Pan I Version 4.0
An Introduction For Users

Laura M. Downs
Michael L. Van De Vanter

Computer Science Division (EECS)
University of California, Berkeley
Berkeley, California 94720
August 1991

Report No. UCB/CSD 91/659

Research sponsored in part by the Defense Advanced Research Projects Agency (DoD), monitored by Space and Naval Warfare Systems Command under Contracts N00039-84-C-0089 and N00039-88-C-0292, by IBM under IBM Research Contract No. 564516, by a gift from Apple Computer, Inc., and by the State of California MICRO Fellowship Program. Michael L. Van De Vanter was supported in part by a MICRO fellowship.
Abstract

Pan is a prototype and testbed for language-based editors and views. Its design addresses the needs of experienced users who manage complex objects such as large software systems. All of Pan's components are multilingual, incremental, description-driven, customizable, and extensible. Viewing is facilitated by semantics-based browsing and an object model that 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 user's manual, describes the basic operational facilities of Pan 1 (version 4.0), 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), view options, and view flags.
Contents

1 Introduction .......................... 1

2 Concepts .................................. 1
2.1 Views .................................. 1
    Special Views .......................... 2
    Views vs. Windows .......................... 2
2.2 Windows .................................. 2
    The Information Panel ............ 4
    The Editing Area ............... 4
2.3 View Styles .......................... 4
2.4 Textual Views ......................... 4
    The Text Cursor ....................... 5
    Regions and The Text Selection .......... 5
    Rings .................................. 5
    The Clipboard .......................... 6
    Kill Rings ............................... 6
    Marks and the Mark Stack .......... 6
2.5 Graphical Views ....................... 6
2.6 Configuration .......................... 7
    Scopes .................................. 7
    Key Bindings ............................. 7
    Menu Bindings ............................ 8
    Operand Level Bindings and Menus .......... 8
    Character Sets .......................... 10
    Options, Flags, and Variables ........ 10
    Font Maps ................................ 11
2.7 Language Description .................. 11

3 Basic Interaction .......................... 12
3.1 Getting Started .......................... 13
3.2 Executing Commands .................... 13
3.3 Quitting .................................. 13
3.4 Supplying Arguments to Commands .... 13
    Numeric Prefix Arguments .................. 13
    Prompts and Pop-Ups .................. 14
3.5 Cancelling Commands .................... 14
3.6 Reviewing Message History ........... 14
3.7 Undoing Actions .......................... 15
3.8 Getting Help ................................ 15
    Apropos ................................. 16
    Help Commands ............................ 16
    Documentation Commands .......... 17
3.9 Views and Windows .................................................. 17
Visiting Files and Other Objects ................................. 17
Directory Editor ...................................................... 18
Saving and Writing Files ........................................... 19
Manipulating Windows ............................................. 20
3.10 Scrolling .......................................................... 20
3.11 When Things Go Wrong ........................................ 21

4 Using Textual Views .................................................. 22
4.1 Text Cursor Motion ............................................... 22
  Moving the Cursor With the Mouse ............................ 22
  Moving Forward and Backward .................................. 22
  Moving Around in the Text Stream ............................. 23
  Mark Commands .................................................. 24
4.2 The Current Operand Level ..................................... 24
4.3 Operand Level Commands ....................................... 25
4.4 The Textual Selection ............................................. 26
4.5 Entering Text ...................................................... 27
4.6 Filling Text ........................................................ 28
4.7 Deleting Text ...................................................... 28
4.8 Killing Text ........................................................ 29
4.9 The Clipboard ...................................................... 30
4.10 Copying and Moving Text ....................................... 30
4.11 Changing Case .................................................... 31
4.12 Transposing Text ................................................ 31
4.13 Searching Text .................................................... 31
  Regular Expressions .............................................. 31
  Balanced Bracket Commands ................................... 33

5 Using Graphical Views ................................................. 33

6 Language-Oriented Services ....................................... 34
6.1 Language-Based Textual Views ................................. 35
6.2 Language-Based Analysis ....................................... 36
6.3 Consistency and Inconsistency ................................ 37
6.4 Operand Levels .................................................... 38
6.5 The Structure Cursor ............................................. 38
6.6 Setting The Structure Cursor .................................. 39
6.7 Structural Navigation ............................................. 40
6.8 Language Errors ................................................... 41
6.9 Editing ............................................................. 42
6.10 Searching in Programs .......................................... 43
6.11 Queries ............................................................ 43
6.12 Debugging Features ............................................. 43
Graphical Views .................................................. 44
Other Language-Oriented Views ............................. 44
Exploring the Tree Structure ................................. 45

7 Simple Customization ................................. 45
  7.1 Start-Up Processing ................................... 45
  Language-Based Configuration .......................... 46
  7.2 Using Scopes ............................................. 47
  7.3 Packaging ................................................ 47
  7.4 Creating Bindings ..................................... 48
  Creating Key Bindings .................................. 48
  Creating Menu Bindings .................................. 50
  Creating Operand Level Bindings ....................... 51
  7.5 Getting and Setting Options ......................... 52
  Setting Font Maps ....................................... 53
  Setting Color Maps ....................................... 53
  7.6 Character Sets .......................................... 54
  7.7 Creating and Modifying Flags ....................... 55
  7.8 Lisp-Oriented Commands .............................. 56
      Defining Variables and Options ....................... 57
      Hooks and Notifiers .................................. 57

8 Acknowledgments ........................................... 58

References .................................................. 59

9 Glossary ................................................... 60

A Default Key Bindings .................................... 67
  A.1 Bindings by Command Name in a Text View .......... 67
  A.2 Bindings by Key in a Text View ...................... 69
  A.3 Special Key Bindings in the View List ............. 71
  A.4 Special Key Bindings in the Help View ............. 71
  A.5 Special Key Bindings in a Directory Editor .......... 71
  A.6 Special Key Bindings in a Language-Based View (SIMPLE2) 71
  A.7 Special Key Bindings in a Graphical View .......... 71

B Default Menu Bindings .................................... 72
  B.1 Bindings in an Ordinary Text View by Command Name 72
  B.2 Bindings in an Ordinary Text View by Menu ........ 73
  B.3 Additional Menu Bindings in the View List .......... 74
  B.4 Additional Menu Bindings in the Help View .......... 74
  B.5 Additional Menu Bindings in a Directory Editor .... 74
  B.6 Additional Menu Bindings in a Language-Based View (SIMPLE2) 74
  B.7 Additional Menu Bindings in a Graphical View .... 74
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>C Operand Level Bindings in a Language-Based View (SIMPLE2)</td>
<td>75</td>
</tr>
<tr>
<td>D Options Defined in Pan</td>
<td>77</td>
</tr>
<tr>
<td>E Flags Defined in Pan</td>
<td>89</td>
</tr>
<tr>
<td>F Character Sets in a Text View</td>
<td>90</td>
</tr>
<tr>
<td>G View-Styles</td>
<td>91</td>
</tr>
</tbody>
</table>
List of Figures

1  Screen image of Pan showing view list, help view, and a text (TEX) view 3
2  Pan with a default menu visible 9
3  Two language-based views on a program 35
4  The operand levels for SIMPLE2 39
5  The options that control structural highlighting 42
6  Bindings and names for special keys 49
1 Introduction

*Pan* is an editing system for text- and language-structured documents that uses the mouse, menus, and multiple windows to provide both textual and language-based browsing and editing. *Pan* is implemented in *Common LISP* with PCL and C. It runs on Sun3 and Sparc Workstations\(^1\) under SUNOS and X11R4.

This document is an informal introduction to *Pan* I Version 4.0. It contains essential information for editing with *Pan*. All of the commands necessary for editing files, programs, and directories are described along with the key-sequences or menu choices which make them quickly accessible. With the help of this manual and the on-line help system, a new user should be able to edit text files with *Pan*. If you have not used *Emacs* or a similar editor before this, you can get started on *Pan* by focusing on the cursor motion commands (see Section 4.1) and using *Pan*’s menus to visit files, open views, use the clipboard, and get help.

The text-oriented facilities of *Pan* are modeled partially on the *Emacs* family of text editors, so users familiar with an *Emacs*-style editor will have little trouble learning *Pan*. *Pan* is extensible and customizable in the spirit of *Emacs*\(^7\).

Some information for extending the system is included in this document. Using the information contained in the manual, any user should be able to set up a “.panrc” file which alters key- and menu-bindings and changes the default value for any options to customize the editing environment (see Section 7). For more help, you may want to contact the *Pan* group (panpipes@sequoia.berkeley.edu) directly. For more general background on *Pan*, consult a recent description of the project [2].

*Pan* also provides for manipulating and editing programs using the syntax and semantics of the language being edited. *Pan* understands descriptions of language syntax using the language-definition language *Ladle*[3]. The semantics of the language may also be described using the semantics-description language *Colander*[1]. To effectively use the language-based facilities within *Pan*, all the user should need to know is the language in question. The most useful of *Pan*’s language facilities, operand-sensitive commands, is described in this manual and in *Pan*’s help system. For more information and background on *Pan*’s language-based features, see *Coherent User Interfaces for Language-Based Editing Systems*[8].

Adding new language definitions to *Pan* is beyond the scope of this manual.

2 Concepts

This section is a brief introduction to the terminology and notation of *Pan*.

2.1 Views

A view represents a format for presenting the contents of or information about a file, directory, program, or other editable object. A view may also provide services with which a user may browse, explore, and modify the object being presented. For example, a text view and a graphical view of a program would be considered different views of the same program.

\(^{1}\)Sun Workstation is a registered trademark of Sun Microsystems, Inc.
Each view has a configurable environment which includes specifications for many aspects of the view's behavior, such as:

- key, menu, and operand-level bindings.
- definitions of character sets.
- option values, and
- flag specifications.

All views in *Pan* are named. The name of a view is composed of the name of the object being edited in that view, the type of the view, and possibly some extra qualifying information. In the current release, the name of a text view is the name of the file being edited.

**Special Views**

The view list is your doorway to *Pan*: within its edit window are the names of the views being edited. As new views are created, their names are added to this list. The view list appears in the upper left of Figure 1 on page 3, a typical workstation screen running *Pan*; it lists two other views, the help view and a view named "manual.tex".

The help view is a special text view which is used for displaying help and other information. A window on the help view appears in the upper right of Figure 1 and currently shows the key bindings in effect for the active view. There can be only one help view at any given time (although this view can have several windows). Most commands which output information to the help view clear its contents first. See Section 3.8 for more about help facilities.

**Views vs. Windows**

Most commands act on the active window of the active view. The active window is the window in which the most recent keystroke or mouse action occurred. The active view is the view which owns the active window.

Views exist independently from the windows (see below) in which they are displayed. Thus there can be views that are not visible on the workstation.

In Figure 1, the text view "manual.tex" (containing a file of the same name) is currently active and has two separate windows open onto it.

**2.2 Windows**

A window represents the actual X window in which the contents of the view are displayed. A view may have any number of independent windows. For example, the text view "manual.tex" in Figure 1 has two separate windows open onto it.

A window provides a display mechanism, scroll bars, a message line, view flags, and a text cursor as seen in Figure 1. Every window also has its own operand level (see Section 2.6). All of the windows opened onto a single view share that view’s contents: its bindings, its option values, and its flag specifications. A textual view (see Section 2.4) may at times have a single text selection,
Figure 1: Screen image of *Pan* showing view list, help view, and a text (**TEX**) view

shared by all of the view's windows, but each text window has its own independent text cursor and any window may show a different part of the view.

The horizontal scroll bar and horizontal scrolling commands allow you to see text to the right (or left) of the current edit window (see Section 3.10). If the Text-Fill option is on then the flag "\(" appears in the information panel and newly typed lines are caused to wrap around rather than being truncated (see Section 4.6).

