Productivity Engineering in the UNIX† Environment

Pan I: An Introduction for Users

Technical Report

S. L. Graham
Principal Investigator

(415) 642-2059

"The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Defense Advanced Research Projects Agency or the U.S. Government."

Contract No. N00039-84-C-0089
August 7, 1984 - August 6, 1987

Arpa Order No. 4871

†UNIX is a trademark of AT&T Bell Laboratories
Pan I
An Introduction For Users

Robert A. Ballance
Michael L. Van De Vanter

Computer Science Division
Department of Electrical Engineering and Computer Science
University of California
Berkeley, California 94720

September, 1986
Revised September, 1987

PIPER Working Paper 87-5

Abstract

Pan is a prototype and testbed for language-based editors and viewers. Its design addresses the needs of experienced users who manage complex objects such as large software systems. All of Pan's components are multi-lingual, incremental, description-driven, customizable, and extensible. Viewing is facilitated by semantics-based browsing and an object model which integrates text and structure. Pan is intended to share information with other tools, allowing integration into a larger language, program, and document development environment.

This document, a users manual, describes the basic operational facilities of Pan I, the current implementation. It explains the concepts behind Pan's editing environment, introduces editing commands, and discusses techniques for customization. Appendices list command bindings (to both keystrokes and menus), buffer options, buffer flags, and a compatibility guide for GNU Emacs users.

1Sponsored by the Defense Advanced Research Projects Agency (DoD), Arpa Order No. 4871 (monitored by Space and Naval Warfare Systems Command under Contract No. N00039-84-C-0089), by IBM under IBM Research Contract No. 564516, and by the State of California MICRO program. Robert A. Ballance and Michael L. Van De Vanter were supported in part by MICRO fellowships.
Contents

1 Introduction 1

2 Concepts 1
  2.1 Buffers ........................................... 1
  2.2 Viewers ........................................... 2
    2.2.1 Frames ........................................ 3
    2.2.2 The Information Panel ......................... 4
    2.2.3 The Editing Area ................................ 5
    2.2.4 Font Maps ...................................... 5
  2.3 Bindings ........................................... 5
    2.3.1 Key Bindings .................................... 5
    2.3.2 Menu Bindings .................................... 6
    2.3.3 Operand Level Bindings .......................... 7
  2.4 The Edit Cursor .................................... 7
  2.5 Regions and The Selection ........................... 7
  2.6 Rings .............................................. 8
    2.6.1 The Clipboard ................................... 8
    2.6.2 Kill Rings ....................................... 8
  2.7 Marks and the Mark Stack ............................ 8
  2.8 Syntax Classes ..................................... 9
  2.9 Options, Flags, and Variables ....................... 9
  2.10 Communication with Other SunView Clients ............ 9
  2.11 What's Where? .................................... 9

3 Editing with Pan 10
  3.1 Getting Started .................................... 10
  3.2 Quitting Pan ....................................... 10
  3.3 Suppling Arguments to Commands ...................... 11
    3.3.1 Numeric Prefix Arguments ....................... 11
    3.3.2 Prompts and Pop-Ups ............................ 11
  3.4 Aborting Commands .................................. 11
  3.5 Undoing Actions .................................... 12
  3.6 Getting Help ....................................... 12
  3.7 Buffers and Viewers ................................ 13
    3.7.1 Visiting Files and Buffers ....................... 13
    3.7.2 Saving and Writing Files ....................... 14
    3.7.3 Manipulating Viewers ............................ 14
  3.8 Scrolling ........................................... 15
  3.9 Cursor Motion ...................................... 16
    3.9.1 Mark Commands .................................. 16
  3.10 Editing Text ...................................... 17
    3.10.1 Setting the Operand Level ...................... 17
## Pan I: An Introduction For Users

3.10.2 Setting the Selection .................................................. 17
3.10.3 Inserting Text .............................................................. 18
3.10.4 Filling Text ................................................................. 19
3.10.5 Deleting Text ............................................................... 19
3.10.6 Killing Text ................................................................. 20
3.10.7 The Clipboard .............................................................. 20
3.10.8 Copying and Moving Text ............................................... 21
3.10.9 Commands for Changing Case ......................................... 21
3.10.10 Transposing ............................................................... 22

3.11 Searching Text ............................................................... 22
  3.11.1 Regular Expressions ................................................ 22
  3.11.2 Balanced Bracket Commands ........................................ 22

3.12 Editing Programs .......................................................... 23
  3.12.1 Language-Oriented Viewers .......................................... 23
  3.12.2 Selection .................................................................. 23
  3.12.3 Navigation .................................................................. 25
  3.12.4 Parsing and Syntactic Errors ........................................ 26
  3.12.5 Editing .................................................................. 27
  3.12.6 Displaying Trees using Ptree ....................................... 28

3.13 When Things Go Wrong .................................................... 28

4 Simple Customization ......................................................... 28
  4.1 Start-Up Processing .......................................................... 29
  4.2 Bindings .................................................................. 29
  4.3 Getting and Setting Option Values ...................................... 30
  4.4 Lisp-Oriented Commands ................................................ 31

5 Acknowledgments ................................................................. 31

References ............................................................................. 31

6 Glossary ................................................................................ 32

A Default Key Bindings .......................................................... 34
  A.1 Bindings By Command Name ............................................. 34
  A.2 Bindings By Key ............................................................... 36

B Default Menu Bindings in a Text Buffer .................................. 38
  B.1 Bindings by Command Name ............................................. 38
  B.2 Bindings by Menu ........................................................... 38

C Default Menu Bindings in the Base Buffer ................................ 39
  C.1 Bindings by Command Name ............................................. 39
  C.2 Bindings by Menu ........................................................... 40
### D Default Menu Bindings in the Help Buffer
- D.1 Bindings by Command Name ........................................... 40
- D.2 Bindings by Menu .................................................. 41

### E Options Defined in Pan .................................................. 41

### F Flags Defined in Pan .................................................. 46

### G Pan for GNU Emacs Users ............................................. 48
- G.1 Key Bindings .......................................................... 48
- G.2 Menus ................................................................. 53
- G.3 Undo ................................................................. 53
- G.4 Operand Level ........................................................ 54
- G.5 Cut/Paste/Kill/Yank ................................................... 54
- G.6 Options ............................................................... 58
- G.7 Help ................................................................. 58
- G.8 Special Editing Modes ............................................... 59
1 Introduction

Pan is an editor for text- and tree-structured documents that uses the mouse, menus, and multiple windows to provide "cut and paste" editing. The system runs on Sun Workstations under UNIX and Release 3.2 (and later) of SunView.

The text-oriented facilities of Pan are modeled on the Emacs family of text editors. Users familiar with an Emacs-style editor will have little trouble learning Pan. Pan is extensible and customizable in the spirit of Emacs[6].

Pan also provides for manipulating and editing programs using the syntax and semantics of the language being edited. The current version of Pan uses only information about the syntax of the language; the component that uses semantic information is now under development.

This document is an informal introduction to Pan I. It contains essential information for editing with Pan. To extend the system, though, you'll want the information in The Pan Extension Manual[2]. Until that manual is available, you may want to contact the Pan group (panpipes@sequoia) directly. For more general background on Pan, consult The Architecture of Pan I[3].

Throughout this document, "Pan" refers to Pan I, the prototype implementation. Pan II, a major revision of Pan I, is being designed. Your constructive comments on Pan I will help us to provide a better environment with Pan II. Let us know what you think!

2 Concepts

This section is a brief introduction to the terminology and notation of Pan. Since many of the concepts differ in meaningful ways from similar notions in Emacs, it bears careful attention. Appendix G compares the text-oriented concepts and facilities of Pan with those supported by GNU Emacs[7]

2.1 Buffers

A buffer is the focus of editing attention for a single object. (A single object is a text file in the current version.) During editing, Pan manages several buffers.

Along with the object being edited, each buffer has

- one or more viewers,
- a single visible selection,
- key, menu, and operand-level bindings,
- an operand-level setting,
- option values,
- flag values,

---

\[2\text{Sun Workstation, and SunView are registered trademarks of Sun Microsystems, Inc.}\]
\[3\text{UNIX is a registered trademark of AT&T Bell Laboratories.}\]
Pan I: An Introduction For Users

- a mark stack,
- a kill ring, and
- definitions of character syntax classes.

The viewers associated with a buffer share the selection, bindings, options, and other values owned by that buffer.

All buffers in Pan are named. The name of a buffer is the name of the object being edited in that buffer. In the current release, the name is the name of the file being edited.

The base buffer and the help buffer are special buffers maintained by the system. The base buffer is your doorway to Pan: within its edit window are the names of the buffers being edited. As new buffers are created, their names are added to this list.

The base buffer differs from other buffers in two ways: only the base buffer contains Quit in its frame menu and only the base buffer can be made iconic. The base buffer is always visible on the screen, perhaps as an icon.

The help buffer is used for displaying help and other information. See Section 3.6 for more about help facilities.

Most commands act on the active viewer of the active buffer. The active viewer is the viewer in which the most recent keystroke or mouse action occurred. The active buffer is usually the buffer that owns the active viewer.

Figure 1 shows the screen of a workstation running Pan. The base buffer appears in the upper left corner; it lists two other buffers, the help buffer and a buffer named "manual.tex". Buffer manual.tex (containing a file of the same name) is currently active and has two separate viewers open onto it. The help buffer, visible in the upper right corner, currently shows the bindings in effect for the active buffer.

2.2 Viewers

Each buffer has one or more viewers—independent windows onto the contents of the buffer. A viewer provides a display mechanism, scroll bars, a message line, and an edit cursor.

Viewers exist independently from the SunView frames (see below) in which they are displayed. Thus there can be buffers and viewers that are not visible on the screen. When visible on the screen, viewers can be manipulated like any SunView window.

A buffer retains its viewers even when the viewers are not visible on the screen. All of the viewers opened onto a single buffer share that buffer's contents: its bindings, its option and flag values, and its selection. However, each viewer has its own independent edit cursor.

Viewers onto textual objects (the only kind supported in the current version) display text as if it were an infinite quarter-plane of characters, with newlines separating each line. Rather than wrapping lines when they reach the right-hand edge of the viewport, the lines appear truncated. The horizontal scroll bar, and horizontal scrolling commands allow you to see text to the right (or left) of the current edit window.

A viewer is partitioned into three areas: the frame, the information panel, and the editing area.
2.2.1 Frames

The frame surrounding a viewer responds to the normal SunView protocols[1]. Those protocols can be used to independently position and size the viewer. Figure 2 shows a viewer with its frame menu exposed.

The frame associated with the base buffer is the only one containing the items Close and Quit in its frame menu. When Close is chosen, all of the visible viewers disappear from the screen and an icon for Pan appears. Opening the icon reopens the other viewers. When Quit is chosen, you will be offered the chance to save any modified buffers before exiting. This is the normal way to terminate an editing session.

Frames other than the frame associated with the base buffer have the Done menu item in their frame menu. Selecting this item causes its viewer to disappear from the screen. Internal state of the viewer is retained even when the viewer is not visible.

<table>
<thead>
<tr>
<th>Frame menu items</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Close</td>
<td>Close all viewers</td>
</tr>
<tr>
<td>Quit</td>
<td>Save any modified buffers before exiting</td>
</tr>
<tr>
<td>Done</td>
<td>Cause the viewer to disappear</td>
</tr>
</tbody>
</table>

Figure 1: Screen image of Pan showing base, help, and text buffers
Due to limitations of SUNVIEW, only six frames are available to Pan. This means that you can only have six Pan viewers (including the base buffer) visible at any time. Frames are allocated to a viewer when the viewer is opened, and are deallocated when the viewer is closed.

2.2.2 The Information Panel

Inside the frame, at the top of the viewer, is a panel containing information about the object being viewed. This includes the buffer name, the language being edited, the operand selection level, a message line, and the values of various flags. Viewers onto the help buffer contain only a title and a message line.
2.2.3 The Editing Area

Below the information panel is the editing area. This area includes horizontal and vertical scroll bars, and the actual edit window. Pressing the right button of the mouse over the edit window of the viewer activates the Pan menus for that buffer. Selecting an item from those menus executes the command bound to that selection.

2.2.4 Font Maps

Every character in Pan's internal text representation contains a font code. Each viewer has an associated font map, which associates font codes with internal font descriptors. A font map contains from 1 to 16 entries.

Internally, fonts are referred to as "font 0", "font 1", etc. The first element of the list (for: 0) is the default font; unspecified font codes revert to the default font.

Pan maintains three default font maps: one for normal viewers, one for the base buffer, and one for the help buffer. You can alter these defaults by setting the options :text-font-map, :base-frame-font-map, and :help-frame-font-map respectively. The standard specification for a font map is a zero-indexed list of font names containing from 1 to 16 names. For example, the default value of the option :text-font-map is

("screen.r.12" "screen.b.12" "serif.r.12" "cour.r.12" "cour.b.12")

Currently, Pan offers only limited support for manipulating font codes in the text representation. Standard text viewers use only the default font. Section 3.12.4 describes a more elaborate use of font maps in conjunction with tree-structured documents.

2.3 Bindings

A binding associates a sequence of keyboard or mouse actions with a command. Key bindings associate keystroke sequences with commands; menu bindings associate menu item selections with commands. Operand level bindings associate generic operations, such as "next" or "delete" with operands designated by the current operand level (section 2.3.3).

Key and menu bindings are either local to a buffer or are global to all buffers. Every buffer may have its own set of key bindings, and even its own menus and menu selections. Naturally, local bindings take precedence over global bindings. Operand level bindings are local to each buffer: there are no global defaults for them.

2.3.1 Key Bindings

Pan, like Emacs, provides a live keyboard. Keystrokes (including mouse buttons) are read until a valid binding is detected. When a binding is detected, the associated command is executed.

A key binding associates a 1- or 2-character keystroke sequence with a command. The keys Escape, ^X, ^C, and ^Z are reserved to be prefix keys in two-keystroke bindings. The Shift and Control keys are modifier keys rather than prefix keys.

Function keys and mouse buttons can be mapped just like the standard keyboard keys. In fact, it is the standard binding of the right-most mouse button to Execute-From-Menu that implements Pan's menu selection service.
The default bindings for the left-hand function keys (L1-L10) reflect standard SUNVIEW usage when appropriate. The default bindings for mouse buttons are similar to the SUNVIEW bindings. Appendix A lists Pan's default set of key bindings.

### 2.3.2 Menu Bindings

A menu binding consists of a menu, a menu item, and a command name. Like key bindings, menus and menu bindings may be either local or global. The menus associated with a buffer appear when right button of the mouse is pressed while the mouse is positioned over the edit window. Figure 3 shows a menu selection being made.

![Figure 3: Pan with a buffer menu visible](image)

A menu binding is designated by a menu title and a selection name, denoted typographically by `MenuTitle: SelectionName`. It is not necessary for the selection name to be identical to the name of the command bound to the selection.
Default menu bindings are best discovered by mousing around. Appendices B–D list the default menu bindings for a text buffer, for the base buffer, and for the help buffer.

2.3.3 Operand Level Bindings

A keystroke sequence or menu item can be bound to a command that implements a generic operation. The generic operation, in turn, consults the operand-level bindings and the current operand level of the active viewer to determine the actual editor command to execute.

Each viewer has a current operand level which can be used to control the actions of operand-generic operations. The generic operations are: `:next`, `:previous`, `:select`, `:mouse-select`, `:mouse-extend`, `:cut`, `:copy`, `:paste`, and `:delete`.

