Chris Mennie – 97024327
Troy Gonsalves – 97083748

CS746 Assignment 1C – Top Level Architecture of Emacs

January 23rd 2003

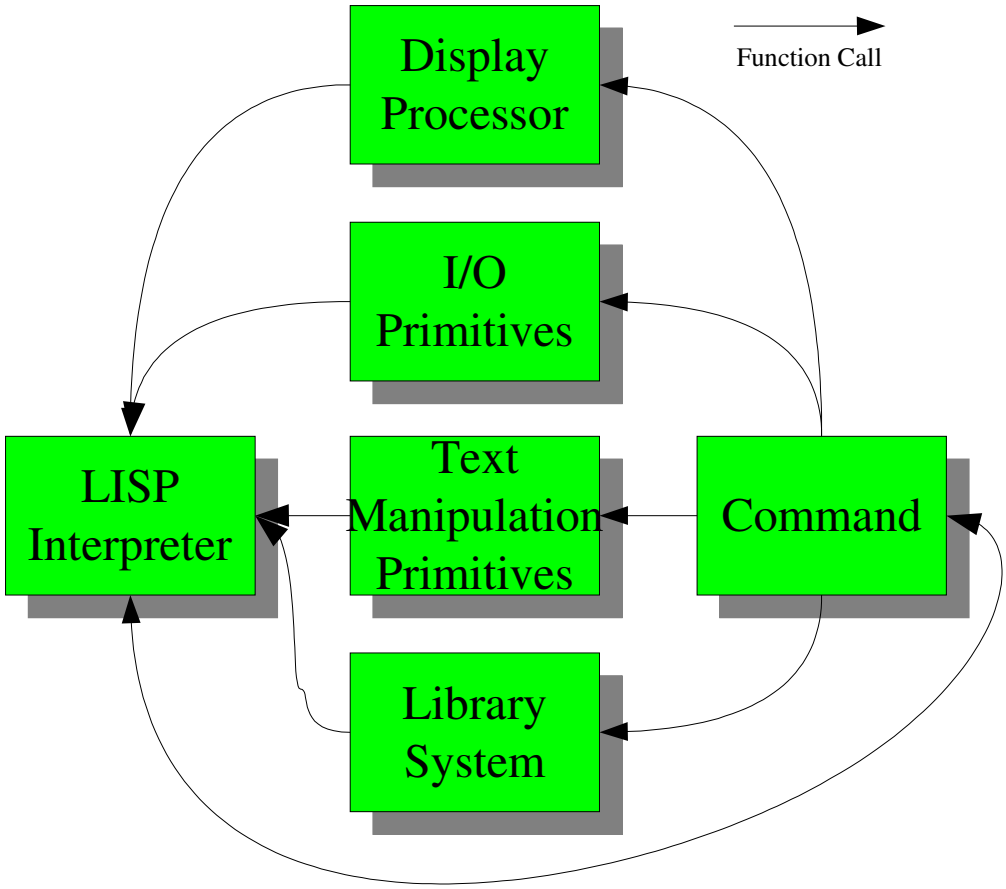
1.Overview

The overall architecture of EMACS is that of a simple text editor and LISP side-by-side with hooks into each other. Rather than statically implementing editing functionality, as most editors do, the approach taken in EMACS is more flexible. EMACS allows LISP functions to be called and perform operations such as manipulating the text buffer. The power of EMACS derives from the integration of LISP into the text editor. This allows users to dynamically bind new functionality at run time, overriding old functionality or providing new functionality.

2.Subsystem Descriptions

While EMACS consists of two main components, each of these components contain a number of subsystems. The following is a diagram of the system, each subsystem is described below.

High-Level Architecture of EMACS



2.1.LISP Interpreter

At the heart of EMACS is a LISP interpreter through which most of the interesting editor functionality is performed. The LISP interpreter uses the Command subsystem to access and manipulate the other components in the system. Thus, when a user writes a script to extend EMACS the user encodes calls to the Command subsystem. At runtime these calls are interpreted and placed by the LISP interpreter.

2.2.I/O Primitives

This subsystem allows the user to perform necessary I/O operations such as loading or saving a file. Some of the I/O functions will be implemented in LISP.

2.3.Text Manipulation Primitives

This subsystem allows the user to manipulate the text buffer. For example, text searching or replacement would be dealt with here. As with I/O Primitives subsystem, some functions will be implemented in LISP.

2.4.Library System

The Library System maintains sets of function names, definitions, and documentation, which may be loaded dynamically at any time. Multiple function sets may be loaded at a time, with duplicate names allowed. This provides the means to override previously defined functions if desired. Most of the standard EMACS functions are implemented in this way, allowing the user to change the default behaviour of the system.

2.5.Command

The Command subsystem is responsible for maintaining two mappings. The first is the key-binding map, which maps bound-key sequences to functions. The second mapping is from function names or aliases to actual functions. Using these mappings the user may invoke a function by name, or through a given key press. The most common functions are bound to key presses, while the least common require explicit invocation (i.e. by function name). The command dispatcher is also responsible for obtaining arguments from the user, via the terminal interface.

Some of the functions in the mappings mentioned above will be LISP functions. Hence the Command subsystem must access the LISP interpreter.

2.6.Display Processor

This subsystem maintains the on-screen display and internal representation of the edited text. It acts as a low priority task which updates the visible buffer whenever no other task (even editing) is being performed. The intent here is to increase the overall performance of EMACS. As with other components of the system, some functions may be implemented in LISP.

3.References

[EMACS] EMACS: The Extensible, Customizable Display Editor, http://www.gnu.org/software/emacs/emacs-paper.html