*Pan* windows can be manipulated like any X window according to the window manager you are using. Closing a window from the window manager causes that window to be iconified. Closing a window from within *Pan* will cause the window to be iconified if the option Window-Close-With-Icon is set to be true, otherwise no icon is created. In any case the window can be reopened from the view list or by opening a view or window on that file. Internal state of the window is retained even when the window is not visible (see Section 3.9).

A window is partitioned into two areas: the information panel and the editing area.
The Information Panel

At the top of the window appears the information panel, which displays information about the view. It includes the view name, a message line, and the values of various flags. For example, Figure 1 on page 3 shows the Object-Modified? and Text-Fill flags in the “manual.txt” view and the Object-Protected? flag in the help view.

It also may display the language being edited and the operand selection level (see Section 6). Clicking the right mouse button over the name of the current operand level will bring up a menu of all of the possible operand levels for the view.

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 activates the Pan menus for that view. Selecting an item from those menus executes the command bound to that selection (see Section 3.2).

In a text window, pressing the left button of the mouse sets the text cursor to the position of the mouse (see Section 4.1). Pressing the middle button of the mouse selects the region between the text cursor and the mouse (see Section 4.4).

Windows onto textual views 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.

In a graphical window, there is no text cursor but the behavior of the panel, the menu selection mechanism, and the scroll bars is the same (see Section 5).

2.3 View Styles

Within Pan there are many different editing contexts. These different contexts are called view styles. Each view must be associated with a view style. For example, the help view uses the help-view-style, directory editors use the direc-view-style, and views on ordinary text files use the text-edit-view-style. Every Pan view style is either graphical or textual. For a list of all of the available view styles, see Appendix G.

All views of a particular view style inherit its configuration information (see Section 2.6). The menus, key-bindings, flags, and fonts as well as many other less visible aspects of Pan’s behavior provide a look and feel for each view style which may be individually tailored to provide a comfortable environment for different editing purposes.

2.4 Textual Views

Most views in Pan are based on the textual display and manipulation of information. Many, but not all, of the services in textual views reflect familiar styles of text-based editing. This section presents a brief overview of those services. Section 4 contains more detail.
2.4 Textual Views

The Text Cursor

Each text window has a single text cursor independent of any other cursors in other windows on the view. A text cursor appears on the screen as an inverse-video or outlined box highlighting the character selected by that cursor.

All textual 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 cursor (for instance, by deleting the current selection), the cursor is moved to the point of change. Operations that modify text scroll the active window so that the change (and therefore the cursor) is visible.

A view's text cursor may not be visible on the screen due to scrolling or other motion. The command Scroll-To-Cursor-Current-Window in the Window menu scrolls the window so that the cursor is visible as does any cursor movement.

Regions and The Text Selection

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

Every textual view can have a specially designated region of text called the selected region or simply the selection. When the selection is set, it is underlined in all of the windows in which it is visible. Figure 1 on page 3 shows a selection shared by two windows. Commands that alter the contents of the view deselect the current selection.

Pan also has an implicit selection, namely the region between the text cursor and the top mark of the mark stack. This selection is different for each window, since each window has a different 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. The command Select-Region-Dot-To-Mark which is usually bound to the key sequence “Esc “W” makes the implicit selection into the selection.

Pan's style of selection is similar to, but not the same as, X's. Clicking the middle button of the mouse selects and underlines 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 text cursor are all independent.

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.
The Clipboard

The clipboard—a holder for regions of text—is shared among all views. A selection can be copied (or cut) to the clipboard and then pasted in another view. These operations are modeled on the Macintosh\textsuperscript{2} 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.

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 by yanking it.

The size of the kill ring is determined by the value of the option Killring-Max-Size.

In Pan, the kill ring is local to a view rather than global to all views as it is in Emacs. The kill ring and the clipboard are independent entities; cutting text to the clipboard does not affect the contents of the kill ring and killing text does not affect the clipboard. However, if the option Kills-To-Clipboard is set, then the "Kill-" commands use the global clipboard instead of the kill ring and text killing and yanking is then effectively global.

Marks and the Mark Stack

A mark is a character position in a textual view. 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 view 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".

2.5 Graphical Views

For any program or other structure-based view, a graphical view of the internal tree representation may be created. Graphical views may be generated on the entire internal tree or on any subtree of it. The contents of the nodes in the graph may contain the names of the nodes or other information about the node, depending on the command used to generate the view.

The contents of the text stream from which the view was generated can not be altered from within the graphical view. In the current version of Pan, the graph is a display only and not a graphical editor; not even the structural cursor may be set from within a graphical view.

Graphical views may be navigated by thumbing the window to a fixed position by pressing the middle mouse button over the scroll bars or by "Zoom-"ing in and out.

A graphical view also contains a global image in the upper-left corner of the window. Within this image a miniature of the entire graph or subgraph being viewed is displayed, and the region of the graph which is currently visible in the window is outlined. This global image may be resized or toggled on and off.

Graphical views of the tree structure of programs are the only kind of graphical view implemented at present. Because the graphical view functions only as a display of the tree representation

\textsuperscript{2}Macintosh is a registered trademark of Apple Computer, Inc.
and has no interactive capability. Its usefulness is limited mostly to system extenders, particularly
to authors of language descriptions.

Other types of graphical views are possible, such as a program call graph, but have not yet
been implemented. This sort of view would have more practical use to the programmer.

2.6 Configuration

Most of the user interface and much of the operation of Pan may be customized. Each view in Pan
is a separate editing context in which many aspects of Pan's behavior and user interface may be
configured. The default configuration of Pan is defined by a collection of declarations in the same
format as those a user might write to customize one or more of Pan's editing contexts (Section 7).
A binding associates a sequence of keyboard or mouse actions with a command. Pan provides
for adding key- and menu-bindings, setting options, configuring operand-level bindings, describing
character sets, creating and manipulating flags, and creating new editing environments for specific
uses. The on-line help system provides documentation for any configurable Pan command, option,
or other object. All of these are scoped to provide flexible control of the editing environment.

Scopes

There are normally three possible scopes\(^3\) for any binding or option:

- :global - global to all views,
- :view-style - restricted to all views of a particular view style, and
- :view - local to a single view.

Any configurable data can lie within any of these scopes. Therefore they are either local to a
view, shared by all views of a particular view style, or global to all views. Thus, every view may
have its own set of key bindings, and even its own menus and menu selections. Naturally, local
bindings take precedence over global bindings.

Key Bindings

Key bindings associate keystroke sequences with commands. They provide the user with a quick
way to execute commands. Most commonly used commands are bound to some key sequence. Pan,
like Emacs, provides a live keyboard. Keystrokes (including mouse buttons and special keyboard
keys) are read until a valid binding is detected. When a binding is detected, the associated command
is executed.

A key binding associates a keystroke sequence with a command. These key sequences are
generally one or two characters but there is no restriction on how long a binding can be. The
keys "Escape", "X", "C", and "Z" are, by convention, used as prefix keys. The "Shift" and
"Control" keys are modifier keys rather than prefix keys. Any keystroke sequence may be bound
to any command. In your configuration, you can set aside other keys to be prefixes as well as those

\(^{3}\) The scope :buffer, which consists of all views on a single editable object, is also defined internally but is not
often used for configuration.
mentioned above. However, if you bind a key sequence to a command, that binding will shadow any previously defined bindings which have that sequence as a prefix or which are a prefix of that sequence. For example, if you bind "-A "X" to a command, you will no longer have the usual binding for "-A" or if you bind "-X" to a command, you will no longer have any of the key bindings for which "-X" is a prefix.

Any key sequence can be bound in any scope. A binding in a :view-style scope will shadow a binding to the same sequence in the :global scope; a binding in the :view scope will shadow a binding in its :view-style scope or the :global scope. Thus each view-style or view can have its own customized editing environment.

Mouse buttons and function keys can be bound to commands just like the standard keyboard keys. In fact, it is the standard binding of the right-most mouse button to Execute-Menu that implements Pan’s menu selection service. The function keys available vary between keyboards and key mappings. Pan’s default key bindings for the special keyboard keys generally reflect the standard usage for the keys. For example the “Undo”, “Cut”, and “Paste” keys on the left and the “Up-” and “Down-Arrow” keys on the right are bound to commands which reflect their purpose.

The default bindings for mouse buttons are similar to the X bindings. Appendix A lists Pan’s default key bindings.

Menu Bindings

Each view has a default menu, often inherited from the view’s view style, but bindable in any scope. A Pan menu contains a collection of menu bindings, each of which associate menu item selections with commands. A menu binding consists of a menu, a menu item, and a command name.

The advantages of a menu binding are that you don’t need to remember a command name or a key binding to execute the command, because you need only recognize the command in the menu and select the corresponding menu item with the mouse. Because of this, menu bindings are extremely useful for the new user. Like key bindings, menus and menu bindings may be scoped by :view, :view-style, or :global. The default menus associated with a view appear when the right button of the mouse is pressed while the mouse is positioned over the edit window. Figure 2 on page 9 shows a menu selection being made.

A menu binding is designated by a menu title and a selection name, denoted typographically by MenuTitle: SelectionName. For example, the selection Menus in Figure 2 is bound to the command Display-Default-Menu. 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.1–B.4 list the default menu bindings for a text view, for the view list, and for the help view.

As with key bindings, any command may be bound to a menu item. Each different menu or submenu is globally defined. The binding of a submenu or a menu selection to a menu affects the global definition of the menu. However, menus may be copied and the copy may be altered to achieve a :view-style or :view local configuration.

Operand Level Bindings and Menus

Operand level bindings associate generic operations, such as Cursor-Forward or Delete with
2.6 Configuration

Figure 2: Pan with a default menu visible

operands designated by the current operand level (see Section 6). 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 window to determine the actual command to execute.

Each window has a **current operand level** which can be used to control the actions of operand-generic operations. The generic operations are: Cursor-Forward, Cursor-Backward, Cursor-To-First, Cursor-To-Last, Cursor-In, Cursor-Out, Select, Cursor-To-Mouse, Mouse-Extend, and Delete as well as the search operations: Query-Replace-All, Replace-All, Cursor-Search-Forward, and Cursor-Search-Backward. The Oplevel-Menu command, which is normally bound to "C Mouse-Right" brings up a menu which contains all of the operand-level operations for the current level.

For example, the key "Right-Arrow" is bound to the command Oplevel-Cursor-Forward. If the current operand level is "Character", and the binding of the generic command Cursor-Forward at the level "Character" is the command Next-Character, then Oplevel-Cursor-Forward will execute Next-Character.
Setting the operand level is a bit like changing modes in a moded editor, except that it only affects the generic bindings. The current operand level persists across operations. In pure text editing, the operand level is of limited usefulness. When editing views that have a richer operand domain, for example programs supported by one of Pan's language descriptions, 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 4 on page 39 shows the operand-level choices for a language-based view.

The current operand level affects only those commands that consult it. These commands are syntactically distinguished by containing the phrase "Oplevel" in their name. An operand level binding is designated by a generic operation and an operand level denoted typographically by Operation at level "OperandLevel".

As with key-bindings, operand level bindings may also be scoped by :view :view-style or :global to provide a specialized editing environment for different views.

Character Sets

A character set has a name and may contain any number of ASCII characters. For example, the set indentation-characters contains the characters \{ Tab Space \}. The default character sets are listed in Appendix F. Any character may belong to any number of sets or to no set. Character sets are used to recognize white-space, to identify matching brackets, and to find the extent of a word among other things. Commands for finding and altering the contents of a character set are described in Section 7.6.

Character set definitions are scoped and local definitions shadow global definitions. For example, the set word-characters is defined differently in the help view than in text views. The command Display-Char-Sets will display the contents of the character sets in the active view.

Options, Flags, and Variables

An option is implemented as a variable which is used to control user-configurable settings. Option variables, flags, and variables are provided for controlling and extending the system.