For example, the key "-N" is bound to the command `Next-@Level`. If the current operand level is "Character", and the binding of the generic command `:next` at the level "Character" is the command `Next-Character`, then `Next-@Level` will execute `Next-Character`.

Setting the operand level is a bit like changing modes in a moded editor. The current operand level persists across operations. In pure text editing, the operand level is of limited usefulness. When editing objects that have a richer operand domain, such as computer programs, the ability to select and navigate using the operand level bindings is a bonus. For instance, a programming language might define operand levels such as "Expression", "Statement", and "Declaration". Figure 5 on page 25 shows the operand-level choices for a language-based viewer.

The current operand level affects only those commands that consult it. These commands are syntactically distinguished by containing the phrase "@Level" in their name.

2.4 The Edit Cursor

Each viewer has a single edit cursor; buffers have as many cursors as they have viewers. The edit cursors are independent from each other. An edit cursor appears on the screen as an inverse-video or outlined box highlighting the character selected by that cursor.

All insertions and deletions occur at the position to the left of the character selected by the cursor. If a command alters text not located at the edit cursor (for instance, by deleting the current selection), the cursor is moved to the point of change. Operations that modify the object being edited move the edit window so that the change (and therefore the cursor) is visible.

A viewer's edit cursor may not be visible on the screen due to scrolling or other motion. The command `Frame-Edit-Cursor` in the Window menu moves the edit window so that the cursor is visible.

2.5 Regions and The Selection

Many of the text-oriented commands in Pan operate on a contiguous sequence of characters called a region of text.

Every buffer can have a specially designated region of text called the selected region or simply the selection. When the selection is set, it is highlighted in all of the viewers in which it is visible. Figure 1 on page 3 shows a selection shared by two viewers. The value of the option `:region-highlight-op` determines whether the current selection is highlighted using underlining or inverse-video. Commands that alter the contents of the buffer deselect the current selection.
Pan also has an implicit selection, namely the region between the edit cursor and the top mark of the mark stack. This selection is different for each viewer, since each viewer has a different edit cursor. In Emacs, the implicit selection is the only region available, while in Pan, it is secondary to the visible selection. Commands that operate on the implicit selection are provided mostly for Emacs compatibility.

Pan's style of selection is similar to, but not the same as, SUNVIEW's. Clicking the middle button of the mouse selects and highlights the region between the cursor and the mouse, but placing or dragging the cursor does not affect the selection. The selection, the position of the mouse, and the position of the edit cursor are all independent.

2.6 Rings

A ring is a circular bounded stack. Adding an item to a ring pushes the other items just like a bounded stack. The oldest value in the stack may be discarded to preserve the boundedness.

Rings can also be “cycled”, where the top value is moved to the position of the oldest value, and all of the other values move up—the second youngest becoming the top. Cycling a ring by $n$ values moves the $n$th element (modulo the size of the ring) to the top. The top value in a ring is called the “contents” of the ring.

Both the clipboard and the kill rings are implemented as rings.

2.6.1 The Clipboard

The clipboard—a holder for regions of text—is shared among buffers. A selection can be copied (or cut) to the clipboard and then pasted into another buffer. These operations are modeled on the Macintosh user interface. Unlike the Macintosh clipboard, Pan's clipboard is a ring that contains several items. The size of the clipboard is determined by the value of the option :clipboard-max-size.

2.6.2 Kill Rings

The kill ring is a repository for deleted text. Commands that “kill” text place the killed text into the kill ring. This text can be retrieved at a later time.

The kill ring facility is less powerful than Emacs. In Pan, the kill ring is local to a buffer rather than global to all buffers. We are still experimenting with this aspect of the user interface. The size of the kill ring is determined by the value of the option :killring-max-size.

Cutting text to the clipboard does not affect the entry in the kill ring; killing text does not affect the clipboard.

2.7 Marks and the Mark Stack

A mark is a character position in a textual object. Marks associate with the character to the left of the position; when text is deleted, affected marks migrate to the beginning of the deletion.

Each buffer has a stack of marks called the mark stack. Marks in Pan are used for two purposes: to remember a cursor position, and to construct regions. The top mark on the stack is usually referred to as “the mark”.

---

4Macintosh is a registered trademark of Apple Computer, Inc.
2.8 Syntax Classes

Every buffer has a set of syntax class definitions. Within a buffer, each ASCII character is a member of exactly one syntax class. The possible classes are: :space-char, :word-char, :punct-char, :lbracket, :rbracket, or :other. Every character in the classes :lbracket or :rbracket must have a matching bracket specified.

2.9 Options, Flags, and Variables

Options, flags, and variables are provided for controlling and extending the system. The following paragraphs provide a brief summary of their properties and use.

Options are used to control user-configurable settings. They are strongly-typed, scoped variables. The value of an option local to a buffer generally hides the value of that option in the global space. Appendix E lists the basic options, their types, and their default values. The value of an option is obtained using the command Option-Value (see section 4.3).

A flag is strongly-typed, scoped variable capable of holding only a single bit of information. Like options, flags are scoped relative to buffers. Unlike options, flag values can be displayed on the information panel. For instance, if the :text-modified flag is set (meaning that the text of an object has been modified since its last save), a "*" appears on the information panel. Appendix F lists the default set of flags, together with their display properties.

A Pan variable is a Lisp global variable that is known to the help system.

2.10 Communication with Other SUNVIEW Clients

Communication with other workstation client programs using the SUNVIEW selection mechanism is not yet implemented.

2.11 What's Where?

Just as a buffer ties several viewers together, there is some global state shared by all buffers. This state includes default bindings, default option values, the list of buffers appearing in the base buffer, and the clipboard.

The following table provides a synopsis of which objects have custody of various other objects. In general, local values override more global values when there is a choice.
3 Editing with Pan

This section is an introduction simple editing using Pan. At the beginning of each subsection is a list of the commands discussed, together with their default bindings. Control keys are represented by prepending the character "-" in front of the key, e.g., Control-X is shown as -X, and the keystroke sequence Control-X Control-F is shown as -X-F where the hyphen separates the keys in the sequence. The prefix Escape is denoted Esc-.

Menu bindings are denoted by MenuTitle: SelectionName where MenuTitle is the title of the menu and SelectionName is the name appearing in the menu. Operand-level bindings are denoted by (Command, "level").

3.1 Getting Started

Pan for Sun Model 3 workstations resides in "piper/bin/pan3. Pan can be run from a shell tool, a command tool, or a menu under SUNVIEW.

When Pan is started, the files named on the command line are read into edit buffers and prepared for editing. Once initialization is complete, the base frame will appear. Section 4.1 provides more information about Pan’s start-up processing.

3.2 Quitting Pan

Write-Files-Exit
Write-Files-Exit
Exit!

Write-Files-Exit is the normal method for terminating an editing session. When modified buffers exist you will be asked whether they should be saved. The command Exit! terminates an editing session without saving any modified buffers.
3.3 Suppling Arguments to Commands

There are three ways to provide arguments to Pan commands: by a numeric prefix argument, by setting the current selection, or by responding to a prompt. The actual method used depends on the particular command. Commands that can use a numeric argument normally check for the presence of a numeric prefix, while commands that require textual arguments must use prompting. A few, like Visit-Selection, check for a selection before prompting.

3.3.1 Numeric Prefix Arguments

`Read-Prefix-Arguments`

The behavior of many commands can be altered by suppling numeric prefix arguments. In most cases, the argument is interpreted as a repetition factor, and the effect of the command is simply repeated. `Read-Prefix-Arguments` reads the prefix arguments from the keyboard. There are two ways to type such arguments: by typing zero or more minus signs (hyphens) followed by a sequence of digits, or by repeating a keystroke sequence bound to `Read-Prefix-Arguments`. In the latter case, each repetition corresponds to addition by 4.

3.3.2 Prompts and Pop-Ups

Pan prompts for input by making a small pop-up window appear on the screen. The pop-up remains on the screen until you complete the input. When a pop-up is present it seize all window input and output on your workstation.

Pop-ups have one or more "buttons" on their lower edge; clicking the left button of the mouse over one of those areas completes and confirms the prompt. When typing a textual argument into a pop-up, your standard UNIX editing characters serve to edit the input. Pop-ups for textual arguments are confirmed by selecting a button or by hitting the Return key.

3.4 Aborting Commands

`Abort-Command`  \(^G\)

`Abort-Command`  \(^C^-G\)

`Abort-Command`  `Esc^-G`

`Abort-Command`  `^X^-G`

`Abort-Command`  `^Z^-G`

`Cancel-Command`  "Cancel" on Popups

To abort a keystroke sequence, type \(^G\) with any prefix. To abort a menu selection, move the mouse cursor outside (away from) the menu, and release the mouse button. When responding to a prompt, selecting the Cancel button aborts the command.

Once a command is initiated, there is little you can do to stop it. If the command takes more than an instant to complete, the mouse cursor image, normally an arrow, may be replaced by a light bulb.
3.5 Undoing Actions

```
Undo
Undo
```

The undo facilities of Pan allow you to undo the most recent action or series of similar actions. For instance, typing a series of characters and then invoking Undo will remove the entire series of characters just typed.Undoing a second time restores the text removed by the first Undo.

3.6 Getting Help

```
Reset-Help-Buffer
```

Pan is largely self-documenting. Each viewer has an associated Help menu that provides access to the help information supplied by the system. This information will be displayed in the help buffer—a special buffer known to the system. Figure 1 on page 3 shows the help buffer in the upper right-hand corner of the screen. The help buffer acts like a normal text viewer, except that it is altered only by the help commands.

Most help commands delete the text in the help buffer before adding their contribution. Others, such as Describe-Selection and Apropos-Selection add additional information to the buffer. The information in the help buffer can be saved to a text file at any time. The command Reset-Help-Buffer empties the help buffer.

```
Apropos-Selection
Apropos-All-Symbols
Apropos-Symbol
```

The "Apropos" family of commands associate a keyword with a list of command names. For instance, invoking the command Apropos-Symbol and supplying the argument "clipboard" lists, in the help buffer, all of the commands that use the clipboard. In another example, if the text "command" is selected, then invoking Apropos-Selection will list all of the editor commands defined in Pan. Both Apropos-Symbol and Apropos-Selection operate identically after their argument has been specified.

```
Describe-Selection
Describe-Symbol
List-All-Commands
List-All-Flags
List-All-Options
List-All-Variables
Show-Buffers
Show-Flag-Values
```
Commands beginning with “Describe-” provide detailed information about a Pan object. Commands beginning with “List-” simply list the names of all objects of a given type (e.g., all commands or all options). The “Show-” commands provide the detailed information of a “Describe-” command for all objects of a given type.

3.7 Buffers and Viewers

The commands presented in this section describe basic buffer and viewer handling.

3.7.1 Visiting Files and Buffers

Visit-Selection  
Visit-Selection

Visit-File  
Visit-File

Visit-Buffer  
Remove-Buffer

Remove-Selected-Buffer

List-Files  
List-Files

Show-Status-Line

To edit an object, one must first have opened a viewer onto it. The Visit-Selection command is one way to open a viewer. This command normally appears as Visit in either the top-level or the File menus. It is used by first selecting a buffer name in the base buffer.

Visit-File prompts for a file name. If that file is not already in a buffer, a new buffer is allocated and the file is read and prepared for editing.

The “Remove-” commands remove a given buffer from the set of buffers. If the object has been modified since it was last saved, you will be offered the chance to save the buffer. Once Remove-Buffer or Remove-Selected-Buffer has been executed, all of the state associated with the removed buffer is lost.

List-Files prompts for a directory or file name expression, and lists those files, using the ls command, in the help buffer. Flags to the ls command are determined by the value of the option :Is-flags.

The command Show-Status-Line can be used to display status information about the active buffer in the information panel. The option :mode-line-fmt specifies which status information is displayed.
3.7.2 Saving and Writing Files

Save-Buffer-File  \(^{-X-}{-S}\)  
Save-Buffer-File  
Save-All-Buffers  \(^{-X-}{-R}\)  
Write-Files-Exit  
Write-To-File  \(^{-X-}{-W}\)  
Write-To-File  
Overwrite-File!  
Write-Selection-To-File  
Append-Selection-To-File  
Toggle-Read-Only  \(^{-X-}{-Q}\)  

Save-Buffer-File saves the contents of the current buffer in its associated file; Save-All-Buffers saves all of the modified buffers being edited. Write-Files-Exit performs the standard termination sequence of saving modified buffers and exiting. Both Save-All-Buffers and Write-Files-Exit prompt for whether a particular buffer is to be written to file.

All of the commands Write-To-File, Write-Selection-To-File, Overwrite-File!, and Append-Selection-To-File prompt for the name of a file to write.

Toggle-Read-Only toggles the :read-only flag on the buffer. When this flag is set, a "$" appears on the information panel and you will be prevented from modifying or writing that file. This flag is set by default when you commence editing a file for which you do not have permission to write. Toggling the :read-only flag does not affect the permissions on the file.

3.7.3 Manipulating Viewers

Open-Another-Viewer  \(^{-X-}{-2}\)  
Open-Another-Viewer  
Close-Active-Viewer  \(^{-X-}{-0}\)  
Close-Active-Viewer  
Close-Active-Viewer  
Close-Active-Viewer  
Redraw  \(^{-L}\)  

When you visit a buffer that has no visible viewers, Pan will reopen the buffer's most recently closed viewer. If the buffer has no viewers, a new viewer is created. To open a second (or third, or fourth, ...) viewer onto a buffer, execute the command Open-Another-Viewer.

Closing a viewer causes it to disappear from the screen. Its internal state is retained so that the viewer can be reopened later. To close a viewer, execute Close-Active-Viewer in the viewer that you wish to close. Alternatively, press the key L7 (Open) when the viewer that you wish to close has the input focus. Closing the viewer associated with the base buffer causes the base buffer to become iconic, and closes all other visible viewers.
Viewers are reopened using the same commands as are used for opening new viewers. They are reopened in last-in, first-out order relative to the order in which they were closed. Closing a viewer frees its associated viewport for use by other viewers.

Redraw redraws the active edit window.

3.8 Scrolling

Pan’s scrolling behavior is a simplified version of the SunView scrolling protocols. The scroll bars in a viewer respond to simple scrolling commands. For vertical scrolling, pressing the left button in the vertical scroll bar moves the edit window toward the end of the file, the right button moves the edit window towards the beginning of the file, and the middle button thumbs the edit window to the point in the object indicated by the scroll cursor.

When scrolling horizontally, pressing the left button in the horizontal scroll bar moves the edit window toward the end of the line, the right button moves the edit window towards the beginning of the line, and the middle button thumbs the edit window to the point in the line indicated by the scroll cursor.

Mouse-Forward-Vscroll (Scrollbar) Mouse_Left
Mouse-Backward-Vscroll (Scrollbar) Mouse_Right
Mouse-Abs-Vscroll (Scrollbar) Mouse_Middle
Forward-Vscroll \( ^{-}V \)
Backward-Vscroll \( ^{-}V \)
Left-Hscroll \( ^{-}X-< \)
Right-Hscroll \( ^{-}X-> \)
Frame-Edit-Cursor Window: Frame Edit Cursor

The commands Mouse-Forward-Vscroll, Mouse-Backward-Vscroll, and Mouse-Abs-Vscroll are bound to the left-, right-, and middle mouse buttons when the mouse is over the vertical scroll bar.

