allow communication over UNIX or Internet domain sockets. This use of *two* listeners to the same LISP is possible because of ALLEGRO CL's multiprocessing and stack group facilities. (LISP machines have used stack groups and multiprocessing for some time as the basis for their multi-tasking operating systems. These features have been available in some versions of ALLEGRO CL for almost two years.)

6. Problems in the interface

One problem we faced was selecting the appropriate network address for LISP to monitor for messages from EMACS. If we chose an Internet domain port then only one user on a machine could use the interface unless each user chose a unique port number for LISP-EMACS communication. We support both styles of network addressing, but decided to use, by default, a UNIX domain port (with the address being a file in the user's home directory, thus providing a unique address for each user). One problem with using a UNIX domain address is that EMACS and LISP must be running on the same machine. Another problem with using UNIX domain addresses is that the standard GNU EMACS we started with (version 18.50) did not have support for UNIX domain sockets. It was a straight-forward task to extend **open-network-stream** in GNU EMACS to allow the creation of UNIX domain sockets and this modification has been sent to the maintainers of GNU EMACS and made part of the interface release.

7. Future work

Currently EMACS drives the interface and LISP acts as a slave. We will expand the interface to permit LISP programs to request services from EMACS (e.g. edit files or expressions, pop up buffers, etc). We will also generalize the interface and define in EMACS an 'eval in LISP' function and define in LISP an 'eval in EMACS' function, thus permitting users to extend the interface.

8. Conclusion

General-purpose machines are popular because they run a variety of applications, often written in many different languages. But, if each application provides its own editor it will be unpleasant for the user. The Apple Macintosh is currently the best example of a system that provides a uniform interface across applications, and it is second to none in ease of use.

On UNIX machines we see EMACS as the best solution to providing a uniform application environment. GNU EMACS, in particular, is much more powerful and extensible than the Macintosh 'point and click' interface and the subprocess control facilities of GNU EMACS already provide a uniform interface between editing files and interacting with the UNIX command parser (the shell). With our LISP-EMACS interface we add LISP to the set of programs tightly coupled to EMACS.

9. How to obtain GNU EMACS and the EMACS/LISP interface

You can obtain GNU EMACS by copying the latest distribution version of EMACS from the file */u2/emacs/edist.tar* on host prep.ai.mit.edu (or the file */u2/emacs/edist.tar.Z* which has been run through compress after tar). These files are about 7 and 3 megabytes long, respectively. After you unpack the distribution, be sure to look at the files *README* and *INSTALL*.

To obtain the current release of the EMACS/LISP interface, either:

1) if you have Internet access, copy it from ucbarpa.berkeley.edu or ucbvax.berkeley.edu via FTP (login *ftp*, password your login name) from the directory pub/fi/gnudist-1.2-tar.Z, or

2) send a check (sorry, no PO's accepted) in the amount of $50 for a US address or $75 for a foreign address to Franz Inc. to the following address:

   Franz Inc.
   Attn: LISP/EMACS Interface Request
   Suite 275
   1995 University Ave
   Berkeley, CA 94704

   Please specify the media (*tar* format only) which is one of:

   - 1/2", 1600 bpi, 9-track
   - 1/4", cartridge tape—specify the machine type (e.g., Tektronix 4300)

There is a mailing list for the discussion of issues related to this interface, called 'lisp-emacs-forum@Berkeley.EDU'. If you would like to be on it, send mail to 'lisp-emacs-forum-request@Berkeley.EDU'.