Option variables, or simply "options" are used to control user-configurable settings. They are implemented as strongly-typed, scoped variables. Appendix D lists the basic options, their types, and their default values. The effective value of an option is obtained using the command Announce-Option-Variable (see section 7.5). All options along with their values for a particular view may be listed with the command Display-Option-Variables.

A flag is an option that can be made visible. Flags can be displayed on the information panel by adding the associated option to a list of flag specifications along with an "on-icon" and an "off-icon". When the flag is set, i.e. the associated option is non-nil, then the "on-icon" is displayed in the information panel. The "off-icon" is often invisible. For instance, if the Object-Modified? flag is set (meaning that the text of a view has been modified since last saved), a "*" appears on the information panel. Appendix E lists the default set of flags, together with their display properties.

A Pan variable is scoped, like an option, but is used for internals and not accessible to the user.
2.7 Language Description

Font Maps

Every character in Pan's internal text representation contains a font code. This code determines
the font used when the character is displayed in a window. Each view has an associated font map,
which associates font codes with internal font descriptors. A font map contains from 1 to 8 entries.

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

Pan maintains a default font map option which may be scoped. You can alter these defaults by
setting the option Text-Window-Fontmap globally, in the scope of the view-style you want to affect,
or in a particular view.

The standard specification for a font map is a zero-indexed list of font names containing from
1 to 8 names, for example:

("-adobe-times-medium-i-normal--17-120-100-100-p-84-iso8859-1"
"-b&h-lucida-medium-r-normal-sans-12-120-75-75-p-71-iso8859-1"
"-adobe-times-bold-i-normal--17-120-100-100-p-86-iso8859-1")

Standard text views use only the default font. Section 6.2 describes a more elaborate use of
font maps in conjunction with language-structured documents.

2.7 Language Description

Adding new language descriptions is the most important of Pan’s extension mechanisms. A
Pan language description is distinct from a language definition, which is written by the designer
of the language, although it typically relies heavily on the definition. The language description is
a configuration mechanism that enables Pan to deliver services to the user which are based on the
underlying language structure and which can be tailored to the particular language. There may
be more than one Pan description for a given language definition, each delivering different services
(e.g. demonstration descriptions SIMPLE and SIMPLE2).

Language descriptions have many components, but they fall roughly into three categories: A
syntax description (for the syntactic analyzer), a semantic description (for the semantic analyzer),
and a specification of the editing interface (for the user's services).

- The syntax description consists of a Ladle file plus panic declarations. The Ladle description
  consists of a specification of the lexical tokens followed by a description of the grammar of
  the language in a format similar to YACC input format. These are accompanied by directives
to Pan on how to construct the abstract syntax tree [3].

- The semantic description is optional and consists of one or more ".col" (Colander) files. A
  Colander description consists of a set of goals associated with the rules of the grammar, each
  of which specifies a context-sensitive constraint that should be satisfied at each instance of
  a grammar rule in the internal tree. These are expressed in the language Colander (a Logi-
  cal Constraint Grammar [1], and are accompanied by supporting declarations and function
  definitions.
• the editing interface is configured mostly in a COMMON LISP run-time library file for each language named “<language>-mode.cl” This file may configure nearly all of the parameters described in Section 2.6, especially language-specific operand levels, operand menus, font maps, and other special-purpose functionality.

Further information on constructing these interface description files is found in Section 7.

Errors in a program being edited (with respect to the language description) are divided into two categories in Pan’s underlying implementation: syntactic errors and unsatisfied constraints. **Syntactic errors** are structural inconsistencies between the program and the Ladle syntax description. **Unsatisfied constraints**, also referred to sometimes (somewhat inappropriately) as semantic errors, are unsatisfied contextual constraints in the program, as specified in a Colander semantic description. These may include, for example, type checking (based on the language’s definition) and naming conventions (based on the language’s intended use).

Default error definition and handling is automatically provided by Pan. However, the error handling mechanism may be configured for each specific language description. For example both syntactic errors and unsatisfied constraints may be treated as one via a single operand level definition (e.g. as in demonstration language SIMPLE). Alternately, the two basic categories may be divided further, for example unsatisfied language constraints vs. stylistic constraints.

When you are editing a file with a complete Pan language description, Pan maintains:

• an internal structural representation of the program based on the Ladle description of the language which is updated incrementally with each analysis

• a database of semantic information, structural annotations, and other data derived from the program and the Colander description, also updated with each analysis

• a configuration that controls the look and feel of user interaction with the language as well as a number of user services that exploit the derived information

3 Basic Interaction

This section is an introduction to general kinds of interaction with the Pan system. 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”. The prefix “Escape” is denoted “Esc”.

Menu bindings are denoted by **MenuTitle: SelectionName** where MenuTitle is the title of the menu which appears at the top of the menu and SelectionName is the name of an item appearing in the menu. Operand-level bindings are denoted by **Command at level “OperandLevel”**. All of the configurable bindings are described in Section 2.6 and details of creating and altering bindings are contained in Section 7.4.
3.1 Getting Started

When *Pan* is started, the files named on the command line are read in as *Pan* text views and prepared for editing. Once initialization is complete, the view list will appear. Section 7.1 provides more specific details about *Pan*'s start-up processing.

3.2 Executing Commands

*Esc x*

Mouse-Right

Execute-Command
Execute-Menu

The simplest and most common way to execute a command is via a key or menu binding (see Section 2.6), but it is also possible to execute any *Pan* command by name. To execute a *Pan* command by name, type "Esc x", the keystroke sequence bound by default to Execute-Command. Execute-Command prompts for a command name. The prompt is case-sensitive, so you have to capitalize the command name correctly or it won't be recognized.

To select an item from a view's default menu, hold down "Mouse-Right" within the edit window of the view in which you want to execute the command. When the menu appears, continue to hold down the button and move the mouse over the menu item of your choice. When the item you want is highlighted, release the button. If you choose a sub-menu, do not release the button but move to the appropriate item in the sub-menu and repeat the process.

3.3 Quitting

Exit-Editor
Exit-Editor
Interrupt-Editor

"X" "C

View List: Quit

"X" "2"

Exit-Editor is the normal method for terminating a *Pan* session. When modified views exist you will be asked whether they should be saved.

Interrupt-Editor stops the system and control goes to a break loop in the underlying Common LISP. You can resume editing by typing ":cont". This is usually used for debugging.

3.4 Supplying 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 may use the selection or may prompt for an argument.

Numeric Prefix Arguments

Prefix

"U"
The behavior of many commands can be altered by supplying numeric prefix arguments. In most cases, the argument is interpreted as a repetition factor, and the effect of the command is simply repeated. Prefix reads the prefix arguments from the keyboard. There are two ways to type such arguments: by typing a sequence of digits, or by repeating the keystroke sequence bound to Prefix. In the latter case, each repetition corresponds to multiplication by the value of the option Prefix-Arg-Multiplier (default value 4). For example, the key sequence "*N*" is normally bound to the command Next-Line which takes a prefix argument. Typing "*U *N*" would cause the cursor to move forward four lines: typing "*U *U *N*" would cause the cursor to move forward 16 lines; typing "*U 3 *N*" would cause the cursor to move forward 3 lines.

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 visible, Pan is effectively stopped until the pop-up is dismissed. Pop-ups may be manipulated by the window manager, though.

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, 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. The “Return” key has the same effect as clicking on the button with the bold outline.

3.5 Cancelling Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>Editor-Error</td>
<td>^G</td>
</tr>
<tr>
<td>Editor-Error</td>
<td>^C^G</td>
</tr>
<tr>
<td>Editor-Error</td>
<td>Esc^G</td>
</tr>
<tr>
<td>Editor-Error</td>
<td>^X^G</td>
</tr>
<tr>
<td>Editor-Error</td>
<td>^Z^G</td>
</tr>
<tr>
<td>Editor-Error</td>
<td>&quot;Cancel&quot; on Pop-ups</td>
</tr>
</tbody>
</table>

To cancel a keystroke sequence, type "^G" with any prefix. To cancel a menu selection, move the mouse cursor outside (away from) the menu, and release the mouse button. When responding to a prompt, select the “Cancel” button.

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 clock.

3.6 Reviewing Message History

<table>
<thead>
<tr>
<th>Command</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>Prev-Message-Current-Window</td>
<td>Esc p</td>
</tr>
<tr>
<td>Next-Message-Current-Window</td>
<td>Esc n</td>
</tr>
</tbody>
</table>
3.7 Undoing Actions

These commands allow you to review the messages which have been displayed in a window's message line. Each time one of these commands is invoked, a message is displayed on the message line along with an index that indicates which message is being displayed: the most recent message is indexed [0], the one previous to that is [-1], etc.. A history of the significant messages which have been printed is stored for each window. The size of this history is controlled by the option Window-Message-History-Size whose default value is 32 messages. Insignificant messages are not part of the history, e.g. empty announcements that clear the message line and messages announcing that the system is garbage collecting.

3.7 Undoing Actions

```
Undo
Undo
Undo
Redo
Redo
Sub-Undo
Sub-Undo
```

The undo facilities of Pan allow you to undo the most recent action or series of similar actions. The effect of undoing an action is to return the text to the state it was in before that action was executed.

Invoking Undo will undo the most recent action. Invoking Undo a second time will undo the action previous to that. The command Redo will redo the last command undone. For instance, typing a series of characters and then invoking Undo will remove the entire series of characters just typed. Invoking the command Redo after the first Undo restores the text removed by the first Undo. A sequence of consecutive Undo commands are coalesced into a single action and so may be undone as one action after other commands have been executed.

Both Undo and Redo use the numeric prefix argument to determine the number of previous actions to undo or redo. The option Undo-Size specifies the maximum number of consecutive commands which can be undone.

Sub-Undo and Sub-Redo operate on commands that are subdivided into separate commands such as Replace-All and Query-Replace-All. Sub-Undo will undo the last portion of the previous command (i.e. the last replacement in Replace-All).

3.8 Getting Help

```
Reset-Help
Visit-Help-View
```

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

Most help commands delete any pre-existing text in the help view before adding their contribution. Others, such as Describe-Selection add additional information to the view. The information in the help view can be saved to a text file at any time. The command Reset-Help empties the help view. If there is not already a window open onto the help view, the command Visit-Help-View opens one window onto the help view, otherwise it brings the already open window to the front of the screen. This command is implicitly called every time information is printed to the help view.

Apropos

Apropos-String
Display-Apropos-Index

Help: Apropos
Not bound

The “Apropos-” family of commands associate a keyword with a list of command names. For instance, invoking the command Apropos-String and supplying the argument “clipboard” lists, in the help view, all of the commands that use the clipboard. Display-Apropos-Index \(^4\) will list all of the possible “Apropos” arguments and their corresponding commands.

Help Commands

Document-Symbol
Visit-Symbol-Definition
Visit-Symbol-Definition
Display-Objects
Display-All-Key-Bindings
Display-Key-Binding
Display-Key-Bindings-For-Command
Display-Default-Menu
Display-Operand-Level-Bindings
Display-Operand-Command-Bindings
Display-Operand-Level-Menus
Display-Option-Variables
Display-Flags
Display-Char-Sets
Display-Objects
Display-Language-List
Display-Commands
Display-Auto-Load
Display-Auto-Exec
Display-Version

\(^4\)Many help commands are of limited value to new or casual users, and are therefore not bound to the default menus. Invoking the command Debug adds auxiliary menus in which more help commands are available.
3.9 Views and Windows

Display-Bug-Report-Info
Display-View-Class-Key-Bindings
Display-Configuration
Display-File-Ids
Display-Hooks
Display-Notifiers

Help⇒Pan: How To Report Bugs
Not bound
Not bound
Not bound
Not bound
Not bound

Commands beginning with "Display-" simply list the names of all things of a given type for the current view. (e.g., all commands or all options). Commands in the This View sub-menu refer to things which are scoped and the commands give the values for those things in the current scope. Commands in the Pan sub-menu refer to things which are global to the editor. For more about Auto-Load and Auto-Exec see Section 7.1. For more about hooks and notifiers see section 7.8, and for setting options see Section 7.5.