During vertical scrolling, when the option :proportional-scroll is set to be true (in Lisp, 't), the amount that the edit window is scrolled depends upon the distance between the mouse cursor and the top of the scroll bar. For small movements, place the cursor near the top of the scroll bar. For larger movements, place the mouse cursor near the bottom of the scroll bar. To scroll an entire screen, place the mouse cursor opposite to the last line of text visible in the edit window.

When the option :proportional-scroll is set to be false (in Lisp, 'nil), the vertical scrolling commands move the edit window by a full screen at a time.

To “thumb” the viewer to an absolute position in the file, place the mouse cursor into the scroll bar and press Mouse_Middle. The viewer will be moved to the line in the file corresponding relative distance between the top of the scroll bar and the position of the mouse cursor.

Scrolling the screen by a full screen at a time can also be achieved by using Forward-Vscroll and Backward-Vscroll. Both of these commands consult the numeric prefix argument to determine the number of screens to move.

Left-Hscroll and Right-Hscroll are keyboard variants of the horizontal scrolling commands.

The command Frame-Edit-Cursor shifts the edit window so that the edit cursor will be visible.
3.9 Cursor Motion

Mouse-Select-@Level
Cursor-To-Mouse
Next-@Level
Next-Character
Next-Word
Next-Line
Previous-@Level
Previous-Character
Previous-Character
Previous-Word
Previous-Word
Previous-Line
Previous-Line
Move-To-BOL
Move-To-EOL
Move-To-BOB
Move-To-EOB
End-Of-Word
First-Non-Blank
Goto-Line

The left button of the mouse is bound to the command Mouse-Select-@Level. When the operand level is set to "Character", it sets the edit cursor to the character selected by the mouse icon. The cursor can also be moved using cursor motion commands shown above. The "Next-" and "Previous-" commands use the numeric prefix argument to determine the number of units to move.

Move-To-BOL moves the edit cursor to the first position on the current line; Move-To-EOL moves the edit cursor last position on the current line. To move to the beginning or end of the object being edited, use Move-To-BOB or Move-To-EOB, respectively.

End-Of-Word moves the cursor to the end of the word that encloses the cursor. Finally, First-Non-Blank moves the cursor to the first character (that is not white space) on the current line.

The command Goto-Line moves the cursor to the line specified by the numeric prefix argument. If there is no prefix argument, Goto-Line prompts for a line number. Internally, Pan treats the first line of a file as line number 0. If the option :zero-index-lines is false, the argument to Goto-Line is treated as a 1-indexed line number and is converted appropriately.

3.9.1 Mark Commands

Set-Mark
Pop-Mark

The left button of the mouse is bound to the command Mouse-Select-@Level. When the operand level is set to "Character", it sets the edit cursor to the character selected by the mouse icon. The cursor can also be moved using cursor motion commands shown above. The "Next-" and "Previous-" commands use the numeric prefix argument to determine the number of units to move.

Move-To-BOL moves the edit cursor to the first position on the current line; Move-To-EOL moves the edit cursor last position on the current line. To move to the beginning or end of the object being edited, use Move-To-BOB or Move-To-EOB, respectively.

End-Of-Word moves the cursor to the end of the word that encloses the cursor. Finally, First-Non-Blank moves the cursor to the first character (that is not white space) on the current line.

The command Goto-Line moves the cursor to the line specified by the numeric prefix argument. If there is no prefix argument, Goto-Line prompts for a line number. Internally, Pan treats the first line of a file as line number 0. If the option :zero-index-lines is false, the argument to Goto-Line is treated as a 1-indexed line number and is converted appropriately.
Set-Mark and Push-Mark push and pop the mark stack of the active buffer. The commands Swap-Dot-And-Mark and Dot-To-Mark move the edit cursor to the position indicated by the top of the mark stack. They differ in that the first exchanges the cursor’s position with the top mark, while the second pops stack.

3.10 Editing Text

This section describes the commands for text manipulation.

3.10.1 Setting the Operand Level

You can ignore the operand level settings, and Pan will operate pretty much like Emacs. If you want to experiment, however, you will notice that each viewer panel contains a sequence of items labeled “Level:”. The small button containing an upward-pointing arrow is bound to Up-@Level. Clicking on this button raises the operand level. Similarly, clicking the button labeled with a downward-pointing arrow lowers the operand level by calling Down-@Level. Both Up-@Level and Down-@Level consult the numeric prefix argument to determine the number of levels to move (the default is 1). If the option :wrap-@level is true, then the level will wrap around from top to bottom or bottom to top as necessary.

The current level is displayed on the panel. Depressing Mouse_Right over the displayed level causes a menu containing all of the levels to appear. The levels in the menu appear in order, with the highest level at the top of the menu.

To select a new level, continue to hold down Mouse_Right, move the mouse until the level that you desire is highlighted, and then release Mouse_Right. If no change is desired, move the mouse cursor away from the menu and release Mouse_Right.

The command Set-@Level-To-Character can be used to set the operand level to “Character”.

3.10.2 Setting the Selection

You can ignore the operand level settings, and Pan will operate pretty much like Emacs. If you want to experiment, however, you will notice that each viewer panel contains a sequence of items labeled “Level:”. The small button containing an upward-pointing arrow is bound to Up-@Level. Clicking on this button raises the operand level. Similarly, clicking the button labeled with a downward-pointing arrow lowers the operand level by calling Down-@Level. Both Up-@Level and Down-@Level consult the numeric prefix argument to determine the number of levels to move (the default is 1). If the option :wrap-@level is true, then the level will wrap around from top to bottom or bottom to top as necessary.

The current level is displayed on the panel. Depressing Mouse_Right over the displayed level causes a menu containing all of the levels to appear. The levels in the menu appear in order, with the highest level at the top of the menu.

To select a new level, continue to hold down Mouse_Right, move the mouse until the level that you desire is highlighted, and then release Mouse_Right. If no change is desired, move the mouse cursor away from the menu and release Mouse_Right.

The command Set-@Level-To-Character can be used to set the operand level to “Character”.

Mouse-Select-@Level
Mouse-Extend-@Level
Select-Region-Dot-To-Mark
Select-Word
Select-Buffer

Mouse_Left
Mouse_Middle
Esc-~W
Esc-@Q
~X-h
Mouse-Select-Line
Mouse-Select-Word
Mouse-Select-Fullword
Mouse-Extend-Selection-Fullword
Deselect-Region

The current selection is set by the commands Mouse-Select-@Level, Mouse-Extend-@Level, and Select-Region-Dot-To-Mark. Mouse-Extend-@Level is normally bound to the middle button of the mouse. When the level is "Character", this command selects the region between the edit cursor and the mouse. At other levels, it selects the region that includes both the current selection and the operand (relative to the current operand level) beneath the mouse cursor. Select-Region-Dot-To-Mark selects the implicit region between the top mark on the mark stack and the edit cursor.

The command Select-Word sets the current selection to be the region from the edit cursor to the end of the word surrounding the edit cursor, while Mouse-Select-Fullword sets the current selection to be the full word beneath the mouse cursor. Mouse-Extend-Selection-Fullword extends the current selection to include the full word beneath the mouse cursor. Select-Buffer selects the entire buffer.

The current selection can be cleared using Deselect-Region.

3.10.3 Inserting Text

Self-Insert
Quote-Insert
Newline-And-Indent
Insert-Newline
Indent-Like-Previous-Line
Open-Line
Split-Line
Insert-Parentheses
Insert-Rparen-And-Match
Insert-Rbrace-And-Match
Insert-Rbracket-And-Match
Insert-File
Insert-File

Typing a printable character generally causes that character to be inserted into the text at the position of the edit cursor. Quote-Insert inserts the next ASCII character typed.

Newline-And-Indent is bound to the newline character ‘\n’; if the option :autoindent is true, the next line will be indented to the level of the previous line by inserting tabs and blanks. Indent-Like-Previous-Line simply reindents the current line to the level of the previous line. Open-Line inserts newlines after the cursor; the number of newlines inserted is determined by the value of the numeric prefix argument (default 1). Split-Line does the same thing, but also indents any text following the cursor to its original horizontal position.
The command **Insert-Parentheses** inserts a pair of matching parentheses at the cursor, and positions the cursor between them. The command **Insert-Rparen-And-Match** inserts a right parenthesis and briefly shows the matching left parenthesis by moving the cursor if the match is visible in the viewer or by displaying the line containing the match as a message on the panel otherwise. **Insert-Rbrace-And-Match** and **Insert-Rbracket-And-Match** perform the same action for braces and brackets.

**Insert-File** prompts for a file name and copies the contents that file into the active buffer at the position of the edit cursor.

### 3.10.4 Filling Text

- **Set-Auto-Fill-Column**
- **Toggle-Auto-Fill**

**Pan** has a rudimentary mechanism for filling text lines as you type them. When auto-filling is on, the keys **Space** and **Return** are bound to special procedures. These procedures compare the current horizontal position of the cursor with the value of :auto-fill-column. If the line is too long, it will be broken where appropriate; if not, the procedures act like **Self-Insert**.

Use **Toggle-Auto-Fill** to turn on autofilling and again to turn it back off. When filling is on, you can type text continuously without worrying about line length. Use **Set-Auto-Fill-Column** (with a numeric prefix argument) to set the maximum line length used by auto-filling.

### 3.10.5 Deleting Text

- **Delete-@ Level**
- **Delete-Character**
- **Delete-Previous-Character**
- **Delete-Word**
- **Delete-Fullword**
- **Delete-Previous-Word**
- **Delete-Line**
- **Delete-Selected-Region**
- **Delete-Region-Dot-To-Mark**
- **Delete-Blank-Lines**
- **Delete-Indentation**

Deleted text can be recovered by issuing the **Undo** command immediately after the deletions, but in no other way. Killed text is retained in the kill ring. The standard bindings reflect this limitation by using kill commands when removing large regions. **Pan** merges deletions that are contiguous in space and time into a single undoable action.

**Delete-Character** deletes the character selected by the active cursor; **Delete-Previous-Character** deletes the character before the cursor. The pair of commands **Delete-Word** and **Delete-Previous-Word**...
Pan I: An Introduction For Users

Word delete from the cursor to the end (beginning) of the word enclosing the cursor, while Delete-Fullword deletes the entire word enclosing the cursor. The commands Delete-Selected-Region and Delete-Region-Dot-To-Mark operate on the selection and the implicit selection, respectively.

There are several ways to delete white space around the cursor. Delete-Blank-Lines deletes vertical and horizontal white space, leaving exactly one blank line at the cursor. Delete-Horizontal-Space deletes white space surrounding the cursor on the same line as the cursor; Just-One-Space does the same thing, but leaves exactly one space at the cursor. Delete-Indentation removes any leading white space on the line containing the cursor.

3.10.6 Killing Text

<table>
<thead>
<tr>
<th>Command</th>
<th>Command</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kill-Word</td>
<td>Esc-d</td>
</tr>
<tr>
<td>Kill-Previous-Word</td>
<td>Esc-Del</td>
</tr>
<tr>
<td>Kill-To-EOL</td>
<td></td>
</tr>
<tr>
<td>Kill-Selected-Region</td>
<td>W</td>
</tr>
<tr>
<td>Kill-Selected-Region</td>
<td></td>
</tr>
<tr>
<td>Kill-Region-Dot-To-Mark</td>
<td></td>
</tr>
<tr>
<td>Copy-Selection-As-Kill</td>
<td>Esc-w</td>
</tr>
<tr>
<td>Yank-From-Kill-Ring</td>
<td>Y</td>
</tr>
<tr>
<td>Cycle-Yank</td>
<td>Esc-Y</td>
</tr>
<tr>
<td>Cycle-Kill</td>
<td></td>
</tr>
<tr>
<td>Show-Kill</td>
<td>X-?</td>
</tr>
<tr>
<td>Cycle-Show-Kill</td>
<td>X-!</td>
</tr>
</tbody>
</table>

Like the deletion commands of the previous section, commands that kill text remove the text from an object. Unlike deletion commands, however, commands that kill text also place the removed text into the kill-ring.

The pair of commands Kill-Word and Kill-Previous-Word kill from the cursor to the end (beginning) of the word enclosing the cursor. The commands Kill-Selected-Region and Kill-Region-Dot-To-Mark kill the selection and the implicit selection, respectively. Kill-To-EOL kills all characters up to the end of the line.

Text in the kill ring can be recovered using the command Yank-From-Kill-Ring which inserts the contents of the top of the kill-ring into the buffer at the current cursor position. Cycle-Yank cycles the kill ring before yanking the top of the kill-ring. Copy-Selection-As-Kill copies the current selection to the kill-ring.

The commands Show-Kill, Cycle-Kill, and Cycle-Show-Kill can be used to view the contents of the kill ring, and to cycle the ring.

3.10.7 The Clipboard

<table>
<thead>
<tr>
<th>Command</th>
<th>Command</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cut-To-Clipboard</td>
<td>L10 (Delete^n)</td>
</tr>
<tr>
<td>Cut-To-Clipboard</td>
<td></td>
</tr>
<tr>
<td>Copy-To-Clipboard</td>
<td>L6 (Put)</td>
</tr>
</tbody>
</table>
These commands manipulate the contents of the clipboard. **Cut-To-Clipboard** deletes the current selection and places it onto the clipboard. **Copy-To-Clipboard** places a copy of the current selection onto the clipboard. **Paste-From-Clipboard** inserts a copy of the clipboard's contents at the edit cursor. Finally, **Replace-From-Clipboard** replaces the selected region with the contents of the clipboard.

The command **Show-Clipboard** displays the contents of the clipboard in the help buffer. **Cycle-Clipboard** cycles the clipboard using the prefix argument to determine the number of entries to move, while **Cycle-Show-Clipboard** combines the two actions.

### 3.10.8 Copying and Moving Text

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Copy-Selection-To-Cursor</td>
<td>Copy to Cursor</td>
</tr>
<tr>
<td>Move-Selection-To-Cursor</td>
<td>Move to Cursor</td>
</tr>
</tbody>
</table>

Text can copied or moved within a buffer by using either the clipboard or the local kill ring. However, the above commands are useful short cuts. Both use the current selection as the source to copy or move, and the edit cursor to mark the destination. These commands operate only within a buffer; copying and moving between buffers requires the clipboard.

### 3.10.9 Commands for Changing Case

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Capitalize-Word</td>
<td>Esc-c</td>
</tr>
<tr>
<td>Lowercase-Word</td>
<td>Esc-l</td>
</tr>
<tr>
<td>Uppercase-Word</td>
<td>Esc-u</td>
</tr>
<tr>
<td>Capitalize-Selection</td>
<td>Esc-&quot;c</td>
</tr>
<tr>
<td>Lowercase-Selection</td>
<td>Esc-&quot;l</td>
</tr>
<tr>
<td>Uppercase-Selection</td>
<td>Esc-&quot;u</td>
</tr>
</tbody>
</table>

These commands are use to change cases within a word or region. The word-oriented commands operate on the region from the cursor to the end of the word. They leave the cursor at the end of the region when done.
3.10.10 Transposing

- **Transpose-Characters** exchanges the character at the cursor and the character before the cursor.
- **Transpose-Previous-Characters** exchanges the two characters to the left of the cursor.
- **Transpose-Lines** exchanges the line containing the cursor with line before it.

3.11 Searching Text

*Pan* provides commands for searching for regular expressions and for matching balanced brackets.

### 3.11.1 Regular Expressions

- **Re-Search-Backward** \(^-\text{T}\)
- **Re-Search-Forward** \(^-\text{T}\)

These commands search for text matching the standard UNIX regular expressions. For a description of those expressions, see the ED(1) manual page of the UNIX Programmer's Manuals. *Pan* is unable to search for patterns which contain embedded newline characters. The most recently specified regular expression is shared by all buffers.

