Emacs avoids this kind of problem by taking the easy way out: it doesn't support style sheets, automatically numbered sections, headers or footers, tables, or any number of other features expected from modern word processors. It's purely a text editor, and an Emacs buffer is just a string of characters. In almost every circumstance, all the buffer state that matters is readily apparent. As a result, one is rarely confused over what Emacs has done to the contents of one's files.

How complex is the command set?

How easy is it to discover the actions that are relevant and useful at any given point? How easy is it to discover features one hasn't used yet? In this sense, Emacs's user interface is quite complex. A freshly started Emacs session without customization recognizes around 2,400 commands and 700 key bindings, and will usually load many more as the session goes on.

Fortunately, a new user need not confront this horde all at once, any more than a Unix user needs to learn every shell command. It's perfectly possible for a new user to treat Emacs like any other editor with a graphical user interface, selecting text with the mouse, moving the cursor with the arrow keys, and loading and saving files with menu commands. Commands left unused don't impinge on the visibility of essential functionality.

However, using Emacs in this fashion isn't much better than using any other editor. Becoming a proficient Emacs user entails reading the manual and the online documentation, and learning to search through these resources effectively. Features such as *grep* and compilation buffers, interactive debugging, and source code indexing are what distinguish Emacs from its peers, but they don't reveal themselves unless you explicitly request them—which you wouldn't do unless you already knew they existed.

To make this kind of exploration a little easier, Emacs also includes the *apropos* family of commands, which prompt for a string or regular expression, and then list commands and customization variables with matching names or documentation strings. Although they're no substitute for reading the manual, the *apropos* commands are effective when you have a general idea what you're looking for.

In terms of this kind of complexity, the Emacs user interface has many of the characteristics common to command-line interfaces: it's possible to have many commands available, the user need not know all of them (or even many of them), and it takes deliberate effort to discover new functionality.

Fortunately, the Emacs community has been effective at establishing conventions that commands should follow, so there's a good deal of consistency from one package to the next. For example, almost all Emacs commands are modeless: the standard commands for moving, searching, switching buffers, rearranging windows, and so on are always available, so you needn't worry about how to "exit" a command. As another example, most commands use standard Emacs facilities to prompt the user for parameters, making prompting behavior consistent from one package to the next.