Documentation Commands

Display-Command-Documentation
Display-Operand-Level-Documentation
Display-Operand-Command-Documentation
Display-Option-Documentation
Display-Char-Set-Documentation
Display-Variable-Documentation
Display-Constant-Documentation
Display-Function-Documentation
Display-Macro-Documentation
Display-Hook-Documentation

Not bound
Not bound
Not bound
Not bound
Not bound
Not bound
Not bound
Not bound
Not bound

"-Documentation" commands display the names of all things of a given type along with a documentation string.

3.9 Views and Windows

The commands presented in this section support basic view and window management.

Visiting Files and Other Objects

Visit-Selected-View
Visit-Selected-View
Visit-File
Visit-File
Visit-File
Protected-Visit-File
Visit-Backup
Visit-Checkpoint

View List: Visit View
f in View List
"X "F
View List: Visit New
Store: Visit New...
"X "R
Not bound
Not bound
Visit-Directory
Visit-Directory
Set-Working-Directory
Visit-View
Visit-View-List
Scratch-View
Rename-View
Remove-Selected-View
Remove-Selected-View
Remove-View
Remove-View
Remove-Object

"x ~D
"x d
Not bound

"x b
R7

View List: New Scratch
Not bound

View List: Remove
x in View List

"x k
View: Remove
Not bound

To edit a file, one must first create a view onto it with the command Visit-File or the command Dired-Visit (see below). Visit-File prompts for a file name. If a view does not already exist for that file, a new view is created for it and the file is read and prepared for editing.

To visit an existing view, you can use the Visit-Selected-View command from the view list. This command normally appears as Visit View in the View List menu. It is used by first selecting a name in the view list.

To edit a directory, either use Visit-File with the name of the directory or use the command Visit-Directory. The Visit-Directory command allows you to specify a pattern for file names. For example, if you give it the argument "~/pan/*.tex", it will give you the directory editor for "~/pan" listing only files whose names end in "*tex".

Visit-View prompts for the name of a view and opens a window for that view if it is in the view list. Scratch-View creates a new text view named "**scratch**" that is not associated with any file.

The Remove-Object command removes all views which share the given editable object and offers you a chance to save any changes.

Remove-Selected-View and Remove-View remove one view from the view list. If that is the only active view that exists on that data then it is removed as in Remove-Object. Once a "Remove-" command has been executed, all of the state associated with the removed object is lost.

Directory Editor

Dired-Visit
Dired-Visit
Dired-Visit-Read-Only
Dired-Visit-Read-Only
Dired-Mark-Selection
Dired-Mark-Selection
Dired-Unmark-Selection
Dired-Unmark-Selection
Dired-Delete-Files

Dired: Visit
f
Dired: Visit Read Only
"R
Dired: Mark
d
Dired: Unmark
u
Dired: Delete marked files
3.9 Views and Windows

Direc-Delete-Files
Direc-Mark-Editor-Files
Direc-Mark-Editor-Files
Revert-Object
Revert-Object
Visit-Directory
Visit-Directory

The directory editor view-style is useful for editing objects which are directories. To visit any file in the directory, select that file with the mouse or by normal cursor movement and invoke the command Direc-Visit. The command Direc-Visit-Read-Only works the same except that the view is protected so that its contents cannot be altered.

The directory editor allows you to mark files to be deleted with the command Direc-Mark-Selection then to delete those files with the command Direc-Delete-Files. Direc-Mark-Editor-Files marks for deletion all of the files in the directory which were created by Pan as backups.

Saving and Writing Files

Save-Object
Save-Object
Preserve-All-Objects
Exit-Editor
Save-Object-Copy-As
Save-Object-Copy-As
Write-Selection-To-File
Append-Selection-To-File
Revert-Object
Revert-Object
Backup-Object
Checkpoint-Object
Revert-Object-To-Backup
Revert-Object-To-Checkpoint
Toggle-Object-Protection
Toggle-Object-Protection
Toggle-Object-Modification

Save-Object saves the contents of the current view in its associated file; Preserve-All-Objects saves all of the modified views being edited. Exit-Editor performs the standard termination sequence of saving modified views and exiting. The commands Preserve-All-Objects and Exit-Editor prompt for whether a particular view is to be written to file.

All of the commands Save-Object-Copy-As, Write-Selection-To-File, and Append-Selection-To-File prompt for the name of a file to write. The command Save-Object-Copy-As does not rename the view.
Revert-Object replaces the contents of the object underlying the view in memory with the stored copy.

Toggle-Object-Protection toggles the Object-Protected? option in the view. When this option is set, the flag "@" appears on the information panel and you will be prevented from modifying or writing that file. This option is set by default when you commence editing a file for which you do not have permission to write. Toggling the Object-Protected? flag does not affect the permissions on the stored file itself.

If the option Backup-Object? is set, then the first time a file is saved, a backup of the original is created. The command Revert-Object-To-Backup can be used to restore the original file.

The file will be checkpointed after a number of modifications given by the option Checkpoint-Modification-Interval. The command Visit-Checkpoint allows you to edit the last checkpoint of the file and the command Revert-Object-To-Checkpoint allows you to revert the file to that state.

Toggle-Object-Modification allows you to toggle the Object-Modified? which appears as the flag "*" in the information panel.

Manipulating Windows

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Open-A-Window-In-View</td>
<td>Not bound</td>
</tr>
<tr>
<td>Open-Another-Window-In-View</td>
<td>Window: Open Another</td>
</tr>
<tr>
<td>Open-Another-Window-In-View</td>
<td>X 2</td>
</tr>
<tr>
<td>Close-Current-Window</td>
<td>X 0 Window: Close</td>
</tr>
<tr>
<td>Close-Current-Window</td>
<td>L Window: Redraw</td>
</tr>
<tr>
<td>Redraw-Current-Window</td>
<td></td>
</tr>
</tbody>
</table>

When you visit a view that has no visible windows, Pan will reopen the view's most recently closed window. If the view has no windows, a new window is created. To open a second (or third, or fourth, ...) window onto a view, execute the command Open-Another-Window-In-View.

Closing a window causes it to disappear from the screen. If the option Window-Close-With-Icon is set, then the window will be iconified when it is closed from within Pan. Closing a window from the window manager will always cause it to be iconified. The internal state of the view is retained even when it has no windows so that windows onto that view can be reopened later. You can not remove the view list.

Windows are reopened using the same commands as are used for opening new windows. They are reopened in last-in, first-out order relative to the order in which they were closed.

Redraw-Current-Window redraws the active window.

3.10 Scrolling

<table>
<thead>
<tr>
<th>Command</th>
<th>Keyboard Shortcuts</th>
</tr>
</thead>
<tbody>
<tr>
<td>Forward-Vscroll-Current-Window</td>
<td>^V</td>
</tr>
<tr>
<td>Backward-Vscroll-Current-Window</td>
<td>Esc v</td>
</tr>
<tr>
<td>Left-Hscroll-Current-Window</td>
<td>^X &lt;</td>
</tr>
<tr>
<td>Right-Hscroll-Current-Window</td>
<td>^X &gt;</td>
</tr>
</tbody>
</table>
3.11 When Things Go Wrong

**Scroll-To-Cursor-Current-Window**

Pan’s scrolling behavior is a simplified version of the X scrolling protocols. The scroll bars in a window respond to simple scroll commands. For vertical scrolling, pressing the left button in the vertical scroll bar scrolls the window toward the end of the file, the right button scrolls the window towards the beginning of the file, and the middle button thumbs the window to the point indicated by the scroll “bubble”. Holding down the middle button and moving the mouse allows you to scan or “thumb”, the file.

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

The first four of the above commands permit scrolling from the keyboard instead of from the mouse. During vertical scrolling, when the option **Text-Window-Proportional-Scroll** is set to be true (in COMMON 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 mouse 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 window.

When the option **Text-Window-Proportional-Scroll** is set to be false (in COMMON LISP, ’nil), the vertical scrolling commands scroll the window by a full screen at a time.

To “thumb” the view to an absolute position, place the mouse cursor in the scroll bar and press **Mouse-Middle**. The window will be scrolled to the position corresponding to the 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-Current-Window** and **Backward-Vscroll-Current-Window**. Both of those commands consult the numeric prefix argument to determine the number of screens to move.

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

The command **Scroll-To-Cursor-Current-Window** redraws the window and scrolls so that the cursor appears in the center of the window. The position of the cursor is not affected by scrolling. Therefore, scrolling may cause the cursor to be not visible in the window.

Any cursor motion or alteration of the text will scroll the window so that the cursor is visible.

3.11 When Things Go Wrong

:panic
:resume

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 view. However, 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 COMMON LISP system or by dying altogether. Fortunately, the first rarely happens, and the second won't occur without returning to COMMON LISP.
If an unanticipated Common Lisp error occurs, Pan enters a Common Lisp break loop. If you do somehow end up in a Common 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.) The prompt will look like "\{nn\}" or "\{nn \Rightarrow \}" where "\{nn\}" is a small integer.

In the first case ("\{nn\}"), the system is in a Common Lisp break loop. You can recover from the error by typing the Common Lisp expression :resume which returns the system to the normal command evaluation loop or :panic which executes the normal code for saving the modified views and then exits the system. Naturally, the circumstances of such an error should be noted and passed on to the developers of Pan. (Mail to pan-bug@sequoia.berkeley.edu)

The second case ("\{nn \Rightarrow \}" ) is more serious. In fact, the session is almost over. All that you can do is to type :panic.

4 Using Textual Views

This section describes the basic text-oriented services that are available in Pan's textual views. These correspond in many ways to traditional text editors. and users of Emacs will find much that is familiar. Pan's language-based services (also available in textual views) are described in Section 6.

4.1 Text Cursor Motion

\verb|Announce-Text-Cursor-Info| \phantom{^X=}

The command Announce-Text-Cursor-Info can be used to display status information in the information panel about the text cursor in the active window.

Moving the Cursor With the Mouse

\verb|Oplevel-Cursor-To-Mouse| \phantom{^C Mouse-Left}
\verb|Cursor-To-Mouse| \phantom{Mouse-Left}
\verb|Cursor-To-Mouse| \phantom{Cursor-To-Mouse at level "Character"}
\verb|Mouse-Select-Fullword| \phantom{Cursor-To-Mouse at level "Word"}
\verb|Mouse-Select-Line| \phantom{Cursor-To-Mouse at level "Line"}

The left button of the mouse is bound to the command Cursor-To-Mouse which sets the cursor to the character selected by the mouse icon. This command does not affect the current selection.

The corresponding operand level command Oplevel-Cursor-To-Mouse is normally bound to the key sequence "^C Mouse-Left". When the operand level is set to "Character", it invokes the command Cursor-To-Mouse. At different operand levels, Oplevel-Cursor-To-Mouse selects the operand under the mouse and places the text cursor at the beginning of the selection.