When text matching a pattern is found, the active cursor is moved to the first character in the match, and the matched text is selected.

Both **Re-Search-Forward** and **Re-Search-Backward** process the numeric prefix argument idiosyncratically: the presence of a prefix argument causes the command to search using the last regular expression specified. For instance, \(^-\text{U}^-\text{S}\) invokes **Re-Search-Forward** using the most recent search pattern—and the command will match the next occurrence of the pattern. Alternatively, supplying an empty string as the regular expression causes the previously specified expression to be used.

When the option *:autowrap-search* is true, searches wrap from one end of the buffer to the other; if that option is false, searches terminate when finding the beginning or end of the buffer.

While a Query-Replace command has not yet been implemented, one can rapidly perform that action by putting the replacement text into the clipboard, and then alternating searches with **Replace-From-Clipboard**.

### 3.11.2 Balanced Bracket Commands

- **Backward-Expr** \(\text{Esc}-\text{B}\)
- **Forward-Expr** \(\text{Esc}-\text{F}\)
- **Select-Expr** \(\text{Esc}-\text{G}\)
- **Kill-Expr** \(\text{Esc}-\text{K}\)
- **Show-Match** \(\text{Esc}-\%\)
These commands use syntax class definitions to operate on balanced brackets. Both `Forward-Expr` and `Backward-Expr` move the cursor. The `Select-Expr` command selects the balanced bracket expression surrounding the cursor, and the command `Kill-Expr` kills it.

If the cursor character is a bracket, `Show-Match` moves the cursor to the matching bracket, pauses for the value of the option `:pause-ticks` internal ticks, and returns the cursor to its original position.

### 3.12 Editing Programs

`Pan` provides facilities for editing tree-structured objects described by a formal language. These objects include programs, which are described by a programming language. To take advantage of those facilities, the language being edited must be described to `Pan` using the language-definition language `Ladle[4]`. Language descriptions are beyond the scope of this manual. In this section, the basic language-oriented editing features of `Pan` are described.

`Pan` currently supports two languages: Modula-2[8] and `ASPLE[5]`. `ASPLE` is a simple example language used for demonstrations and for learning `Pan`. Other language descriptions, including `C`, are under development.

Full text editing is always available when editing structures. In fact, the actual operations that alter a structure are reduced to textual operations. Incremental syntactic analysis (parsing) then updates the internal tree structure. Most of the time, this transformation is hidden from the user—it occurs automatically as operations like "delete the selected subtree" are invoked. The next sections describe language-oriented editing with `Pan` in more detail.

#### 3.12.1 Language-Oriented Viewers

Structures are displayed using the same kind of viewer used to display text. A future version of `Pan` will provide a pretty-printing viewer that keeps the displayed structure consistent with its internal form. Right now, the re-indentation must be performed manually. Language-oriented viewers support the same scrolling operations as viewers onto textual objects.

Figure 4 shows a language-oriented viewer for a buffer containing an `ASPLE` program. Note that the "Language:" field in the information panel now reads "ASPLE", and that the root menu has been customized to contain the submenus `Tree` and `Syntax`. The `Tree` menu provides access to the basic tree navigation commands, while the `Syntax` menu provides language-oriented editing commands. As with text viewers, however, those commands are also available from the keyboard.

A language-oriented viewer has three new flags in the information panel. The "L" (`:lex-ok`) flag will be grey when there are changes in the buffer that have not yet been lexically analyzed (the first phase of parsing). The "T" (`:tree-ok`) flag will be visible, but grey, when there are changes that have not yet been incorporated into the tree by the parser. Both "L" and "T" flags are displayed in solid tones when the buffer's internal structures are consistent. The "!" (`:parse-errors`) flag appears when there are lexical or syntactic errors in the program being edited. Section 3.12.4 discusses this topic in more detail.

#### 3.12.2 Selection

Selection during language-oriented editing relies on the notion of operand levels (section 2.3.3). In a language-oriented viewer, the set of operand levels is much richer than in the text world. The
Figure 4: A viewer with a language other than "Text"

operand levels for text editing are included, as well as levels corresponding to basic abstractions in the language being edited. Figure 5 shows the set of operand levels defined for ASPLE. The "Error" level is of special significance: it is used to locate errors in a syntactic structure and to move the selection from one error to another.

As with text, depressing the left button of the mouse selects the object beneath the mouse cursor. The object actually chosen is determined by the setting of the operand level. Thus if the mouse cursor is over the character "*" in Figure 5 when the left button is clicked, the selected object might be the underlying lexeme, expression, or statement depending on whether the operand level is "Lexeme", "Expression", or "Statement" respectively.

What happens if the object beneath the mouse cursor is not an element of the class of objects specified by the current operand level? Then Pan uses a heuristic to find an object close to the mouse cursor that is an element of that class. This behavior is fairly predictable, although in some cases it leads to unforeseen selections.
Figure 5: The operand levels for ASPL

To select arbitrary portions of a structure, set the operand level to "Character" and operate on the textual representation.

3.12.3 Navigation

- Next-@Level
- Previous-@Level
- Tree-Up
- Tree-Up
- Tree-Down
- Tree-Down
- Tree-Left
- Tree-Left

-F
-B
-R6
-Tree: Up
-R14
-Tree: Down
-R9
-Tree: Left
Pan I: An Introduction For Users

Pan provides two sets of language-oriented navigation operations. The most useful set involves the operations Next-@Level and Previous-@Level. They perform preorder and inverse-preorder tree walking operations relative to the current operand level. For example, if the operand level setting is "Error", the command Next-@Level moves the cursor to the "next" error node in the tree.

A second set of navigation commands includes Tree-Up, Tree-Down, Tree-Left, and Tree-Right. These are the usual tree-oriented commands. Tree-Up moves the cursor to the parent, Tree-Down moves it to the leftmost child of the current node, Tree-Left and Tree-Right move the cursor to the appropriate sibling in the tree. These commands can be used to explore the actual tree structure, as opposed to the tree structure imposed by the operand hierarchy mechanism. Use them carefully!

3.12.4 Parsing and Syntactic Errors

Parsing incorporates changes in a buffer's contents into the tree that represents a program. The parsing method used is incremental—only the areas affected by the changes are reparsed. Parsing occurs whenever a language-oriented operation takes place, such as when the operand level is changed to be a non-textual level or when a tree-oriented navigation command is invoked. Parsing can also be invoked manually, using the command Parse-Buffer. The command Rectify-Tree is used internally to invoke parsing if there have been any changes in the contents of the buffer.

Parsing is a two-stage process. In the first stage, the text stream is broken into larger fragments called lexemes. Lexemes are the basic symbols in the language being edited, e.g., keywords, identifiers, constants, and comments. When a buffer is lexically analyzed, various classes of lexemes are given different visual images using fonts. The following table defines the relationship between font codes and characters in lexemes:

<table>
<thead>
<tr>
<th>Class</th>
<th>Example</th>
<th>Font</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unanalyzed characters</td>
<td>inserted text</td>
<td>0</td>
</tr>
<tr>
<td>Ignored by the lexical analyzer</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>Fixed-length lexemes</td>
<td>keywords</td>
<td>2</td>
</tr>
<tr>
<td>Recognized but not parsed</td>
<td>comments</td>
<td>3</td>
</tr>
<tr>
<td>Variable-length lexemes</td>
<td>identifiers</td>
<td>4</td>
</tr>
</tbody>
</table>

Unanalyzed characters are always displayed using the default font 0.

After lexing, provided that there are no lexical errors, the parser updates the internal structured representation. If the parser encounters syntactic errors, the number of errors discovered during the parse is displayed on the annunciator line, and the :parse-errors flag is set. This flag appears as a "!" on the information panel.

When the parser detects an error, the subtrees involved in the error are gathered into an "error subtree". Selecting the subtree rooted at an error node causes the error message from the parser to be displayed on the annunciator line. Figure 6 shows the display of an error.
As noted previously, the operand level can be set to “Error” in order to find and select errors in the tree. Figure 6 also shows a tool displaying the internal tree structure for the error. Generating such displays using Ptree is discussed in Section 3.12.6.

3.12.5 Editing

Delete-@Level

Editing structured objects is quite simple in this version of Pan. All insertions use the text-level commands used for text editing. Top-down tree elaboration, being designed for the next major release, is not yet available. Deletions are accomplished using either text-level commands, or using the Delete-@Level command. Undoing edit actions restores the buffer without restoring the structured representation. Thus undoing currently requires reparsing.
Pan I: An Introduction For Users

3.12.6 Displaying Trees using Ptree

Syntax: Print Parse Tree

Ptree is a separate program that displays tree structures. A Ptree window appears in Figure 6. Ptree is not an integral part of Pan at present, although a Ptree-like viewer for Pan is planned. Ptree runs as a separate tool in the SunView environment. The manual page PTREE(1) provides more details on its operation. Ptree displays are primarily a debugging tool for the authors of Ladle language descriptions.

Pan can be used to create Ptree input. The command Print-Parse-Tree formats the internal tree structure of the active buffer into a file called ptree.out in the working directory. This file can then be displayed using Ptree.

3.13 When Things Go Wrong

Oops
Rats!

Pan has been remarkably (well, reasonably) robust throughout its long development period. Most problems are routinely handled by printing a message on the message line of the active viewer. However, during early-release, provisions have been made for recovering from major catastrophes.

We'd probably all agree that a catastrophe has occurred if Pan failed either by returning to the underlying Lisp system or by dying altogether. Fortunately, the first rarely happens, and the second won't occur without returning to Lisp.

If the option :break-to-lisp is true and an unanticipated error occurs, Pan enters a Lisp break loop. When Pan encounters an internal error and the option :save-on-system-error is true, Pan attempts to save all of the modified buffers. These two options are currently configured so that a break loop is entered, and saving is turned off. The break loop is entered before any saving of files is attempted.

If you do somehow end up in a Lisp break loop, a prompt will appear in the tool window in which the system is running. (If Pan is running from a menu, the prompt will appear in the console tool.) The prompt will look like “{nn}” or “nn ⇒” where “nn” is a small integer. In the first case (“{nn}”), the system is in a Lisp break loop. You can recover from the error by typing the Lisp expression (oops). This returns the system to the normal command evaluation loop. Naturally, the circumstances of such an error should be noted and passed on to the developers of Pan. (Mail to panpipes@renoir.)

The second case (“nn ⇒”) is more serious. In fact, the session is almost over. All that you can do is to type (Rats!). This command executes the normal code for saving the modified buffers and then exits the system.

4 Simple Customization

Pan can be customized by altering option values and bindings, extended by defining new options, flags, and commands, and broadened by defining new, formal languages by using the language
definition language *Ladle*. This section provides a brief introduction to the facilities for tailoring and extending the system.

4.1 Start-Up Processing

Auto-Load file-name regular-expression

Auto-Exec function-name regular-expression

Load-File

At start-up, a run-command file named .panrc is loaded into Pan. The .panrc file should be a file of Lisp and Pan commands located either in your working or your home directory.

The .panrc file is loaded using the command Load-File. All commands that load files use the search path specified by the option :pan-load-search-path. The default value of the option :pan-load-search-path is set to (.. "piper/lib/pan").

Pan can be instructed to automatically load files other than .panrc. One way is to include Load-File directives in the .panrc file. Such files will be loaded once at start-up. This method can be used to ensure that a certain selection of libraries will always be loaded. A second way is to use the Auto-Load command.

The Auto-Load command instructs Pan to ensure that a file has been loaded whenever a buffer whose name matches a given UNIX file expression is created. The file is loaded at most once as a result of Auto-Load.

For instance, (Auto-Load "c-lib" "*.hc") tells the system to load the file “c-lib” the first time that a file whose name matches “*.hc” is edited. The file c-lib can be either lisp code or compiled lisp code; if both c-lib.1 and a c-lib.o are found in the same directory, the most recently modified version is chosen.

Similar to Auto-Load, the command Auto-Execute instructs Pan to execute a given bindable function whenever a buffer having a name that matches a given pattern is created. Taken together, Auto-Load and Auto-Execute can be used to create minor modes.

A **minor mode** is a collection of commands and bindings useful while editing objects of a given type. For instance

(Auto-Load "tex-mode" "*.tex")

(Auto-Execute 'tex-mode "*.tex")

can be used to create a minor mode for TEX input. The parameterless function tex-mode is defined in the library file tex-mode.1 to set up a specialized collection of bindings. The file tex-mode.1 also defines a number of commands useful for manipulating TEXnical text.

4.2 Bindings

Bind-To-Key command key-sequence [location]

Bind-To-Menu command menu-name [label location]

Add-Menu-To-Menu menu-being-added to-menu [location]

Define-Operand-Levels levels

Add-Operand-Levels levels
Bindings are established or altered using the functions Bind-To-Key, Bind-To-Menu, and Add-Menu-To-Menu. The default location of the change is in the local buffer; specifying :global makes the change in the global environment. The commands Add-Menu-To-Menu, Define-Operand-Level, and Add-Operand-Level are described in The Pan Extension Manual.

The syntax for Bind-To-Key is

\[(\text{Bind-To-Key } \text{'function "key sequence" [location]})\]

where ’function is a quoted function name, “key sequence” is a Lisp string specifying a keystroke sequence, and location is either :local or :global. The default value for location is :local.

When specifying a keystroke sequence, control characters such as “Control-?” are denoted by the two-character sequence “?”; the Escape prefix is denoted by Esc-. In multiple keystroke sequences, the keystrokes must be separated by hyphens. For example,

\[(\text{Bind-To-Key } \text{'Delete-Character "^D" :global})\]

establishes the default binding for Delete-Character.

To establish a binding to a menu item, use

\[(\text{Bind-To-Menu } \text{'function "Menu Name" ["Selection Name" location]})\]

Both parts of the menu item must be Lisp strings. If “Selection Name” is omitted, the name of the command is used. The argument location is identical to the argument of the same name for the Bind-To-Key command. For example,

\[(\text{Bind-To-Menu } \text{'Describe-Selection "Help" "Describe" :global})\]

sets up the default binding for Describe-Selection in the Help menu.

Show-Key-Bindings Help: Key Bindings
Show-Menu-Bindings Help: Menu Bindings
Show-Operand-Bindings Help: Operand Bindings

Show-Key-Bindings, Show-Menu-Bindings, and Show-Operand-Bindings print the appropriate bindings visible in the current buffer into the help buffer.

It is possible to invoke a command without binding it by using the commands Execute-Lisp-Line or Execute-Named-Command. (The next section provides more details.)

### 4.3 Getting and Setting Option Values

Option-Value option-name [lookup buffer]

The command Option-Value is used to retrieve (and set) option values. Its syntax is

\[(\text{Option-Value } \text{option-name [lookup buffer]})\]
where option-name is the name of the option, and lookup and buffer are optional arguments. The argument lookup, if specified, must be one of :local, :global, or :default. When lookup is left unspecified, the value :default (look first for a definition in the active buffer; if none is found, look for a global definition) is used. Buffer defaults to the active buffer.

To set an option’s value to value, use the lisp form

(setf (Option-Value option-name [lookup buffer]) value)

In this case, lookup must specify either :local or :global; it defaults to :local. For instance, the expression

(setf (Option-Value :minor-mode :local) "TeX Mode")

sets the value of the option :minor-mode in the active buffer to be "TeX Mode".

4.4 Lisp-Oriented Commands

These commands are for the use of people extending Pan, although sometimes you’ll want to use one to see what a command does. They are included here for completeness.