Moving Forward and Backward

\verb|Oplevel-Cursor-Forward| \phantom{^C ^F}

4.1 Text Cursor Motion

Oplevel-Cursor-Forward
Next-Character
Next-Character
Next-Word
Next-Word
Next-Line
Next-Line
Oplevel-Cursor-Backward
Oplevel-Cursor-Backward
Previous-Character
Previous-Character
Previous-Word
Previous-Word
Previous-Line
Previous-Line
Oplevel-Cursor-In
Oplevel-Cursor-In
Next-Line
Next-Line
Next-Line
Oplevel-Cursor-Out
Oplevel-Cursor-Out
Previous-Line
Previous-Line
Previous-Line
Previous-Line

Right-Arrow
"F"
Cursor-Forward at level “Character”
Esc f
Cursor-Forward at level “Word”
"N"
Cursor-Forward at level “Line”
"C "B
Left-Arrow
"B"
Cursor-Backward at level “Character”
Esc b
Cursor-Backward at level “Word”
"P"
Cursor-Backward at level “Line”
"C "I
Down-Arrow
Oplevel-Cursor-In at level “Character”
Oplevel-Cursor-In at level “Word”
Oplevel-Cursor-In at level “Line”
"C "O
Up-Arrow
Oplevel-Cursor-Out at level “Character”
Oplevel-Cursor-Out at level “Word”
Oplevel-Cursor-Out at level “Line”
"C Esc <
Cursor-To-First at level “Character”
Esc <
Cursor-To-First at level “Word”
Cursor-To-First at level “Line”
"C Esc >

The cursor can also be moved using cursor motion commands shown above. If the operand level were “Word”, then Oplevel-Cursor-Forward would move the cursor to the word following the word containing the cursor and Oplevel-Cursor-To-First would move the cursor to the first word in the view. The “Next-” and “Previous-” commands use the numeric prefix argument to determine the number of units to move.

Note that the generic commands Cursor-In and Cursor-Out (bound by default to “Down-Arrow” and “Up-Arrow” respectively) have little meaning for ordinary textual editing. They are bound to Next-Line and Previous-Line respectively at all three textual operand levels.

Moving Around in the Text Stream
Move-To-Eos
Move-To-Eos
Move-To-Last-Word
Move-To-Last-Line
Move-To-Bol
Move-To-Eol
End-Of-Word
First-Non-Blank
Goto-Line

Move-To-Bol moves the cursor to the first position on the current line. Move-To-Eol moves the cursor to the last position on the current line. To move to the beginning or end of the view being edited, use Move-To-Bos or Move-To-Eos, 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 beginning of 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.

Mark Commands

Set-Mark
Pop-Mark
Push-Mark
Swap-Dot-And-Mark
Dot-To-Mark

Set-Mark and Push-Mark both push the location of the cursor to the top of the mark stack of the active view. Pop-Mark pops the mark stack of the active view. The commands Swap-Dot-And-Mark and Dot-To-Mark move the text 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.

4.2 The Current Operand Level

Set-Oplevel-To-Character
Set-Oplevel-To-Word
Set-Oplevel-To-Line

The current operand level is a mode which affects the subsequent operation of operand level commands (a.k.a. level-sensitive commands), see Section 4.3. The level does not affect the operation
4.3 Operand Level Commands

of any other commands: in particular, text may be entered and manipulated freely, independent of level.

The operand level may be set by using these commands or selected using "Mouse-Right" over the menu in the information panel where the current level is always visible. You can leave the operand level at "Character", and Pan will operate much like Emacs.

Other operand levels related to languages are described in Section 6.4.

4.3 Operand Level Commands

```
Oplevel-Cursor-Forward  C F
Oplevel-Cursor-Backward C B
Oplevel-Cursor-To-First C Esc <
Oplevel-Cursor-To-Last C Esc >
Oplevel-Cursor-To-Mouse C Mouse-Left
Oplevel-Select          C Backspace
Oplevel-Mouse-Extend    C Mouse-Middle
Oplevel-Delete         C D
Oplevel-Cursor-Search-Forward C S
Oplevel-Cursor-Replace-All  C R
Oplevel-Query-Replace-All  C Esc R
Oplevel-Menu           C Mouse-Right
```

These are generic commands whose behavior depends on the current operand level and on operand level bindings (both command and menu). They may be bound to different commands at any operand level in any given scope. Note that the key bindings for these commands are the same as the key bindings for the analogous character level commands prefixed by the key sequence "C".

The first five commands control cursor movement relative to the operand level: see Section 4.1 for more detail. `Oplevel-Select` selects the operand under the edit cursor. `Oplevel-Mouse-Extend` extends the selection to include the mouse position; see Section 4.4. `Oplevel-Delete` deletes the operand under the edit cursor. The "Search" and "Replace" commands are discussed in Section 4.13.

The operand level menu is specific to each particular operand level and usually contains all of the commands which are bound to the generic commands by default at that operand level.
4.4 The Textual Selection

<table>
<thead>
<tr>
<th>Command</th>
<th>Equivalent</th>
</tr>
</thead>
<tbody>
<tr>
<td>Oplevel-Cursor-To-Mouse</td>
<td>C Mouse-Left</td>
</tr>
<tr>
<td>Oplevel-Mouse-Extend</td>
<td>C Mouse-Middle</td>
</tr>
<tr>
<td>Oplevel-Select</td>
<td>Not bound</td>
</tr>
<tr>
<td>Cursor-To-Mouse</td>
<td>Mouse-Left</td>
</tr>
<tr>
<td>Select-Region-Dot-To-Mark</td>
<td>Esc ^W</td>
</tr>
<tr>
<td>Select-Region-Dot-To-Mouse</td>
<td>Mouse-Middle</td>
</tr>
<tr>
<td>Select-Region-Dot-To-Mouse</td>
<td>Mouse-Extend at level &quot;Character&quot;</td>
</tr>
<tr>
<td>Select-Word</td>
<td>Esc @</td>
</tr>
<tr>
<td>Select-Line</td>
<td>Select at level &quot;Line&quot;</td>
</tr>
<tr>
<td>Select-Fullword</td>
<td>Select at level &quot;Word&quot;</td>
</tr>
<tr>
<td>Select-Full-Line</td>
<td>Not bound</td>
</tr>
<tr>
<td>Select-Text-Stream</td>
<td>^X h</td>
</tr>
<tr>
<td>Mouse-Select-Word</td>
<td>Not bound</td>
</tr>
<tr>
<td>Mouse-Select-Line</td>
<td>Cursor-To-Mouse at level &quot;Line&quot;</td>
</tr>
<tr>
<td>Mouse-Select-Fullword</td>
<td>Cursor-To-Mouse at level &quot;Word&quot;</td>
</tr>
<tr>
<td>Mouse-Select-Fullword</td>
<td>Esc Mouse-Left</td>
</tr>
<tr>
<td>Mouse-Select-Full-Line</td>
<td>Not bound</td>
</tr>
<tr>
<td>Mouse-Extend-Selection-Fullword</td>
<td>Esc Mouse-Middle</td>
</tr>
<tr>
<td>Mouse-Extend-Selection-Full-Line</td>
<td>Not bound</td>
</tr>
<tr>
<td>Mouse-Extend-Olevel-Text</td>
<td>Mouse-Extend at level &quot;Word&quot;</td>
</tr>
<tr>
<td>Mouse-Extend-Olevel-Text</td>
<td>Mouse-Extend at level &quot;Line&quot;</td>
</tr>
<tr>
<td>Next-Line-Select</td>
<td>Not bound</td>
</tr>
<tr>
<td>Previous-Line-Select</td>
<td>Not bound</td>
</tr>
<tr>
<td>Select-Expr</td>
<td>Esc @</td>
</tr>
<tr>
<td>Null-Select</td>
<td>Not bound</td>
</tr>
<tr>
<td>Deselect-Region</td>
<td>Select at level &quot;Character&quot;</td>
</tr>
<tr>
<td>Deselect-Region</td>
<td>Esc ^D</td>
</tr>
</tbody>
</table>

Each textual view has a single textual selection, shared by all windows. It may be set in any window, but it is visible in all windows. Any alteration of the text stream deselects the textual selection. The text selection is generally set by the commands Select-Region-Dot-To-Mouse, Oplevel-Cursor-To-Mouse, Oplevel-Mouse-Extend, and Select-Region-Dot-To-Mark. Select-Region-Dot-To-Mouse is bound to "Mouse-Middle" and selects the region between the text cursor and the mouse. Oplevel-Cursor-To-Mouse is bound to the key sequence "C Mouse-Left" and selects one unit at the current operand level beneath the mouse cursor. Oplevel-Mouse-Extend is normally bound to the key sequence "C Mouse-Middle". When the level is "Character", this command selects the region between the text 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 text cursor.
4.5 Entering Text

"-Fullword" commands select the entire word back to the first letter. Likewise, "-Full-Line" commands select the entire line from the first character in the line.

The command Select-Word selects the region from the text cursor to the end of the word surrounding the text cursor, while Mouse-Select-Fullword selects the full word beneath the mouse cursor. Mouse-Extend-Selection-Fullword extends the text selection to include the full word beneath the mouse cursor. Select-Text-Stream selects all the text in the view.

When the cursor is positioned on a left brace, bracket, or parenthesis, Select-Expr selects the region from that character to the matching right brace, bracket, or parenthesis.

Null-Select creates a selection with one blank at the cursor.

The commands Select-Line and Select-Full-Line consult the numeric prefix and select that number of lines. The commands Next-Line-Select and Previous-Line-Select also consult the numeric prefix and move that number of lines forward or backward before selecting one line.

The current selection can be cleared using Deselect-Region.

4.5 Entering Text

Self-Insert
Quote-Character
Newline-And-Indent
Insert-Newline
Indent-Like-Previous-Line
Center-Line
Open-Line
Split-Line
Insert-Parentheses
Insert-File
Insert-File

most printable characters

```
^Q
LineFeed
Return
Esc Tab
Not bound
^D
Esc ^D
Esc (  
^X Tab
Text Edit⇒File: Insert Text from...
```

Typing a printable character generally causes that character to be inserted into the text at the position of the cursor. When printable characters are typed consecutively, the Self-Insert commands are merged into a single action for the purpose of Undo operations. Quote-Insert inserts the next ASCII character typed even if it is a control character.

Newline-And-Indent is bound to the "LineFeed" character; the next line will be indented to the level of the previous line by inserting tabs and blanks. Center-Line centers the current line between the beginning of the line and the value of the Text-Line-Length. Indent-Like-Previous-Line simply reindents the current line to the level of the previous line.

Open-Line inserts "newline" characters after the cursor; the number of "newline" characters 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.

Insert-File prompts for a file name and copies the contents that file into the active view at the position of the text cursor.
4.6 Filling Text

Set-Text-Line-Length
Toggle-Text-Fill
Insert-Space-Fill
Insert-Newline-Fill
Fill-Selected-Lines
Set-Text-Fill-Prefix
Text-Fill-To-Blank-Line

To fill text means to move text from line to line so that all the lines are approximately the same length. Pan has a rudimentary mechanism for filling text lines as you type them. When the Text-Fill option is on, the flag “✓” appears in the information panel and the keys “Space” and “Return” are bound to special procedures. These procedures compare the current horizontal position of the cursor with the value of Text-Line-Length. If the line is too long, it will be broken where appropriate; if not, the procedures act like Self-Insert. Use Toggle-Text-Fill to turn on text filling and again to turn it back off. When filling is on, you can type text continuously without worrying about line length.

Use Set-Text-Line-Length (with a numeric prefix argument) to set the maximum line length used by auto-filling.

The commands Fill-Selected-Lines and Text-Fill-To-Blank-Line fill more than one line at a time. The first fills the selection: the second fills all lines from the cursor to the first blank line following the cursor.

Set-Text-Fill-Prefix sets the line prefix to the characters before the cursor on the line with the cursor. This prefix is then added to the beginning of each new line as it is started. This is useful for indenting an entire section of text or for prefixing each line with a fixed string of characters.

4.7 Deleting Text

Oplevel-Delete
Delete-Character
Delete-Character
Delete-Previous-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-Horizontal-Space
Just-One-Space
Delete-Indentation

Delete at level “Character”
Delete at level “Word”
Delete at level “Line”

Delete at level “Character”
Delete
Backspace
Not bound
Not bound

Delete at level “Word”
Not bound
Not bound

Delete at level “Line”
Not bound
Not bound

^C ^D
^D

Delete at level “Character”
Delete
Backspace
Not bound
Not bound

Delete at level “Word”
Not bound
Not bound

Delete at level “Line”
Not bound
Not bound

^X ^O
Esc \n
Esc Space
Esc ~
Deleted text can be recovered by issuing an "Undo" command immediately after the deletions, but in no other way. Deleted text is not 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 undo-able action.

**Oplevel-Delete** deletes one unit at the current operand level. Thus, if the operand level is "Character", **Oplevel-Delete** deletes one character. **Delete-Character** deletes the character under the cursor. **Delete-Previous-Character** deletes the character before the cursor. The pair of commands **Delete-Word** and **Delete-Previous-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.

### 4.8 Killing Text

<table>
<thead>
<tr>
<th>Command</th>
<th>Binding</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>&quot;K</td>
</tr>
<tr>
<td>Kill-Expr</td>
<td>Esc &quot;K</td>
</tr>
<tr>
<td>Kill-Selected-Region</td>
<td>&quot;W</td>
</tr>
<tr>
<td>Kill-Selected-Region</td>
<td>Text Edit: Kill</td>
</tr>
<tr>
<td>Kill-Region-Dot-To-Mark</td>
<td>Not bound</td>
</tr>
<tr>
<td>Copy-Selection-As-Kill</td>
<td>Esc w</td>
</tr>
<tr>
<td>Yank-From-Kill-Ring</td>
<td>&quot;Y</td>
</tr>
<tr>
<td>Yank-From-Kill-Ring</td>
<td>Text Edit: Yank</td>
</tr>
<tr>
<td>Cycle-Yank</td>
<td>Esc y</td>
</tr>
<tr>
<td>Cycle-Kill</td>
<td>Not bound</td>
</tr>
<tr>
<td>Show-Kill</td>
<td>&quot;X ?</td>
</tr>
<tr>
<td>Cycle-Show-Kill</td>
<td>&quot;X !</td>
</tr>
</tbody>
</table>

Like the deletion commands of the previous section, commands that kill text remove the text from a view. Unlike deletion commands, however, these also copy 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 command **Kill-Expr** kills the text of the next word or of the balanced bracket expression following the text cursor. The commands **Kill-Selected-Region** and **Kill-Region-Dot-To-Mark** kill the current textual 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 copies the contents of the top item in the kill ring into the view at the current cursor position. If a prefix argument is present, then `Yank-From-Kill-Ring` will replace the current selection with the copied item.

`Cycle-Yank` cycles the kill ring (see Section 2.4) before yanking the top of the kill ring. `Copy-Selection-As-Kill` copies the current selection to the kill ring without removing it from the text stream.

The command `Show-Kill` displays the first item in the kill ring. The commands `Cycle-Kill` and `Cycle-Show-Kill` both cycle the kill ring; `Cycle-Show-Kill` also shows the new top item.

The maximum number of items stored in the kill ring is determined by the option `Killring-Max-Size`. If the option `Kills-To-Clipboar`d is set then all kill commands use the global clipboard instead of the view-specific kill rings.

### 4.9 The Clipboard

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cut-To-Clipboard</td>
<td>Clipboard: Cut</td>
</tr>
<tr>
<td>Cut-To-Clipboard</td>
<td>L10</td>
</tr>
<tr>
<td>Copy-To-Clipboard</td>
<td>Clipboar: Copy</td>
</tr>
<tr>
<td>Copy-To-Clipboard</td>
<td>L6</td>
</tr>
<tr>
<td>Paste-From-Clipboard</td>
<td>Clipboard: Paste</td>
</tr>
<tr>
<td>Paste-From-Clipboard</td>
<td>L8</td>
</tr>
<tr>
<td>Replace-From-Clipboard</td>
<td>Not bound</td>
</tr>
<tr>
<td>Cycle-Clipboard</td>
<td>Not bound</td>
</tr>
<tr>
<td>Show-Clipboard</td>
<td>Clipboard: Show Contents</td>
</tr>
<tr>
<td>Show-Clipboard</td>
<td>Esc ?</td>
</tr>
<tr>
<td>Cycle-Paste</td>
<td>Not bound</td>
</tr>
<tr>
<td>Cycle-Show-Clipboard</td>
<td>Clipboard: Cycle and Show</td>
</tr>
<tr>
<td>Cycle-Show-Clipboard</td>
<td>Esc !</td>
</tr>
</tbody>
</table>

These commands manipulate the contents of the global clipboard shared by all views. `Cut-To-Clipboard` deletes the current textual 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 top item of the clipboard at the text cursor. If a prefix argument is present, then `Paste-From-Clipboard` will replace the current selection with the top item of the clipboard. Finally, `Replace-From-Clipboard` replaces the selected region with the top item of the clipboard.

The command `Show-Clipboard` displays the first item of the clipboard in the help view. `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. `Cycle-Paste-Clipboard` cycles the clipboard and pastes the new top element into the text stream at the cursor.

The maximum number of items in the clipboard is determined by the option `Clipboard-Max-Size`.

### 4.10 Copying and Moving Text

`Copy-Selection-To-Cursor`
4.11 Changing Case

Copy-Selection-To-Cursor
Move-Selection-To-Cursor

Text Edit: Copy To Cursor
Esc Return
Text Edit: Move To Cursor

Text can copied or moved within a textual view 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 text cursor to mark the destination. These commands operate only within a view; copying and moving between views requires the clipboard.

4.11 Changing Case

Capitalize-Word
Lowercase-Word
Uppercase-Word
Capitalize-Selection
Lowercase-Selection
Uppercase-Selection

Esc c
Esc l
Esc u
Esc ^C
Esc ^L
Esc ^U

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 and leave the cursor at the end of the word when done. The selection-oriented commands leave the cursor at the beginning of the affected selection.

4.12 Transposing Text

Transpose-Characters
Transpose-Characters-At-Cursor
Transpose-Lines

^T
Not bound
^X ^T

Transpose-Characters exchanges the character at the cursor and the character before the cursor unless the cursor is at the end of a line, in which case it transposes the previous two characters. Transpose-Characters-At-Cursor the character at the cursor and the character before the cursor even if the cursor is at the end of a line. Transpose-Lines exchanges the line containing the cursor with the line before it.

4.13 Searching Text

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

Regular Expressions

Oplevel-Cursor-Search-Forward
Oplevel-Cursor-Search-Forward
Re-Search-Forward

L9
^C ^S
^S
These commands prompt for an expression argument and then 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 views.

The "Oplevel-Cursor-Search" and "Oplevel-Replace" commands step through each unit at the given level and try to match the argument pattern to the given unit. This has more meaning when you are using the operand levels associated with a programming language (see Section 6.4). When you are using a program operand level, the pattern must match the entire operand unless a numeric prefix is present, in which case it may match a subset of the operand. The "Character" and "Word" levels allow the pattern match to span more than one operand. When text matching a pattern is found, the text cursor is moved to the first character in the match, and the matched text is selected.

Both Re-Search-Forward and Re-Search-Backward, the text level search commands, 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, "U 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 view to the other; if that option is false, searches terminate when finding the beginning or end of the view.

The "Replace" commands search for a string and replace all instances of it with the specified replacement string. Oplevel-Query-Replace-All asks the user whether to replace, ignore, or cancel the search at each instance of the sample string.

The set of replacements made during a "Replace" command are grouped together as a single action for the purpose of "Undo" operations. However, the Sub-Undo command allows you to undo the individual replacements one at a time (see Section 3.7).
Balanced Bracket Commands

Backward-Expr
Forward-Expr
Select-Expr
Kill-Expr
Show-Match

Esc "B
Esc "$F
Esc宝
Esc "$K
Esc "$%

These commands use character set definitions to operate on balanced bracket expressions. The set match-characters contains all of the characters which are defined to have matches. 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 in the set match-characters. 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. If the option Auto-Show-Match is set then Show-Match is called every time a right bracket from the match-characters set is typed.

5 Using Graphical Views

Zoom-In
Zoom-In
Zoom-In
Zoom-Out
Zoom-Out
Zoom-Out
Zoom-Reset
Zoom-Reset
Zoom-Reset
Global-Image-Toggle
Global-Image-Toggle
Global-Image-Enlarge
Global-Image-Enlarge
Global-Image-Shrink
Global-Image-Shrink

"in" button in control panel
Window: Zoom In

"out" button in control panel
Window: Zoom Out

"all" button in control panel
Window: Zoom All

A graphical view can be created to represent the internal tree representation of a program or other structure-based view. The textual label on the graph nodes vary depending on which command was used to generate the view. This view functions only as a display and can not be used to alter the contents of the textual view from which it was generated. Other graphical views are anticipated, but not yet implemented.

Currently, the graphical views can be navigated only by zooming in and out via the commands Zoom-In and Zoom-Out, which are bound to the in and out buttons on the control panel, and scrolling with the scroll bars on the window frame. The Zoom- commands scale the view up or down by a factor of Graph-Window-Zoom-Factor (whose default value is 0.1). The "Zoom-"
commands take a numeric prefix and effectively zoom in or out that many times to produce more drastic scale changes. Only the middle button of the mouse functions for scrolling in a graph view. The left and middle buttons of the mouse have no effect when the mouse is over the display area, but the right button still invokes a menu.

The global image shows the entire graph being viewed and marks the portion of the graph which is visible in the window. It appears in the upper left corner of the graph view window. It may be resized using Global-Image-Enlarge and Global-Image-Shrink and it may be toggled on and off using the command Global-Image-Toggle.

6 Language-Oriented Services

In addition to its more conventional text-based services, Pan provides facilities for viewing and modifying textual objects whose underlying structure can be described by formal languages, for example programs. These services require the loading of one or more Pan language descriptions, as discussed in sections 2.7 and 7.1. In no case, however, does the presence of a language description inhibit the operation of Pan’s text facilities; users need incur no loss of convenience to benefit from Pan’s more advanced features.

The basic interface provided by Pan for language-oriented editing is uniform across languages. It is also configured specially for each language and can be customized further for any particular language and for any style of interaction. Finding an appropriate balance between uniformity and specialization is encouraged by Pan’s configuration mechanisms, but by no means assured. This is a matter of user interface design on the part of the author of Pan’s language descriptions, and is the subject of ongoing research.

This section introduces Pan’s basic language-oriented editing features. Many of these are experimental and more are under development. A more thorough discussion of these features appears elsewhere [8].

All interaction with a program takes place in the context of a view on that program, where both textual and graphical views are supported in the current implementation. It is intended, however, that ordinary programming take place in the context of a primary textual view, a view that operates very much like an ordinary text editor but with many extra language-based services available.

Pan’s basic language-oriented services include a number of program views that will be of primary interest to system extenders and especially to authors of language descriptions, see Section 6.12. For example, different tree views can be created which contain subtrees of the entire structure or which display different information at the tree nodes. Views can also be created which display information about the lexical stream or the semantic database. These alternate views are usually updated every time the program is reanalyzed.

Ladle syntax descriptions have been written for Modula-2[9]. Pascal, Ada, Ladle, Colander, and ASPLE[4]. ASPLE is a simple example language used for demonstrations and for learning Pan. Other language descriptions, including C, C++, and FIDIL[6], are under development. A Colander semantic description has been written for Modula-2, as well as some small demonstration languages, including Tiny and Simple. SIMPLE2 is a second language description for the language Simple which provides additional and somewhat different services to the user.
Figure 3 shows a Pan session involving two views on a SIMPLE2 program: one textual and one graphical.

Figure 3: Two language-based views on a program

In contrast to earlier examples, the "Language:" field in the information panel now reads "SIMPLE2", revealing that the language description has been loaded and is being used for this view. The default menu has been augmented: new submenus Language and Variables appear and the View menu contains additional submenus Program Graph and Subgraph At Cursor. The Variables menu provides access to commands which allow you to query or rename a variable or to visit its definition, while the additional View menus provide quick access to the graphical language-based views. As with text views, many of these commands are also available from the keyboard.