<table>
<thead>
<tr>
<th>Command</th>
<th>Key Combination</th>
</tr>
</thead>
<tbody>
<tr>
<td>Execute-Lisp-Line</td>
<td>Esc-Esc</td>
</tr>
<tr>
<td>Execute-Named-Command</td>
<td>Esc-x</td>
</tr>
<tr>
<td>Load-File</td>
<td>&quot;X&quot;-&quot;L&quot;</td>
</tr>
</tbody>
</table>

Execute-Lisp-Line prompts for a Lisp expression to evaluate; the result is printed on the message line. If you want to execute a bindable Pan command, there are two methods. The simplest is to invoke Execute-Named-Command and respond to the prompt with the name of the command, e.g. Next-Character. Alternatively, one can invoke Execute-Lisp-Line and respond to the prompt with the expression (Next-Character). The parentheses are required in the latter case.

Load-File loads a file of lisp and Pan commands into the system.

5 Acknowledgments

Many have helped with the creation of Pan. Thanks especially to Jacob Butcher and Christina Black. Jacob implemented the language-description processor Ladle and the tree data structures. Christina is developing the pretty-printing viewer and helped with the preparation of these reports. Eduardo Pelegri-Llopart and Phillip Garrison have also made many valuable suggestions.

References

6 Glossary

**Apropos**  A Help command for information gathered during command, flag, and option definition. For instance, (`Apropos 'Cursor`) lists all of the commands dealing with the cursor.

**Base Buffer**  The base buffer is a special buffer that is the root for all Pan buffers. The base buffer has is the only frame that can be made iconic. Quitting the base buffer terminates Pan. All currently editable objects are listed in the base buffer.

**Buffer**  A buffer is the locus of editing attention for a single editable object (currently a text file). Buffers contain a copy of the object being edited, key and menu bindings, a selection, viewers, and other objects.

**Clipboard**  The clipboard is an area, shared by all buffers, that contains a region of text. It is used to implement cut and paste between buffers, or between Pan and other processes.

**Command**  A command is a user-level procedure for effecting edit operations. Commands are defined using `Define-Command`.

**Dot**  The dot is another name for the edit cursor. In effect, it is the integer offset (in characters) of the edit cursor from the beginning of the file.

**Edit Cursor**  A cursor is an object marking the location where alterations of the edit object can occur. In particular, characters are inserted or deleted at the character position to the left of the character designated by the cursor.

**Edit Window**  The area of a viewer in which the object being edited is displayed.

**Flag**  A flag is a user-definable object that stores a single bit of information. Flags can be defined having a user-visible representation on the information panel.
Frame  The outer surrounding edge of a viewer that responds to SUNVIEW window protocols.

Help Buffer  A special buffer used for displaying help information.

Mark  A mark is a character position in a text file. Marks are generally used to remember positions for later processing. Pan provides for a stack of marks. The top mark in the stack is known as “the mark”.

Operand Level  The operand level of a viewer designates the type of operand to be used by generic operations. For instance, if the operand level is “Word”, then the Next-@Level command moves the cursor to the next word in the object being edited.

Option  An option is a user-definable typed variable. Many of the customizations available to a user are provide via predefined options. Unlike a flag, an option does not have a visible presentation. However, options can have special “notifier” functions that are called whenever the option value is changed.

Region  A region is a contiguous sequence of characters. Most text operations involve regions either as source, destination, or both.

Selection  A specially designated region. There is one selection per buffer.

Viewer  The counterpart to a window in Emacs, a viewer displays an object. Each viewer has its own edit cursor and display state.
# Default Key Bindings

## A.1 Bindings By Command Name

<table>
<thead>
<tr>
<th>Command</th>
<th>Key Bindings</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abort-Command</td>
<td>Ctrl-G</td>
</tr>
<tr>
<td>Abort-Command</td>
<td>Ctrl-Right</td>
</tr>
<tr>
<td>Abort-Command</td>
<td>Ctrl-Left</td>
</tr>
<tr>
<td>Abort-Command</td>
<td>Ctrl-Shift</td>
</tr>
<tr>
<td>Backward-Expr</td>
<td>Ctrl-End</td>
</tr>
<tr>
<td>Capitalize-Selection</td>
<td>Ctrl-End</td>
</tr>
<tr>
<td>Default Key Bindings</td>
<td>Left-Hscroll</td>
</tr>
<tr>
<td>List-Files</td>
<td>Ctrl-D</td>
</tr>
<tr>
<td>Load-File</td>
<td>Ctrl-L</td>
</tr>
<tr>
<td>Lowercase-Selection</td>
<td>Ctrl-L</td>
</tr>
<tr>
<td>X-</td>
<td>Mouse-Extend-OLevel</td>
</tr>
<tr>
<td>X-</td>
<td>Mouse-Select-OLevel</td>
</tr>
<tr>
<td>X-</td>
<td>Mouse-Extend-Selection-Fullword</td>
</tr>
<tr>
<td>X-</td>
<td>Mouse-Select-Fullword</td>
</tr>
<tr>
<td>Esc-B</td>
<td>Move-Selection-To-Cursor</td>
</tr>
<tr>
<td>Esc-v</td>
<td>Move-To-BOB</td>
</tr>
<tr>
<td>Esc-c</td>
<td>Move-To-BOL</td>
</tr>
<tr>
<td>X-</td>
<td>Move-To-EOB</td>
</tr>
<tr>
<td>Esc-w</td>
<td>Move-To-EOL</td>
</tr>
<tr>
<td>Esc-y</td>
<td>Newline-And-Indent</td>
</tr>
<tr>
<td>Esc-\</td>
<td>Previous-OLevel</td>
</tr>
<tr>
<td>Esc-\</td>
<td>Previous-Line</td>
</tr>
<tr>
<td>Esc-y</td>
<td>Paste-From-Clipboard</td>
</tr>
<tr>
<td>L6</td>
<td>Next-Line</td>
</tr>
<tr>
<td>L10</td>
<td>Next-Word</td>
</tr>
<tr>
<td>Esc-F</td>
<td>Previous-Word</td>
</tr>
<tr>
<td>Esc-\</td>
<td>Previous-Word</td>
</tr>
<tr>
<td>Backspace</td>
<td>Re-Search-Backward</td>
</tr>
<tr>
<td>Del</td>
<td>Re-Search-Forward</td>
</tr>
<tr>
<td>Esc-\D</td>
<td>Read-Prefix-Arguments</td>
</tr>
<tr>
<td>F3</td>
<td>Redraw</td>
</tr>
<tr>
<td>Mouse-Right</td>
<td>Replace-From-Clipboard</td>
</tr>
<tr>
<td>Esc-\x</td>
<td>Right-Hscroll</td>
</tr>
<tr>
<td>Esc-m</td>
<td>Save-All-Buffers</td>
</tr>
<tr>
<td>Esc-\F</td>
<td>Save-Buffer-File</td>
</tr>
<tr>
<td>V</td>
<td>Select-Buffer</td>
</tr>
<tr>
<td>\x-</td>
<td>Select-Expr</td>
</tr>
<tr>
<td>\x-</td>
<td>Select-Region-Dot-To-Mark</td>
</tr>
<tr>
<td>\x-</td>
<td>Select-Word</td>
</tr>
<tr>
<td>Return</td>
<td>Self-Insert</td>
</tr>
<tr>
<td>Esc-()</td>
<td>Self-Insert</td>
</tr>
<tr>
<td>Insert-File</td>
<td>Space-(</td>
</tr>
<tr>
<td>Insert-Parentheses</td>
<td>Tab</td>
</tr>
<tr>
<td>Insert-Rbrace-And-Match</td>
<td>\x-</td>
</tr>
<tr>
<td>Insert-Rbracket-And-Match</td>
<td>\x-</td>
</tr>
<tr>
<td>Insert-Rparen-And-Match</td>
<td>\x-</td>
</tr>
<tr>
<td>Interrupt-Pan</td>
<td>Tab</td>
</tr>
<tr>
<td>Just-One-Space</td>
<td>\x-</td>
</tr>
<tr>
<td>Kill-Expr</td>
<td>Space-(</td>
</tr>
<tr>
<td>Kill-Previous-Word</td>
<td>Tab</td>
</tr>
<tr>
<td>Kill-Selected-Region</td>
<td>\x-</td>
</tr>
<tr>
<td>Kill-To-EOL</td>
<td>\x-</td>
</tr>
<tr>
<td>Kill-Word</td>
<td>\x-</td>
</tr>
<tr>
<td>Command</td>
<td>Keypress</td>
</tr>
<tr>
<td>-------------------------</td>
<td>------------------</td>
</tr>
<tr>
<td>Show-Match</td>
<td>\textasciicircum c-%</td>
</tr>
<tr>
<td>Show-Status-Line</td>
<td>\textasciicircum x-=</td>
</tr>
<tr>
<td>Split-Line</td>
<td>\textasciicircum c-\textasciitilde o</td>
</tr>
<tr>
<td>Sun-Again</td>
<td>L2</td>
</tr>
<tr>
<td>Sun-Expose</td>
<td>L5</td>
</tr>
<tr>
<td>Sun-Find</td>
<td>L9</td>
</tr>
<tr>
<td>Sun-Open</td>
<td>L7</td>
</tr>
<tr>
<td>Sun-Props</td>
<td>L3</td>
</tr>
<tr>
<td>Sun-Stop</td>
<td>L1</td>
</tr>
<tr>
<td>Swap-Dot-And-Mark</td>
<td>\textasciicircum x-\textasciitilde x</td>
</tr>
<tr>
<td>Toggle-Auto-Fill</td>
<td>\textasciicircum x-\textasciitilde a</td>
</tr>
<tr>
<td>Toggle-Read-Only</td>
<td>\textasciicircum x-\textasciitilde q</td>
</tr>
<tr>
<td>Transpose-Characters</td>
<td>\textasciitilde t</td>
</tr>
<tr>
<td>Transpose-Lines</td>
<td>\textasciicircum x-\textasciitilde t</td>
</tr>
<tr>
<td>Undo</td>
<td>L4</td>
</tr>
<tr>
<td>Undo</td>
<td>\textasciicircum x-u</td>
</tr>
<tr>
<td>Up-@Level</td>
<td>F2</td>
</tr>
<tr>
<td>Uppercase-Selection</td>
<td>\textasciicircum c-\textasciitilde u</td>
</tr>
<tr>
<td>Uppercase-Word</td>
<td>\textasciicircum u</td>
</tr>
<tr>
<td>Visit-Buffer</td>
<td>\textasciicircum x-b</td>
</tr>
<tr>
<td>Visit-File</td>
<td>\textasciicircum x-f</td>
</tr>
<tr>
<td>Write-Files-Exit</td>
<td>\textasciicircum x-c</td>
</tr>
<tr>
<td>Write-To-File</td>
<td>\textasciicircum x-w</td>
</tr>
<tr>
<td>Yank-From-Kill-Ring</td>
<td>\textasciicircum y</td>
</tr>
</tbody>
</table>
A.2 Bindings By Key

- A  Set-Mark -X-u
- B  Previous-@Level -Y
- C-|  Abort-Command -Z-^G
- D  Delete-@Level -|
- E  Move-To-EOL -
- F  Next-@Level *--\n- G  Abort-Command Backspace
- K  Kill-To-EOL Del
- L  Redraw Esc-! 
- N  Next-Line Esc-(
- O  Open-Line Esc-<
- P  Previous-Line Esc->
- Q  Quote-Insert Esc-?
- R  Re-Search-Backward Esc-@
- S  Re-Search-Forward Esc-Del
- T  Transpose-Characters Esc-Esc
- U  Read-Prefix-Arguments Esc-L8
- V  Forward-Vscroll Esc-Mouse_Left
- W  Kill-Selected-Region Esc-Mouse_Middle
- X-!  Cycle-Show-Kill Mouse-Extend-Selection-Fullword
- X-0  Close-Active-Viewer Move-Selection-To-Cursor
- X-2  Open-Another-Viewer indent-Like-Previous-Line
- X-<  Show-Status-Line Show-Match
- X-=  Left-Hscroll Select-Expr
- X->  Right-Hscroll Backward-Expr
- X-?  Show-Kill Backward-Vscroll
- X-Return  Save-All-Buffers Capitalize-Selection
- X-Tab  Insert-File Deselect-Region
- X-"A  Toggle-Auto-Fill Forward-Expr
- X-"B  Show-Buffers Abort-Command
- X-"C  Write-Files-Exit Kill-Expr
- X-"D  List-Files Lowercase-Selection
- X-"F  Visit-File Split-Line
- X-"G  Abort-Command Uppercase-Selection
- X-"L  Load-File Select-Region-Dot-To-Mark
- X-"O  Delete-Blank-Lines Copy-Selection-To-Cursor
- X-"Q  Toggle-Read-Only Delete-Indentation
- X-"S  Save-Buffer-File Previous-Word
- X-"T  Transpose-Lines Capitalize-Word
- X-"W  Write-To-File Kill-Word
- X-"X  Swap-Dot-And-Mark Next-Word
- X-"Y  Interrupt-Pan Lowercase-Word
- X-b  Visit-Buffer First-Non-Blank
- X-f  Set-Auto-Fill-Column Uppercase-Word
- X-h  Select-Buffer Backward-Vscroll
- X-k  Remove-Buffer Copy-Selection-As-Vscroll
- X-l  Goto-Line
**Esc-x**  Execute-Named-Command
**Esc-y**  Cycle-Yank
**F1**  Set-@Level-To-Character
**F2**  Up-@Level
**F3**  Down-@Level
**L10**  Cut-To-Clipboard
**L1**  Sun-Stop
**L2**  Sun-Again
**L3**  Sun-Props
**L4**  Undo
**L5**  Sun-Expose
**L6**  Copy-To-Clipboard
**L7**  Sun-Open
**L8**  Paste-From-Clipboard
**L9**  Sun-Find
**Mouse_Left**  Mouse-Select-@Level
**Mouse_Middle**  Mouse-Extend-@Level
**Mouse_Right**  Execute-From-Menu
**Newline**  Newline-And-Indent
**Return**  Insert-Newline
**Space-(**  Self-Insert
**Tab**  Self-Insert
B Default Menu Bindings in a Text Buffer

B.1 Bindings by Command Name

Append-Selection-To-File .................. File: Append Selection To ...
Apropos-All-Symbols ...................... Help: Apropos All
Apropos-Selection .......................... Help: Apropos
Close-Active-Viewer ...................... Window: Close
Copy-Selection-To-Cursor ................. Edit: Copy To Cursor
Copy-To-Clipboard .......................... Edit: Copy
Cut-To-Clipboard ........................... Edit: Cut
Delete-Selected-Region .................... Edit: Delete
Describe-Operand-Hierarchy .............. Help: Operand Hierarchy
Describe-Selection ....................... Window: Frame Edit Cursor
Insert-File .............................. File: Insert from...
Kill-Selected-Region ...................... Edit: Kill
List-All-Commands ....................... Help: Commands
List-Files .................................. File: Directory list
Move-Selection-To-Cursor ............... Edit: Move To Cursor
Open-Another-Viewer ..................... Window: Open Another Viewer
Overwrite-File! .......................... File: Overwrite ...
Paste-From-Clipboard ..................... Edit: Paste
Reset-Help-Buffer ....................... Help: Reset Help Buffer
Save-Buffer-File .......................... File: Save
Show-Buffers ............................. Help: Buffers
Show-ClipBoard ............................ Edit: Show Clipboard
Show-Flag-Values .......................... Help: Flag Values
Show-Key-Bindings ....................... Help: Key Bindings
Show-Menu-Bindings ...................... Help: Menu Bindings
Show-Operand-Bindings ................... Help: Operand Bindings
Show-Option-Values ...................... Help: Option Values
Undo .................................. Pan: Undo
Visit-File ................................. File: Visit File
Write-Selection-To-File ................. File: Write Selection To ...
Write-To-File ............................. File: Write To ...