6.1 Language-Based Textual Views

Most user interaction with programs takes place in the context of what appears at first glance to be an ordinary Pan textual view. for example the view whose window appears at the upper left of Figure 3 on page 35. In fact, all of the conventional text-based support provided by Pan continues to available at all times with no restrictions. The language-based information derived by Pan's analyzers is kept behind the scenes. The information is used to support a variety of user services, but
only when those services are requested. Whenever structural information is required for execution of a command, incremental syntactic analysis (parsing) 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. Whenever possible, the language-based services provided by Pan are similar in form, name, and default key binding to analogous text operations.

The information panel in windows owned by language-based views differs slightly from those for purely text-based views. For example, the name of the language description being used appears in place of “Text” in the “Language:” field.

Two additional flags reveal the presence and status of derived information:

- The “✓” (Parse-is-Current?) flag is visible, but gray, when there are textual changes that have not yet been incorporated into the tree by the syntactic analyzer, that is when the tree is present but inconsistent with the text. The flag appears solid when text and structure are consistent.

- The “D” (Database-Updated?) flag is visible, but gray, when there are changes that have not yet been incorporated into the semantic database. This flag is absent entirely when there is no Colander component to the language description being used. The flag appears solid when the database is consistent with text and structural information.

Pan’s default behavior at present for language errors (that is, for situations where a program violates part of the language’s definition) is to show the user two categories of error, reflecting Pan’s underlying implementation. In the default configuration two additional flags reveal the presence of language errors:

- The “!” (Syntax-Errors-Present?) flag appears when there are lexical or syntactic errors in the program being edited.

- The “#” (Semantic-Errors-Present?) flag appears when there are semantic errors in the program being edited.

It is possible, however, to configure a language description so that the user only sees one category of error in place of the two. Section 6.8 discusses this topic in more detail.

6.2 Language-Based Analysis

Language: Analyze
Analyze-Changes
Analyze-Changes
Visit-And-Check-File

Language-based analysis is the process of deriving information from text based on an underlying description of the structure of the language, a set of contextual constraints, and possibly other information.

The analysis method used is incremental—only the areas affected by the changes are reanalyzed. Analysis occurs whenever the derived information is out of date with respect to the text and when a language-oriented operation takes place, for example when the user invokes a tree-oriented
navigation command. Analysis can also be invoked explicitly, with the command Analyze-Changes. This command is also used internally to invoke analysis if there have been any changes in the contents of the view. Visit-And-Check-File performs the same function as Visit-File but also analyzes the view’s contents.

Syntactic analysis incorporates changes in a view’s contents into the tree that represents a program. Syntactic analysis 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 view 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. These codes permit the assignment of specific fonts (via the font map mechanism, see Section 7.5) to each category for each language.

<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>keywords</td>
<td>2</td>
</tr>
<tr>
<td>Fixed-length lexemes</td>
<td>comments</td>
<td>3</td>
</tr>
<tr>
<td>Recognized but not analyzed</td>
<td>identifiers</td>
<td>4</td>
</tr>
</tbody>
</table>

Semantic analysis checks a program against a set of contextual constraints defined in the Colander component of the language description. These constraints may be based on the language definition, for example type checking, on the intended use, for example naming conventions, and many other restrictions as needed. At the same time, semantic analysis records facts about the program in the semantic database that are used to support various services.

6.3 Consistency and Inconsistency

Unlike syntax directed editors, which don’t permit you make a mistake or restrict your options when you do, Pan allows the text to be modified without restriction at all times. When text has been changed since last analysis, the derived information is said to be inconsistent: it should be considered unreliable and may be incomplete. In this situation, the Parse-Is-Current? and Database-Updated? flags appear gray and newly entered unanalyzed text is displayed using the default font 0. Any text which has previously been analyzed and assigned a non-default font remains in that font until it is reanalyzed. Also any text which has been highlighted as a syntactic or semantic error remains highlighted until it is reanalyzed. When derived information becomes inconsistent, the structure cursor is cleared (see Section 6.5).

Pan’s current implementation requires consistency for any structural operation. Therefore, reanalysis takes place whenever any structural information is needed. Work is underway to loosen this restriction.

Note that derived information may be consistent or inconsistent whether or not there are language errors (Pan separates the two), see Section 6.8.

\footnote{No single combination of font assignments seems to be best for all languages. Designing a font map is part of the user interface design work for the author of a Pan language description.}
6.4 Operand Levels

- Set-Oplevel-To-Lexeme: \texttt{C \textasciicircum x}
- Set-Oplevel-To-Syntactic-Error: \texttt{C !}
- Set-Oplevel-To-Semantic-Error: \texttt{C \#}
- Set-Oplevel-To-Query: \texttt{C q}
- Set-Oplevel-To-Node: Not bound
- Oplevel-Menu: \texttt{C Mouse-Right}

Unlike simple textual views in which only the three textual operand levels are available, the list of operand levels available in a language-based text view is part of Pan's configuration for the particular language, defined by \texttt{Operand-Level-Choices}. This list defines what components of a program (more accurately, of Pan's representation of a program) that the user "sees" and can manipulate.

Some of the operand levels are standard across all languages, for example the three textual levels and the lexical level. Some levels have the same or similar names as analogous levels in other languages (e.g. statement, expression), but their internal definitions differ according to the underlying language description.

These are the operand levels for the example language, \texttt{SIMPLE2}: "Lexeme", "Expression", "Statement", "Declaration", "Placeholder", "Syntactic Error", "Unsatisfied Constraint", and "Query". Figure 4 on page 39 shows the set of operand levels defined for \texttt{SIMPLE2}.

All of the previously defined Oplevel commands may have bindings at these levels. Thus cursor motion, selection, deletion, and searches may all be effected relative to the structure of the program. The operand levels for a language are related to nodes in the Ladle description of the language. They must be specified in the language-mode file which is loaded in and executed when a view in that language is edited. (See \texttt{Auto-Load} and \texttt{Auto-Exec} in Section 7.1) To see the available operand levels for a language, simply inspect the panel "Level:" menu. There are also key bindings designed to permit level selection from the keyboard: standard bindings for the standard levels, language-specific ones for the language-specific levels.

There are menus available for each level, executable by the command \texttt{Oplevel-Menu}, which make available many of the same generic commands that have key bindings. In some cases, the level-specific menus may be extended in ways specific to each language.

The command \texttt{Set-Oplevel-To-Node} is used for debugging. It sets the operand level to all nodes of the internal tree, whereas most operand levels define useful subsets of the internal tree. Note however that this command, like all \texttt{Set-Oplevel} commands, works only if the specified level is permitted in the view, as controlled by option \texttt{Operand-Level-Choices}.

6.5 The Structure Cursor

- Clear-Structure-Cursor: \texttt{Esc \textasciicircum D}
- Cursor-To-Token: Not bound
- Announce-Structure-Cursor-Info: \texttt{C \textasciicircum X =}
- Announce-Structure-Cursor-Node-Rule: Not bound
The structure cursor is a reference to one component in the internal tree representation of a structured object: it is shared by all views on a program. At any point the structure cursor may or may not be present. Currently, the structure cursor is only usable when the derived information is consistent with the text. When an operand level command is invoked and the operand level corresponds to a program structure type (such as "Lexeme" or "Expression"), the structure cursor is set to the current node at that operand level. Commands which operate on a structure node, such as "Visit-Program-Subgraph-" commands, take the structure cursor's node as their argument.

Whenever the structure cursor is set, the textual selection will also be set to the region of text which corresponds to the structure cursor and the text cursor in the active window will be moved to the beginning of that region. If the option Highlight-Structure-Cursor is set, then the region of text in the program which corresponds to the structure cursor will be visible as a light blue highlighted region. The highlight color for the structure cursor may be set to any color by setting the option Text-Window-Bg-Colormap in the appropriate view (see Section 7.5). The text cursor

---

6 Setting or clearing the text selection explicitly clears the structure cursor as a side effect.
may be moved away from the structure cursor without clearing it, but any alterations to the text stream render the derived information inconsistent and clear the structure cursor.

In some cases at certain operand levels (this is configurable), an announcement may accompany positioning the structural navigation at a node, for example error diagnostics (see Section 6.8).

Clear-Structure-Cursor clears the structure cursor without clearing the textual selection. Cursor-To-Token sets the cursor to be the token which contains the text cursor regardless of the current operand level. Announce-Structure-Cursor-Info displays information about the node at the structure cursor in the information panel. Announce-Structure-Cursor-Node-Rule displays information in the information panel about the reduction rule in the language for the node at the structure cursor (generally used for debugging).

6.6 Setting The Structure Cursor

Oplevel-Cursor-To-Mouse	uC Mouse-Left
Oplevel-Mouse-Extend	uC Mouse-Middle
Oplevel-Select	uC Backspace
Mouse-Select-Lexeme	Cursor-To-Mouse at level “Lexeme”
Mouse-Select-Olevel-Node	Cursor-To-Mouse at level “Any Structure Level”
Mouse-Extend-Lexeme	Mouse-Extend at level “Lexeme”
Mouse-Extend-Olevel-Node	Mouse-Extend at level “Any Structure Level”
Select-Lexeme	Select at level “Lexeme”
Select-Olevel-Node	Select at level “Any Structure Level”

Selection during language-oriented editing relies on the notion of operand levels (section 2.6). In a language-oriented view, the set of operand levels is much richer than in the text world. The operand levels for text editing are included, as well as levels corresponding to basic abstractions in the language being edited (see Section 6.4).

As with text, the command Oplevel-Cursor-To-Mouse, bound to UC Mouse-Left, selects the operand beneath the mouse cursor. The structure node which will be designated by the structure cursor and the region of text actually chosen are determined by the operand level. Thus if the mouse cursor is over the character “*” in Figure 4 on page 39 when the left button is clicked, the selected object might be the underlying lexeme “*”, expression “outerI * valueI”, or statement “countI := outerI * valueI + resultI” 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.

To select arbitrary portions of a structure, use the “Character” level commands and operate on the textual representation.

6.7 Structural Navigation

Oplevel-Cursor-Forward	UC UF
Structural navigation of a program allows you to move the structure cursor from one node of a
given type to another where type is defined in the language description by operand levels. Simple
navigation involves the “Oplevel-” operations Oplevel-Cursor-Forward and Oplevel-Cursor-Backward.
They perform preorder and reverse-preorder tree walking operations relative to the current operand
level. For example, if the operand level setting is “Expression”, the command Oplevel-Cursor-Forward
moves the structure cursor to the “next” expression node. Oplevel-In and Oplevel-Out move the
structure cursor to the nearest enclosed or enclosing node, again relative to the current operand
level. Oplevel-Cursor-To-First and Oplevel-Cursor-To-Last move the structure cursor to the “first”
and “last” instances of the current operand level respectively. Oplevel-Cursor-To-Mouse moves the
structure cursor to the instance of the operand level which is closest to the position of the mouse.

All of the operand level navigation commands are available from the operand level menu as
well. This menu can be invoked by the command Oplevel-Menu which is normally bound to the key
sequence “C Mouse-Right”.

6.8 Language Errors

The operand levels “Syntax Error” and “Unsatisfied Constraint” are preset by default in most lan-
guage descriptions and are of special significance: they are used to locate errors in a structured
document and to move the structure cursor from one error node to another. The standard set of
“Oplevel-” commands are bound at the “Syntax Error” and “Unsatisfied Constraint” levels to allow
quick and easy navigation between errors. The presence of errors in a program in no way interferes
with normal text editing. In Pan, the identification and diagnoses of language errors is just another
kind of derived information; as much service as possible will be supplied in their presence.
Some language descriptions may combine these two types of language errors into a single operand level or may split different types of syntax errors or different unsatisfied constraints into separate levels. The two operand levels and their corresponding flags described here are used in most of the existing language descriptions.

After lexical analysis, provided that there are no lexical errors, the syntactic analyzer updates the internal structured representation.