B.2 Bindings by Menu

Edit: Copy To Cursor ..................... Copy-Selection-To-Cursor
Edit: Copy ................................ Copy-To-Clipboard
Edit: Cut ................................ Cut-To-Clipboard
Edit: Delete ........................... Delete-Selected-Region
Edit: Kill ................................ Kill-Selected-Region
Edit: Move To Cursor .................... Move-Selection-To-Cursor
Edit: Paste ............................... Paste-From-Clipboard
Edit: Show Clipboard .................... Show-ClipBoard
File: Append Selection To ............. Append-Selection-To-File
File: Directory list ........................ List-Files
C Default Menu Bindings in the Base Buffer

C.1 Bindings by Command Name

Apropos-All-Symbols ............................................. Help: Apropos All
Apropos-Selection ................................................ Help: Apropos
Copy-To-Clipboard ............................................... Base Buffer: Copy
Describe-Operand-Hierarchy ................................... Help: Operand Hierarchy
Describe-Selection ............................................... Help: Describe
List-All-Commands ................................................ Help: Commands
List-Files .......................................................... Base Buffer: Directory list
Remove-Selected-Buffer .......................................... Base Buffer: Remove-Selected-Buffer
Reset-Help-Buffer ................................................ Help: Reset Help Buffer
Show-Buffers ....................................................... Help: Buffers
Show-Flag-Values ................................................ Help: Flag Values
Show-Key-Bindings ............................................... Help: Key Bindings
Show-Menu-Bindings .............................................. Help: Menu Bindings
Show-Operand-Bindings .......................................... Help: Operand Bindings
Show-Option-Values .............................................. Help: Option Values
Undo ................................................................. Base Buffer: Undo
Visit-File .......................................................... Base Buffer: Visit File
Visit-Selection .................................................... Base Buffer: Visit
C.2 Bindings by Menu

Base Buffer: Copy ....................................................... Copy-To-Clipboard
Base Buffer: Directory list .......................... List-Files
Base Buffer: Remove-Selected-Buffer .............. Remove-Selected-Buffer
Base Buffer: Undo .................................................. Undo
Base Buffer: Visit File ................................. Visit-File
Base Buffer: Visit .................................................. Visit-Selection
Help: Apropos All ........................................ Apropos-All-Symbols
Help: Apropos ................................................... Apropos-Selection
Help: Buffers ......................................................... Show-Buffers
Help: Commands ................................................. List-All-Commands
Help: Describe ..................................................... Describe-Selection
Help: Flag Values .............................................. Show-Flag-Values
Help: Key Bindings ........................................... Show-Key-Bindings
Help: Menu Bindings ........................................ Show-Menu-Bindings
Help: Operand Bindings ................................... Show-Operand-Bindings
Help: Operand Hierarchy ......................... Describe-Operand-Hierarchy
Help: Option Values ........................................ Show-Option-Values
Help: Reset Help Buffer .......................... Reset-Help-Buffer

D Default Menu Bindings in the Help Buffer

D.1 Bindings by Command Name

Append-Selection-To-File .......................... File: Append Selection To
Apropos-All-Symbols .................................. Help: Apropos All
Apropos-Selection ....................................... Help Buffer: Apropos
Apropos-Selection ....................................... Help: Apropos
Close-Active-Viewer .................................. Window: Close
Copy-To-Clipboard ....................................... Help Buffer: Copy
Describe-Operand-Hierarchy ....................... Help: Operand Hierarchy
Describe-Selection ....................................... Help Buffer: Describe
Describe-Selection ....................................... Help: Describe
Frame-Edit-Cursor ...................................... Window: Frame Edit Cursor
List-All-Commands ........................................ Help: Commands
List-Files ....................................................... File: Directory list
Open-Another-Viewer .................................. Window: Open Another Viewer
Overwrite-File! ............................................... File: Overwrite
Reset-Help-Buffer ....................................... Help Buffer: Reset Help Buffer
Reset-Help-Buffer ....................................... Help: Reset Help Buffer
Show-Buffers ............................................... Help: Buffers
Show-Flag-Values ......................................... Help: Flag Values
Show-Key-Bindings ..................................... Help: Key Bindings
Show-Menu-Bindings .................................. Help: Menu Bindings
Show-Operand-Bindings ......................... Help: Operand Bindings
Show-Option-Values ................................... Help: Option Values
Undo .......................................................... Help Buffer: Undo
Visit-File ......................................................... File: Visit File
Visit-Selection ........................................... File: Visit
Write-Selection-To-File ................................ File: Write Selection To ...
Write-To-File .............................................. File: Write To ...

D.2 Bindings by Menu

File: Append Selection To .................................. Append-Selection-To-File
File: Directory list ........................................ List-Files
File: Overwrite ............................................. Overwrite-File!
File: Visit File ............................................. Visit-File
File: Visit .................................................. Visit-Selection
File: Write Selection To .................................... Write-Selection-To-File
File: Write To ............................................... Write-To-File
Help Buffer: Apropos ...................................... Apropos-Selection
Help Buffer: Copy ........................................... Copy-To-Clipboard
Help Buffer: Describe ..................................... Describe-Selection
Help Buffer: Reset Help Buffer .......................... Reset-Help-Buffer
Help Buffer: Undo .......................................... Undo
Help: Apropos All .......................................... Apropos-All-Symbols
Help: Apropos ............................................... Apropos-Selection
Help: Buffers .............................................. Show-Buffers
Help: Commands ........................................... List-All-Commands
Help: Describe ............................................. Describe-Selection
Help: Flag Values ......................................... Show-Flag-Values
Help: Key Bindings ........................................ Show-Key-Bindings
Help: Menu Bindings ....................................... Show-Menu-Bindings
Help: Operand Bindings .................................... Show-Operand-Bindings
Help: Operand Hierarchy ................................... Describe-Operand-Hierarchy
Help: Option Values ....................................... Show-Option-Values
Help: Reset Help Buffer ................................... Reset-Help-Buffer
Window: Close ............................................... Close-Active-Viewer
Window: Frame Edit Cursor ................................ Frame-Edit-Cursor
Window: Open Another Viewer ............................ Open-Another-Viewer

E Options Defined in Pan

:auto-fill
Type: boolean
Default Value: nil
T iff Space & Newline should cause long lines to be broken.

:auto-fill-column
Type: fixp
Default Value: 65
Max line length for auto-fill.

:autoindent
Type: boolean
Default Value: \texttt{t}
T if Newline-And-Indent should indent.

\textbf{:autowrap-search}
Type: boolean
Default Value: \texttt{t}
Wrap search from one end of file to other?

\textbf{:backup-on-read}
Type: boolean
Default Value: \texttt{t}
Make backup file when beginning to edit a file

\textbf{:base-frame-font-map}
Type: listp
Default Value: \texttt{("cour. b. 14")}
Font map for base window.

\textbf{:base-win-cols}
Type: fixp
Default Value: 60
Default number of columns in base window.

\textbf{:base-win-rows}
Type: fixp
Default Value: 6
Default number of rows in base window.

\textbf{:break-to-lisp}
Type: boolean
Default Value: \texttt{t}
Break to underlying LISP on tragic error?.

\textbf{:camel-debug}
Type: boolean
Default Value: \texttt{nil}
Turn on internal parser tracing.

\textbf{:checkpoint-freq}
Type: fixp
Default Value: 450
Maximum number of command invocations before checkpointing.

\textbf{:checkpoint-min}
Type: fixp
Default Value: 400
Minimum number of command invocations prior to checkpointing.

\textbf{:clipboard-max-size}
Type: fixp
Default Value: 8
Maximum number of clips in clipboard
:col-popover
  Type: fixp
  Default Value: 16
  Minimum characters to move window when scrolling left/right.

:dont-reuse-nodes
  Type: boolean
  Default Value: t
  Turn off node reuse during incremental parsing.

:empty-line-char
  Type: fixp
  Default Value: 46
  Character prefix for empty lines.

:full-parse-tree
  Type: boolean
  Default Value: nil
  Create full parse tree during parsing?

:help-frame-font-map
  Type: listp
  Default Value: ("cour.r.12")
  Font map for help windows.

:help-title
  Type: stringp
  Default Value: "Help window"
  Title for help window

:help-win-cols
  Type: fixp
  Default Value: 60
  Default number of columns in help window.

:help-win-rows
  Type: fixp
  Default Value: 20
  Default number of rows in help window.

:indent-with-tabs
  Type: boolean
  Default Value: nil
  T means use tabs when performing computed indentations

:indentation-chars
  Type: stringp
  Default Value: " "
  List of characters to skip over during autoindent.

:iparse-debug
  Type: boolean
  Default Value: nil
Turn on incremental parser tracing.

**:killring-max-size**
Type: **fixp**
Default Value: 16
Maximum number of kills retained in kill ring.

**:kills-to-clipboard**
Type: **boolean**
Default Value: **nil**
If true, the clipboard is used instead of the local kill ring in kill commands.

**:is-flags**
Type: **stringp**
Default Value: "-IF"
Flags for /bin/is command use in List-Files

**:minor-mode**
Type: **stringp**
Default Value: "Normal"
Name for local binding set.

**:mode-line-fmt**
Type: **stringp**
Default Value: "File %F %* %C %w Mode: %M"
Format string for status information.

**:pan-load-search-path**
Type: **listp**
Default Value: ("/" "piper/lib/pan")
Search path used by Load-File command.

**:pause-ticks**
Type: **fixp**
Default Value: 250
Constant multiplier for Pause command.

**:proportional-scroll**
Type: **boolean**
Default Value: t
Vertical scroll proportionally, else fixed screenful at a time.

**:region-highlight-op**
Type: (**underline** **invert**)
Default Value: **underline**
Operation for highlighting: **underline** or **invert**.

**:row-popup**
Type: **fixp**
Default Value: 4
Minimum number rows to move window when scrolling up/down.
An Introduction For Users

**:save-on-system-error**
Type: boolean
Default Value: nil
Save all buffers when error is encountered?

**:show-node-reuse**
Type: boolean
Default Value: nil
Turn on tracing for incremental parsing node reuse

**:syntax-classes**
Type: listp
Default Value: (:word-char :space-char :punct-char :lbracket :rbracket :other)
List of valid syntax classes.

**:tabwidth**
Type: fixp
Default Value: 8
Number of characters per tab.

**:text-font-map**
Type: listp
Default Value: ("screen.r.12" "screen.b.12" "serif.r.12" "cour.r.12" "cour.b.12")
Font map for text viewers.

**:verbose-load**
Type: boolean
Default Value: nil
T iff verbose file loading is desired.

**:verbose-parse**
Type: boolean
Default Value: nil
T iff verbose parsing information is desired.

**:verbose-trace**
Type: boolean
Default Value: nil
T iff verbose tracing mode is desired.

**:visible-flags**
Type: check-flags
Default Value: (:text-modified :read-only)
Flags visible on control panel.

**:win-cols**
Type: fixp
Default Value: 78
Default number of columns per window.

**:win-cols-min**
Type: fixp
Default Value: 10
Minimum number of columns in window; guards against user resizing.

```
:win-rows
Type: fixp
Default Value: 30
Default number of rows per window.
```

```
:win-rows-min
Type: fixp
Default Value: 2
Minimum number of rows in window; guards against user resizing.
```

```
:wrap-Qlevel
Type: boolean
Default Value: t
Automatically wrap operand hierarchy from top to bottom, or from bottom to top.
```

```
:zero-index-lines
Type: boolean
Default Value: t
If T, then arguments to Goto-Line are interpreted as 0-indexed. Otherwise, the arguments are interpreted as 1-indexed.
```

### F Flags Defined in Pan

```
:auto-exec
Presentation: "E"
Behaviour: :invisible-when-cleared
Set when auto-execution related to file suffix has occurred.
```

```
:blank-flag
Presentation: " "
Behaviour: :invisible-when-cleared
Dummy flag for padding flag array.
```

```
:lex-ok
Presentation: "L"
Behaviour: :gray-when-cleared
Set when language file has been scanned.
```

```
:panic
Presentation: "P"
Behaviour: :invisible-when-cleared
Set when panic-mode error recovery has been invoked.
```

```
:parse-errors
Presentation: "!
Behaviour: :invisible-when-cleared
Set when there are errors in the parse tree.
```

```
:re-search-successful
Presentation: "9"
Behaviour: :invisible-when-cleared
```
Set if the last regular expression search was successful

:read-only
Presentation: "$"
Behaviour: invisible-when-cleared
Set when buffer file cannot be written.

:text-modified
Presentation: "*"
Behaviour: invisible-when-cleared
Set when buffer text has been modified but not saved.

:tree-ok
Presentation: "T"
Behaviour: gray-when-cleared
Set when language file has been parsed.
G Pan for GNU Emacs Users

Since Pan's text-oriented facilities are modeled on the emacs family of text editors, users familiar with emacs will find much that is familiar. This appendix will help emacs users get started with Pan as conveniently as possible.

This appendix compares Pan version 1.9 and the emacs editor in use locally, GNU Emacs Version 18. The comparison reveals compatibility between the two as text editors, but does not pretend to give a comprehensive picture of the functionality of either system. Consult the main body of this manual for a more thorough introduction to Pan.

Pan also provides for manipulating and editing programs using the syntax and semantics of the language being edited. This structure-based, description-driven facility is fundamentally more powerful than the language modes supported by emacs, but is not discussed in this appendix.

Finally, Pan is extensible and customizable in the spirit of emacs, but techniques for doing so are beyond the scope of this appendix.

G.1 Key Bindings

Both Pan and GNU Emacs use keymaps, a dynamic mechanism for binding editor commands to keystrokes (or keystroke sequences). A global keymap is always present during an editing session, but may be effectively extended and altered by a local keymap associated with each buffer being edited. Local keymaps are typically created as part of special editing modes (see G.8, "Special Editing Modes").

This section compares only default key bindings in the global keymap; these are the bindings normally in effect for ordinary text editing. Keymaps, both global and local, may be easily customized in both editors; both editors support many commands that are not bound in the default keymap.

The key bindings currently in effect for a Pan buffer may be displayed by invoking the "Key Bindings" command from the help menu.

Columns 1 and 2 of this list are derived from the global default key bindings in the local version of GNU Emacs. Column 3 identifies Pan's compatibility with one of the following symbols:

- some equivalent supported now
- some equivalent anticipated, but unimplemented
- now bound to something else in Pan, as noted
- blank no equivalent supported in Pan, contributions welcome

and column 4 lists the Pan command binding. Many potential key stroke sequences remain unbound in both editors, and there are many commands to which no sequences are bound. A list of all available Pan commands may be displayed by invoking the "Commands" command from the help menu.
<table>
<thead>
<tr>
<th>Key</th>
<th>GNU Emacs Binding</th>
<th>Code</th>
<th>Pan Binding</th>
</tr>
</thead>
<tbody>
<tr>
<td>C-@</td>
<td>set-mark-command</td>
<td></td>
<td>Set-Mark</td>
</tr>
<tr>
<td>C-a</td>
<td>beginning-of-line</td>
<td></td>
<td>Move-To-BOL</td>
</tr>
<tr>
<td>C-b</td>
<td>backward-char</td>
<td></td>
<td>Previous-@Level</td>
</tr>
<tr>
<td>C-c</td>
<td>mode-specific-command-prefix</td>
<td></td>
<td>Command prefix</td>
</tr>
<tr>
<td>C-d</td>
<td>delete-char</td>
<td></td>
<td>Delete-@Level</td>
</tr>
<tr>
<td>C-e</td>
<td>end-of-line</td>
<td></td>
<td>Move-To-EOL</td>
</tr>
<tr>
<td>C-f</td>
<td>forward-char</td>
<td></td>
<td>Next-@Level</td>
</tr>
<tr>
<td>C-g</td>
<td>keyboard-quit</td>
<td></td>
<td>Abort-Command</td>
</tr>
<tr>
<td>C-h</td>
<td>help-command</td>
<td></td>
<td>Delete-Previous-Character</td>
</tr>
<tr>
<td>TAB</td>
<td>indent-for-tab-command</td>
<td></td>
<td>Self-Insert</td>
</tr>
<tr>
<td>LFD</td>
<td>newline-and-indent</td>
<td></td>
<td>Newline-And-Indent</td>
</tr>
<tr>
<td>C-k</td>
<td>kill-line</td>
<td></td>
<td>Kill-To-EOL</td>
</tr>
<tr>
<td>C-l</td>
<td>recenter</td>
<td></td>
<td>Redraw</td>
</tr>
<tr>
<td>RET</td>
<td>newline</td>
<td></td>
<td>Insert-Newline</td>
</tr>
<tr>
<td>C-n</td>
<td>next-line</td>
<td></td>
<td>Next-Line</td>
</tr>
<tr>
<td>C-o</td>
<td>open-line</td>
<td></td>
<td>Open-Line</td>
</tr>
<tr>
<td>C-p</td>
<td>previous-line</td>
<td></td>
<td>Previous-Line</td>
</tr>
<tr>
<td>C-q</td>
<td>quoted-insert</td>
<td></td>
<td>Quote-Insert</td>
</tr>
<tr>
<td>C-r</td>
<td>isearch-backward</td>
<td></td>
<td>Re-Search-Backward</td>
</tr>
<tr>
<td>C-s</td>
<td>isearch-forward</td>
<td></td>
<td>Re-Search-Forward</td>
</tr>
<tr>
<td>C-t</td>
<td>transpose-chars</td>
<td></td>
<td>Transpose-Characters</td>
</tr>
<tr>
<td>C-u</td>
<td>universal-argument</td>
<td></td>
<td>Read-Prefix-Arguments</td>
</tr>
<tr>
<td>C-v</td>
<td>scroll-up</td>
<td></td>
<td>Forward-Vscroll</td>
</tr>
<tr>
<td>C-w</td>
<td>kill-region</td>
<td></td>
<td>Kill-Selected-Region</td>
</tr>
<tr>
<td>C-x</td>
<td>Control-X-prefix</td>
<td></td>
<td>Command prefix</td>
</tr>
<tr>
<td>C-y</td>
<td>yank</td>
<td></td>
<td>Yank-From-Kill-Ring</td>
</tr>
<tr>
<td>C-z</td>
<td>suspend-emacs</td>
<td></td>
<td>Command prefix</td>
</tr>
<tr>
<td>ESC</td>
<td>ESC-prefix</td>
<td></td>
<td>Command prefix</td>
</tr>
<tr>
<td>C-]</td>
<td>abort-recursive-edit</td>
<td></td>
<td></td>
</tr>
<tr>
<td>C_</td>
<td>undo</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SPC ..</td>
<td>self-insert-command</td>
<td></td>
<td>Insert-Previous-Character</td>
</tr>
<tr>
<td>DEL</td>
<td>delete-backward-char</td>
<td></td>
<td></td>
</tr>
<tr>
<td>C-x C-a</td>
<td>add-mode-abbrev</td>
<td></td>
<td>Toggle-Auto-Fill</td>
</tr>
<tr>
<td>C-x C-b</td>
<td>list-buffers</td>
<td></td>
<td>Show-Buffers</td>
</tr>
<tr>
<td>C-x C-c</td>
<td>save-buffers-kill-emacs</td>
<td></td>
<td>Write-Files-Exit</td>
</tr>
<tr>
<td>C-x C-d</td>
<td>list-directory</td>
<td></td>
<td>List-Files</td>
</tr>
<tr>
<td>C-x C-e</td>
<td>eval-last-sexp</td>
<td></td>
<td></td>
</tr>
<tr>
<td>C-x C-f</td>
<td>find-file</td>
<td></td>
<td>Visit-File</td>
</tr>
<tr>
<td>C-x C-h</td>
<td>inverse-add-mode-abbrev</td>
<td></td>
<td></td>
</tr>
<tr>
<td>C-x TAB</td>
<td>indent-rigidly</td>
<td></td>
<td>Insert-File</td>
</tr>
<tr>
<td>C-x LFD</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C-x C-k</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C-x C-l</td>
<td>downcase-region [see Esc C-l]</td>
<td></td>
<td>Load-file</td>
</tr>
<tr>
<td>C-x RET</td>
<td></td>
<td></td>
<td>Save-All-buffers</td>
</tr>
<tr>
<td>C-x C-n</td>
<td>set-goal-column</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Key</td>
<td>GNU Emacs Binding</td>
<td>Code</td>
<td>Pan Binding</td>
</tr>
<tr>
<td>---------</td>
<td>------------------------------</td>
<td>---------------</td>
<td>-------------------</td>
</tr>
<tr>
<td>C-x C-o</td>
<td>delete-blank-lines</td>
<td></td>
<td>Delete-Blank-Lines</td>
</tr>
<tr>
<td>C-x C-p</td>
<td>mark-page</td>
<td></td>
<td></td>
</tr>
<tr>
<td>C-x C-q</td>
<td>toggle-read-only</td>
<td></td>
<td>Toggle-Read-Only</td>
</tr>
<tr>
<td>C-x C-r</td>
<td>find-file-read-only</td>
<td></td>
<td></td>
</tr>
<tr>
<td>C-x C-s</td>
<td>save-buffer</td>
<td></td>
<td></td>
</tr>
<tr>
<td>C-x C-t</td>
<td>transpose-lines</td>
<td></td>
<td></td>
</tr>
<tr>
<td>C-x C-u</td>
<td>upcase-region [see Esc C-u]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>C-x C-v</td>
<td>find-alternate-file</td>
<td></td>
<td></td>
</tr>
<tr>
<td>C-x C-x</td>
<td>exchange-point-and-mark</td>
<td></td>
<td></td>
</tr>
<tr>
<td>C-x C-y</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C-x C-z</td>
<td>suspend-emacs</td>
<td></td>
<td></td>
</tr>
<tr>
<td>C-x ESC</td>
<td>repeat-complex-command</td>
<td></td>
<td></td>
</tr>
<tr>
<td>C-x $</td>
<td>set-selective-display</td>
<td></td>
<td></td>
</tr>
<tr>
<td>C-x (</td>
<td>star-kbd-macro</td>
<td></td>
<td></td>
</tr>
<tr>
<td>C-x )</td>
<td>end-kbd-macro</td>
<td></td>
<td></td>
</tr>
<tr>
<td>C-x +</td>
<td>add-global-abbrev</td>
<td></td>
<td></td>
</tr>
<tr>
<td>C-x -</td>
<td>inverse-add-global-abbrev</td>
<td></td>
<td></td>
</tr>
<tr>
<td>C-x .</td>
<td>set-fill-prefix</td>
<td></td>
<td></td>
</tr>
<tr>
<td>C-x /</td>
<td>point-to-register</td>
<td></td>
<td></td>
</tr>
<tr>
<td>C-x 0</td>
<td>delete-window</td>
<td></td>
<td>Close-Active-Viewer</td>
</tr>
<tr>
<td>C-x 1</td>
<td>delete-other-windows</td>
<td></td>
<td></td>
</tr>
<tr>
<td>C-x 2</td>
<td>split-window-vertically</td>
<td></td>
<td>Open-Another-Viewer</td>
</tr>
<tr>
<td>C-x 4</td>
<td>ctl-&lt;-&gt; prefix</td>
<td></td>
<td></td>
</tr>
<tr>
<td>C-x 5</td>
<td>split-window-horizontally</td>
<td></td>
<td></td>
</tr>
<tr>
<td>C-x ;</td>
<td>set-comment-column</td>
<td></td>
<td></td>
</tr>
<tr>
<td>C-x &lt;</td>
<td>scroll-left</td>
<td></td>
<td>Left-Hscroll</td>
</tr>
<tr>
<td>C-x =</td>
<td>what-cursor-position</td>
<td></td>
<td>Show-Status-Line</td>
</tr>
<tr>
<td>C-x &gt;</td>
<td>scroll-right</td>
<td></td>
<td>Right-Hscroll</td>
</tr>
<tr>
<td>C-x ?</td>
<td>!</td>
<td></td>
<td>Show-Kill</td>
</tr>
<tr>
<td>C-x [</td>
<td>backward-page</td>
<td></td>
<td></td>
</tr>
<tr>
<td>C-x ]</td>
<td>forward-page</td>
<td></td>
<td></td>
</tr>
<tr>
<td>C-x `</td>
<td>enlarge-window</td>
<td></td>
<td></td>
</tr>
<tr>
<td>C-x '</td>
<td>next-error</td>
<td></td>
<td></td>
</tr>
<tr>
<td>C-x a</td>
<td>append-to-buffer</td>
<td></td>
<td></td>
</tr>
<tr>
<td>C-x b</td>
<td>switch-to-buffer</td>
<td></td>
<td>Visit-Buffer</td>
</tr>
<tr>
<td>C-x d</td>
<td>dired [see C-x C-d]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>C-x e</td>
<td>call-last-kbd-macro</td>
<td></td>
<td></td>
</tr>
<tr>
<td>C-x f</td>
<td>set-fill-column</td>
<td></td>
<td>Set-Auto-Fill-Column</td>
</tr>
<tr>
<td>C-x g</td>
<td>insert-register</td>
<td></td>
<td></td>
</tr>
<tr>
<td>C-x h</td>
<td>mark-whole-buffer</td>
<td></td>
<td>Select-Buffer</td>
</tr>
<tr>
<td>C-x i</td>
<td>insert-file [see C-x C-i]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>C-x j</td>
<td>register-to-point</td>
<td></td>
<td></td>
</tr>
<tr>
<td>C-x k</td>
<td>kill-buffer</td>
<td></td>
<td>Remove-Buffer</td>
</tr>
</tbody>
</table>
### Pan I: An Introduction For Users

<table>
<thead>
<tr>
<th>Key</th>
<th>GNU Emacs Binding</th>
<th>Code</th>
<th>Pan Binding</th>
</tr>
</thead>
<tbody>
<tr>
<td>C-x l</td>
<td>count-lines-page</td>
<td>!</td>
<td>Goto-Line</td>
</tr>
<tr>
<td>C-x m</td>
<td>mail</td>
<td></td>
<td></td>
</tr>
<tr>
<td>C-x n</td>
<td>narrow-to-region</td>
<td></td>
<td></td>
</tr>
<tr>
<td>C-x o</td>
<td>other-window</td>
<td></td>
<td></td>
</tr>
<tr>
<td>C-x p</td>
<td>narrow-to-page</td>
<td></td>
<td></td>
</tr>
<tr>
<td>C-x q</td>
<td>kbd-macro-query</td>
<td></td>
<td></td>
</tr>
<tr>
<td>C-x r</td>
<td>copy-rectangle-to-register</td>
<td></td>
<td></td>
</tr>
<tr>
<td>C-x s</td>
<td>save-some-buffers [see C-x C-m]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>C-x t</td>
<td>advertised-undo</td>
<td>•</td>
<td>Undo</td>
</tr>
<tr>
<td>C-x w</td>
<td>widen</td>
<td></td>
<td></td>
</tr>
<tr>
<td>C-x x</td>
<td>copy-to-register</td>
<td></td>
<td></td>
</tr>
<tr>
<td>C-x y</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C-x {</td>
<td>shrink-window-horizontally</td>
<td></td>
<td></td>
</tr>
<tr>
<td>C-x }</td>
<td>enlarge-window-horizontally</td>
<td></td>
<td></td>
</tr>
<tr>
<td>C-x DEL</td>
<td>backward-kill-sentence</td>
<td>o</td>
<td></td>
</tr>
<tr>
<td>ESC C-@</td>
<td>mark-sexp</td>
<td></td>
<td>Select-Expr</td>
</tr>
<tr>
<td>ESC C-a</td>
<td>beginning-of-defun</td>
<td>•</td>
<td>[see lisp-mode]</td>
</tr>
<tr>
<td>ESC C-b</td>
<td>backward-sexp</td>
<td>•</td>
<td>Backward-Expr</td>
</tr>
<tr>
<td>ESC C-c</td>
<td>exit-recursive-edit</td>
<td>!</td>
<td>Capitalize-Selection</td>
</tr>
<tr>
<td>ESC C-d</td>
<td>down-list</td>
<td>!</td>
<td>Deselect-Region</td>
</tr>
<tr>
<td>ESC C-e</td>
<td>end-of-defun</td>
<td>•</td>
<td>[see lisp-mode]</td>
</tr>
<tr>
<td>ESC C-f</td>
<td>forward-sexp</td>
<td>•</td>
<td>Forward-Expr</td>
</tr>
<tr>
<td>ESC C-h</td>
<td>mark-defun</td>
<td>•</td>
<td>[see lisp-mode]</td>
</tr>
<tr>
<td>ESC TAB</td>
<td></td>
<td></td>
<td>Indent-Like-Previous-Line</td>
</tr>
<tr>
<td>ESC LFD</td>
<td>indent-new-comment-line</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ESC C-k</td>
<td>kill-sexp</td>
<td>•</td>
<td>Kill-Expr</td>
</tr>
<tr>
<td>ESC C-l</td>
<td></td>
<td>!</td>
<td>Lowercase-Selection</td>
</tr>
<tr>
<td>ESC RET</td>
<td></td>
<td>!</td>
<td>Move-Selection-To-Cursor</td>
</tr>
<tr>
<td>ESC C-n</td>
<td>forward-list</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ESC C-o</td>
<td>split-line</td>
<td>•</td>
<td>Split-Line</td>
</tr>
<tr>
<td>ESC C-p</td>
<td>backward-list</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ESC C-q</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ESC C-r</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ESC C-s</td>
<td>isearch-forward-regexp</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ESC C-t</td>
<td>transpose-sexps</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ESC C-u</td>
<td>backward-up-list</td>
<td>!</td>
<td>Uppercase-Selection</td>
</tr>
<tr>
<td>ESC C-v</td>
<td>scroll-other-window</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ESC C-w</td>
<td>append-next-kill</td>
<td>!</td>
<td>Select-Region-Dot-To-Mark</td>
</tr>
<tr>
<td>ESC C-y</td>
<td></td>
<td>!</td>
<td>Copy-Selection-To-Cursor</td>
</tr>
<tr>
<td>ESC ESC</td>
<td>eval-expression</td>
<td>•</td>
<td>Execute-Lisp-Line</td>
</tr>
<tr>
<td>ESC C\</td>
<td>indent-region</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ESC SPC</td>
<td>just-one-space</td>
<td>•</td>
<td>Just-One-Space</td>
</tr>
<tr>
<td>ESC !</td>
<td>shell-command</td>
<td>!</td>
<td>Cycle-Show-Clipboard</td>
</tr>
<tr>
<td>ESC $</td>
<td>spell-word</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ESC %</td>
<td>query-replace</td>
<td>!</td>
<td>Show-Match</td>
</tr>
</tbody>
</table>
The auxiliary table below displays Pan default bindings for keys (and key sequences) that do not exist in a strictly keyboard based editor like GNU Emacs.
Section 2.3.1 in the manual discusses *Pan* key binding more generally; section 4.2 explains how to customize key bindings.

### G.2 Menus

Commands in *Pan* may be bound to menus as conveniently as they may be bound to keys, both locally and globally. Menu bindings have the advantage of being visible to the user, but the disadvantage of taking longer to invoke.

Default menu bindings include some commands that are bound to both menus (for easy learning) and to keys (for convenience). Notable examples are the clipboard operations (see G.5, "Cut/Paste/Kill/Yank").

The base buffer and help buffer each have specialized menus. All other viewers provide a standard menu to which special purpose items may be appended in special editing modes (see G.8, "Special Editing Modes").

The *Pan* menu associated with a viewer appears in response to a press on the right mouse button when the cursor is over the viewer’s text viewing area.

Section 2.3.2 in the manual discusses *Pan* menu binding more generally; section 4.2 explains how to customize menu bindings.

### G.3 Undo

*Pan* has a general undo facility that behaves more like the one in vi than the one in GNU Emacs. The command Undo (bound by default to keys L4 and `X-u and to the main menu) reverses the most recent editing action. Undo is itself an editing action, so two consecutive invocations of Undo will result in no net change.

Certain classes of editing actions (notably insertions and deletions), when performed consecutively, are treated by *Pan*'s Undo as a single action that represents their aggregate effect.

<table>
<thead>
<tr>
<th>Key</th>
<th><em>Pan</em> Binding</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mouse_Left</td>
<td>Mouse-Select-@Level</td>
</tr>
<tr>
<td>Mouse_Middle</td>
<td>Mouse-Extend-@Level</td>
</tr>
<tr>
<td>Mouse_Right</td>
<td>Execute-From-Menu</td>
</tr>
<tr>
<td>Esc-Mouse_Left</td>
<td>Mouse-Select-Fullword</td>
</tr>
<tr>
<td>Esc-Mouse_Middle</td>
<td>Mouse-Extend-Selection-Fullword</td>
</tr>
<tr>
<td>L4</td>
<td>Undo</td>
</tr>
<tr>
<td>L6</td>
<td>Copy-To-Clipboard</td>
</tr>
<tr>
<td>L8</td>
<td>Paste-From-Clipboard</td>
</tr>
<tr>
<td>Esc-L8</td>
<td>Replace-From-Clipboard</td>
</tr>
<tr>
<td>L10</td>
<td>Cut-To-Clipboard</td>
</tr>
<tr>
<td>F1</td>
<td>Set-@Level-To-Character</td>
</tr>
<tr>
<td>F2</td>
<td>Up-@Level</td>
</tr>
<tr>
<td>F3</td>
<td>Down-@Level</td>
</tr>
</tbody>
</table>
Unlike GNU Emacs, cursor motion in Pan is considered an undoable editing action. Section 3.5 in the manual discusses Pan’s undo mechanism more generally.

G.4 Operand Level

The operand level mechanism in Pan has no counterpart in GNU Emacs or any other common editor. The operand level (or simply the level) is a persistent mode, local to each viewer. Its current value (usually one of “character,” “word,” or “line”) is visible on the viewer’s control panel and may be set either from the control panel or by three commands bound to keys F1, F2, and F3.

The level specifies how the operands of certain generic commands (those whose names contain @Level) will be determined. For example, the generic command Next-@Level moves the cursor forward by one character when the level is “character,” but when the level is “word” it moves the cursor forward by one word. Level-sensitive (generic) commands are bound by default so that when the level is “character,” they mimic GNU Emacs. Thus, Next-@Level is bound to `F, the slot where the Next-Character command would otherwise appear.

The motivation for this mechanism becomes clear in special editing modes where tree building occurs. Extra operand levels may be defined by specifications in the Ladle description of a programming language. For example, in asple-mode (asple is a demonstration programming language) possible levels are:

- Declaration
- Statement
- Expression
- Error
- Lexeme
- Line
- Word
- Character

The operand level mechanism is supported by a second level of mapping, analogous to keymaps. Thus, for example, the key `F is bound to Next-@Level in the keymap; Next-@Level is bound, in turn, to the command Next-Character when the level is “character.”Operand bindings are, of course, customizable too.

Section 2.3.3 in the manual discusses Pan’s operand binding mechanism more generally.

G.5 Cut/Paste/Kill/Yank

The functional area of Pan most likely to confuse experienced GNU Emacs users is the management of text that is to be deleted and/or moved among buffers. The two editors support models that are superficially alike but differ in crucial ways. The potential for confusion is exacerbated by the inherently invisible nature of some manipulations in this category.
This section presents the basic model supported by each editor. It enumerates the basic abstractions (drawing the pivotal distinction between buffer-local and editor-global) and gives examples of commands that operate upon them. Section 3 of the manual presents a more thorough introduction to Pan’s editing model.

G.5.1 The GNU Emacs Model

Window Abstractions

- A GNU Emacs buffer may have one or more windows that provide an independently scrollable view of the buffer.
- Exactly one GNU Emacs window is active at any time.
- The active window has a visible dot (a.k.a. cursor), at a specific point in the buffer; the dot is constrained to be always visible in the active window.
- A visible, inactive window does not display its dot, but retains its location should it become active again. Invisible windows do not exist, and therefore retain no state.

Buffer Abstractions

- Each buffer may have a invisible mark, independent of any dots, at a specific point in the buffer.
- If a buffer has a mark currently set, the text interval between dot and mark in the active window implicitly (and invisibly) defines the region.

Global Abstractions

- The editor contains a single, invisible kill ring onto which text from various buffers may be pushed.

Operations

- Killing any text (^K, ^W, etc.) pushes it onto the global kill ring.
- The command copy-region-as-kill (Esc-w) pushes the region onto the global kill ring without killing.
- The command yank (^Y) inserts the most recently pushed text (from any buffer) at the dot in a specific buffer.
- The command yank-pop (Esc-y) removes the result of an immediately preceding yank operation, pops the most recently pushed text off of the kill ring, and replaces the removed text with the new top of the kill ring.

There are, of course, more operations for manipulating the kill ring explicitly, but this description is sufficient for comparison. GNU Emacs also supports a number of named global registers into which text can be stored. Pan supports nothing similar, so they will not be discussed here.
G.5.2 The Pan Model

In Pan it is necessary to distinguish between buffers and viewers. A single file is always edited in a single buffer. A Pan buffer may have one or more viewers visible on the screen, or it may have none at all. Viewers are something like GNU Emacs windows, since many of them may be attached to a buffer, but they retain more state, more persistently than GNU Emacs windows.

Viewer Abstractions

- Each viewer associated with a buffer has a visible dot (a.k.a. cursor) at a specific point in the file. Like GNU Emacs, there as many dots associated with a buffer as there are viewers. Unlike GNU Emacs, (a) the dot is not constrained to be in the visible part of a viewer, (b) the dot is visible even when a viewer is not active (doesn't have the keyboard focus in SunView terminology), and (c) the location of the dot (along with size, scroll position, and screen location) persists while a viewer is invisible.

Buffer Abstractions

- Like GNU Emacs, each buffer may have an invisible mark, independent of viewer dots, at a specific point in the file.
- Somewhat like GNU Emacs, the text interval between a dot (in a viewer) and mark (in a buffer, if set) implicitly (and invisibly) defines the region. Unlike GNU Emacs, this region is of little interest to the Pan user.
- Each buffer may have a visible selection, independent of the mark and any dots. The selection appears highlighted wherever visible in any active viewer. The selection appears underlined, but a user option (global or buffer local) may be changed to request inverse highlighting instead.
- Each buffer has a kill ring that is, unlike GNU Emacs, local. The mechanisms by which successive kills are coalesced are less well developed than in GNU Emacs.

Global Abstractions

- The editor contains a single clipboard, also a ring, onto which text from various buffers may be pushed.

Operations

This section lists examples of Pan commands related to these abstractions. Each command mentioned will be followed by the key to which it is bound by default, if any.

- Like GNU Emacs, deleting text removes it from a buffer, and it is not retained (except on the undo stack):
Delete-Character ("D)
Delete-Previous-Character (DEL)
Delete-Word
Delete-Region-Dot-To-Mark
Delete-Selected-Region
Delete-Blank-Lines ("X-O)
Delete-Horizontal-Space (Esc-)

etc.

- Killing any text pushes it onto the buffer's local kill ring:
  
  Kill-Word (Esc-d)
  Kill-To-EOL ("K)
  Kill-Selected-Region ("W)
  Kill-Region-Dot-To-Mark
  etc.

- Like GNU Emacs, text can be pushed onto the buffer's local kill ring without removing it:
  
  Copy-Selection-As-Kill (Esc-w)

- Like GNU Emacs, text may be retrieved from the top of a buffer's local kill ring by yanking, causing it to be inserted at the dot; Cycle-Yank specifies that the kill ring be cycled before the yank and insertion. However, unlike GNU Emacs, both yank operations may be modified by a prefix argument, in which case yanked text replaces the current selection instead of being inserted at the dot. Thus, Pan's Yank-From-Kill-Ring with no prefix arguments (key sequence "Y") is similar to the GNU Emacs "yank" command, and Pan's Cycle-Yank with prefix arguments (key sequence "U Esc-y") has an effect similar to the GNU Emacs "yank-pop" command.

  Yank-From-Kill-Ring ("Y")
  Cycle-Yank (Esc-y)

- A buffer's local kill ring may be cycled, bringing successive entries to the top, and the text currently at the top may be viewed in the Help buffer.

  Cycle-Y::!
  Show-Kill ("X-?)
  Cycle-Show-Kill ("X-!)

- Some operations use the buffer's current selection as an operand (see also clipboard operations below):

  Append-Selection-To-File
  Apropos-Selection
  Write-Selection-To-File
  Kill-Selected-Region ("W)
  Lowercase-Selection ("X-'U)
  etc.
Pan I: An Introduction For Users

- One can create a selection either with the mouse or from the dot and mark; the latter option is included only for compatibility with emacs.

  Select-Region-Dot-To-Mouse (Mouse-Right)
  Select-Region-Dot-To-Mark (Esc-^W)
  Deselect-Region (Esc-^D)

- Some commands operate on both the selection and the dot. GNU Emacs has no equivalents because it supports no abstraction corresponding to a selection that is independent of the dot.

  Copy-Selection-To-Cursor (Esc-^Y)
  Move-Selection-To-Cursor (Esc-^M)

- Text may be pushed onto the clipboard from the current selection (Copy or Cut). Text may be retrieved from the top of the clipboard, and it may be either inserted at the dot (Paste) or used to replace the current selection (Replace).

  Copy-To-Clipboard (L6)
  Cut-To-Clipboard (L10)
  Paste-From-Clipboard (L8)
  Replace-From-Clipboard (Esc-L8)

- The clipboard may be cycled, bringing successive entries to the top, and the text currently at the top may be viewed in the Help buffer.

  Cycle-Clipboard
  Show-Clipboard (Esc-?)
  Cycle-Show-Clipboard (Esc-!)

G.6 Options

Many aspects of Pan’s operation may be controlled by the settings of options, somewhat analogous to GNU Emacs variables (although Pan also has variables). As with bindings, options may have both global and buffer-local values. The help command List-All-Options lists the option values currently in effect for the active buffer. Some user-level commands are available for setting options dynamically, for example Set-Auto-Fill-Column and Toggle-Read-Only. Section 4.3 of the manual explains in more detail how to manage Pan options.

G.7 Help

Pan has a help mechanism that is somewhat different from the one in GNU Emacs. All output from help commands appears in the distinguished buffer named "Help Info." Output from some commands (for example the command Describe that provides additional information about a selected Pan symbol) is inserted among existing text in the help buffer. The command Reset clears the contents of the help buffer; some help commands do this automatically before responding.
Section 3.6 in the manual presents a more detailed explanation of Pan's help mechanism.

G.8 Special Editing Modes

Both editors support specialized editing modes that are optimized for editing certain classes of files, C programs or TeX documents for example. A mode is typically invoked by pattern matching against file extensions.

In its simplest form, a mode contains alternate key bindings and special purpose commands.

Pan modes can operate like GNU Emacs modes, but have additional flexibility. For example, Pan modes may also add special purpose submenus. Language editing modes load Ladle tables for parsing and tree building and provide extra commands for tree navigation and manipulation.

The short tables below describe two textual modes supported by GNU Emacs, along with such textual Pan equivalents as currently exist. These modes are experimental and incomplete. Full structure- and semantic-based Pan support for these languages is under development. For a discussion of Pan's true language editing modes, see section 3.12 of the manual.

Lisp Mode Bindings

Pan contains a small set of text-oriented functions for operating on LISP expressions. More powerful functions await true language-based editing on LISP.

<table>
<thead>
<tr>
<th>Key</th>
<th>GNU Emacs Binding</th>
<th>Code Pan Binding</th>
</tr>
</thead>
<tbody>
<tr>
<td>TAB</td>
<td>lisp-indent-line</td>
<td></td>
</tr>
<tr>
<td>DEL</td>
<td>backward-delete-char-untabify</td>
<td></td>
</tr>
<tr>
<td>ESC</td>
<td>Prefix Command</td>
<td></td>
</tr>
<tr>
<td>ESC C-a</td>
<td>beginning-of-defun [std.]</td>
<td>Previous-Function</td>
</tr>
<tr>
<td>ESC C-e</td>
<td>end-of-defun [std.]</td>
<td>End-Of-Function</td>
</tr>
<tr>
<td>ESC C-h</td>
<td>mark-defun [std.]</td>
<td>Select-Function</td>
</tr>
<tr>
<td>ESC C-q</td>
<td>indent-sexp</td>
<td></td>
</tr>
<tr>
<td>ESC C-x</td>
<td>lisp-send-defun</td>
<td></td>
</tr>
<tr>
<td>ESC [</td>
<td>backward-paragraph [std.]</td>
<td>Previous-Function</td>
</tr>
<tr>
<td>ESC ]</td>
<td>forward-paragraph [std.]</td>
<td>Next-Function</td>
</tr>
</tbody>
</table>

C Mode Bindings

There is no special mode yet for C in Pan, either text-oriented or language-based.
<table>
<thead>
<tr>
<th>Key</th>
<th>GNU Emacs Binding</th>
<th>Code Pan Binding</th>
</tr>
</thead>
<tbody>
<tr>
<td>TAB</td>
<td>c-indent-command</td>
<td></td>
</tr>
<tr>
<td>DEL</td>
<td>backward-delete-char-untabify</td>
<td></td>
</tr>
<tr>
<td>ESC</td>
<td>Prefix Command</td>
<td></td>
</tr>
<tr>
<td></td>
<td>:</td>
<td>electric-c-terminator</td>
</tr>
<tr>
<td></td>
<td>;</td>
<td>electric-c-semi</td>
</tr>
<tr>
<td></td>
<td>}</td>
<td>electric-c-brace</td>
</tr>
<tr>
<td></td>
<td>{</td>
<td>electric-c-brace</td>
</tr>
<tr>
<td>ESC C-q</td>
<td>indent-c-exp</td>
<td></td>
</tr>
<tr>
<td>ESC C-h</td>
<td>mark-c-function</td>
<td></td>
</tr>
</tbody>
</table>