The syntactic analyzer checks the structure (or syntax) of the program against the L&d analysis of the language. If the syntactic analyzer encounters syntactic errors, the number of errors discovered during the analysis is displayed on the annunciator line, and the Syntax-Errors-Present? flag is set. This flag appears as a "!" on the information panel. When the syntactic analyzer 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 syntactic analyzer to be displayed on the annunciator line. If the option Highlight-Syntax-Errors is set then the text associated with the syntax error node will be displayed in red ink. Text colors may be changed by setting the option Text-Window-Fg-Colormap (see Section 7.5).

The options that control highlighting and color text rendering for languages are described in Figure 5.

<table>
<thead>
<tr>
<th>Option</th>
<th>Configured By</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>Highlight-Syntax-Errors</td>
<td>Ink (foreground color) 1</td>
<td>Red</td>
</tr>
<tr>
<td>Highlight-Query-Results</td>
<td>Ink (foreground color) 2</td>
<td>Blue</td>
</tr>
<tr>
<td>Highlight-Semantic-Errors</td>
<td>Shading (background color) 1</td>
<td>Pale Red</td>
</tr>
<tr>
<td>Highlight-Structure-Cursor</td>
<td>Shading (background color) 2</td>
<td>Pale Blue</td>
</tr>
</tbody>
</table>

Figure 5: The options that control structural highlighting

Figure 3 on page 35 shows the display of a syntax error as well as a graphical view displaying the internal tree structure for the error.

The semantic analyzer checks that the program satisfies a set of contextual constraints, for example type restrictions (based on the underlying language definition) and naming conventions (based on intended usage). If the semantic analyzer detects any semantic errors, the number of errors discovered during the analysis is displayed on the annunciator line, and the Semantic-Errors-Present? flag is set. This flag appears as a "#" on the information panel. The semantic errors or unsatisfied constraints are represented as annotations on the internal tree. If the option Highlight-Semantic-Errors is set then the text associated with the semantic error node will be highlighted in red ink. The highlight colors may be set to any set of colors by setting the option Text-Window-Bg-Colormap in the appropriate view (see Section 7.5).

6.9 Editing

Editing structured views is quite simple in Pan. Actual editing is implemented by text operations, but navigation and selection may be optionally driven by language-based information.
All insertions use the text-level commands used for text editing\footnote{The language description SIMPLE2 includes the a rudimentary prototype for the template-expansion style of syntax directed editing. These operations are also implemented textually, but the fact is somewhat hidden from the user since a newly expanded template is always immediately incrementally analyzed, and thereby promoted into structure}. Deletions are accomplished using either text-level commands, or using the Oplevel-Delete command. Oplevel-Delete will delete the region of text under the structure cursor at the current operand level. Other editing, using the clipboard and the kill ring, function the same as for text editing.

Undoing edit actions restores the text without restoring the structured representation. Thus undoing currently requires reanalysis.

6.10 Searching in Programs

<table>
<thead>
<tr>
<th>Lexeme-Search-Forward</th>
<th>Cursor-Search-Forward at level “Lexeme”</th>
</tr>
</thead>
<tbody>
<tr>
<td>Replace-Lexemes</td>
<td>Replace-All at level “Lexeme”</td>
</tr>
<tr>
<td>Query-Replace-Lexemes</td>
<td>Query-Replace-All at level “Lexeme”</td>
</tr>
</tbody>
</table>

In addition to ordinary textual search, which is always available along with all textual services, regular expression searching may be applied to structural components and particular operand levels. At the moment this is only implemented at the lexical level.

The search commands apply the match pattern to each lexeme except for comments (which are not regarded as lexemes). If a prefix argument is not present then the entire lexeme must match the pattern. Otherwise, the pattern may match a subset of the lexeme. If the option Case-Sensitive-Lexeme-Search is set then the match must be case sensitive.

6.11 Queries

The effect of a query is to mark a set of nodes in the internal tree which have some specified property. For example, in SIMPLE2 a variable may be queried. When a variable has been queried, the text associated with the tree nodes representing the declaration and all of the uses of that variable is displayed in blue. (Text colors may be changed by setting the option Text-Window-Fg-Colormap. See Section 7.5) You may then set the operand level to “Query” and navigate among these uses of the variable with the normal operand level commands.

The current prototype implementation of queries insists that object protection be turned on while displaying a query result. This means that you must manually clear a query (by toggling the object protection) before you make further modifications. Work is underway to remove this restriction.

6.12 Debugging Features

These features of Pan are not generally useful for programming in Pan, but are very useful to designers of Pan language descriptions.
Graphical Views

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Visit-Program-Graph-View</td>
<td>View⇒Program Graph: With Node Names</td>
</tr>
<tr>
<td>Visit-Program-Subgraph-View</td>
<td>View⇒Subgraph At Cursor: With Node Names</td>
</tr>
<tr>
<td>Visit-Program-Graph-Operator-View</td>
<td>View⇒Program Graph: With Operators</td>
</tr>
<tr>
<td>Visit-Program-Subgraph-Operator-View</td>
<td>View⇒Subgraph At Cursor: With Operators</td>
</tr>
<tr>
<td>Visit-Program-Graph-Address-View</td>
<td>Not bound</td>
</tr>
<tr>
<td>Visit-Program-Subgraph-Address-View</td>
<td>Not bound</td>
</tr>
<tr>
<td>Visit-Program-Graph-Context-View</td>
<td>Not bound</td>
</tr>
<tr>
<td>Visit-Program-Subgraph-Context-View</td>
<td>Not bound</td>
</tr>
<tr>
<td>Visit-Program-Graph-Parse-Change-View</td>
<td>Not bound</td>
</tr>
<tr>
<td>Visit-Program-Subgraph-Parse-Change-View</td>
<td>Not bound</td>
</tr>
<tr>
<td>Visit-Program-Graph-Parse-Status-View</td>
<td>Not bound</td>
</tr>
<tr>
<td>Visit-Program-Subgraph-Parse-Status-View</td>
<td>Not bound</td>
</tr>
</tbody>
</table>

The Visit-Program-Graph- commands all bring up a graphical representation of the internal tree representation for the program. The Visit-Program-Subgraph- commands generate a representation for the subgraph of the internal subtree which is rooted at the currently selected node of the graph. The tree is laid out horizontally, with the root at the left and at every node the first child above the second child which is above the third child, etc.

A graphical view is redrawn after every reanalysis of the structure. Subgraph views are removed if a reanalysis ever causes the associated node to be deleted.

The views created by the different variations of these commands are graphically similar: they differ in the way that a textual label is created for each node. In the simplest version, Visit-Program-Graph-View, each node is simply named, but other versions add extra information useful for various kinds of debugging.

The colors and line thicknesses of the graph are configurable options.

Other Language-Oriented Views

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Visit-Database-View</td>
<td>Not bound</td>
</tr>
<tr>
<td>Visit-Node-Semantics-View</td>
<td>Not bound</td>
</tr>
<tr>
<td>Visit-Lexeme-Internal-Data-View</td>
<td>Not bound</td>
</tr>
<tr>
<td>Visit-Worklist-View</td>
<td>Not bound</td>
</tr>
<tr>
<td>Visit-Tree-Internal-Data-View</td>
<td>Not bound</td>
</tr>
<tr>
<td>Visit-Subtree-Internal-Data-View</td>
<td>Not bound</td>
</tr>
</tbody>
</table>

These textual views are primarily useful for debugging Ladle and Colander language descriptions. They are not maintained incrementally during text editing. Instead, the first textual modification after analysis causes the view to be marked dirty. As long as the information in the view is inconsistent with the text of the program, it will be displayed in gray. It is then redrawn the next time there is an analysis to reflect the new information.

These views display the internal data representation for the structure of a program. Visit-Lexeme-Internal-Data-View creates a view which displays a description of the lexical stream inter-
nal representation. Visit-Tree-Internal-Data-View and Visit-Subtree-Internal-Data-View both create a view which continually displays a textual description of the internal tree representation.

These views display information derived from the semantic analyzer. Visit-Database-View creates a view which contains the contents of the semantic database. Visit-Node-Semantics-View creates a view which continually displays the contents of the semantic database for the tree node of the tree cursor at the time the command is executed. Visit-Worklist-View creates a view which displays the semantic worklist (currently unsatisfied constraints).

Exploring the Tree Structure

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tree-Up</td>
<td>Not bound</td>
</tr>
<tr>
<td>Tree-Down</td>
<td>unbound</td>
</tr>
<tr>
<td>Tree-Left</td>
<td>Not bound</td>
</tr>
<tr>
<td>Tree-Right</td>
<td>Not bound</td>
</tr>
</tbody>
</table>

This set of navigation commands for textual language-based views is used for debugging. It includes Tree-Up, Tree-Down, Tree-Left, and Tree-Right. These are the usual tree-oriented commands. They function in the text view of the program. From the current node, Tree-Up moves the structure cursor to the parent, Tree-Down moves it to the leftmost child of the current node. Tree-Left and Tree-Right move the structure 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.

7 Simple Customization

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

7.1 Start-Up Processing

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Load-File</td>
<td>X L</td>
</tr>
<tr>
<td>Load-From-Text-Stream</td>
<td>Esc X</td>
</tr>
<tr>
<td>Load-From-Selection</td>
<td>Esc S</td>
</tr>
</tbody>
</table>

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

---

*One could make the same kind of navigation possible by adding the oplevel "Node" to those available on a view and then using the Oplevel-Cursor-Forward, Oplevel-Cursor-Backward, Oplevel-Cursor-In, and Oplevel-Cursor-Out commands.*
Pan loads the ".panrc" file using the command Load-File. All commands that load files use the search path specified by the option Editor-File-Search-List. The default value of the option Editor-File-Search-List is set to

"." "=" "/usr/local/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 described in the next section.

(Load-File "filename")

While running Pan you can explicitly load commands from any file with the Load-File command which prompts for a file name or from any view in Pan with the Load-From-Text-Stream or Load-From-Selection commands.

Language-Based Configuration

Display-Language-List
Auto-Load file-name regular-expression
Auto-Exec function-name regular-expression
Display-Auto-Load
Display-Auto-Exec

Help⇒Pan: Language Descriptions Loaded
Help⇒Pan: Filename Auto-Load Map
Help⇒Pan: Filename Auto-Exec Map

The basic textual views on programs all use the standard text-edit-view-style, so language-specific configuration must be done at the view-instance level (the :view scope). Typically, this is achieved by using the Auto-Exec mechanism. When a file in a given language is opened for editing, an auto-exec function named <language>-mode by default. is executed. These functions and all supporting code are in modules named "<language>-mode.cl" by default. Loading one of these modules loads all aspects of the language description needed: Ladle description. Colander description. editing interface configuration and support. The Pan Engineering Manual[5] contains more information about configuration and language-based information in particular.

These modules may be preloaded in a dumped Pan, or they will be auto-loaded when needed. To see a list of all of the languages for which a Ladle syntactic description or a Colander semantic description have been loaded. use the command Display-Language-List.

The commands Display-Auto-Load and Display-Auto-Exec display the list of Auto-Load and Auto-Exec commands which have been defined. The Auto-Load command instructs Pan to ensure that a file has been loaded whenever a file whose name matches a given UNIX file expression is created. If the Auto-Load is set up properly. then the language mode for a file should be loaded when a view is created for the file. The file is loaded at most once as a result of Auto-Load.

For instance, (Auto-Load "c-mode" "*.[hc]") tells the system to load the file "c-mode" the first time that a file whose name matches "*.[hc]" is edited. The file "c-mode" can be either lisp code or compiled lisp code; if both "c-mode.cl" and a "c-mode.fas1" are found in the same
directory, the most recently modified version is chosen.

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

A **minor mode** is a collection of commands and bindings useful while editing views of a given type. For instance these commands can be used to create a minor mode for \TeX input:

\begin{verbatim}
(Auto-Load "tex-mode" "*.tex")
(Auto-Exec 'tex-mode "*.tex")
\end{verbatim}

The parameterless function tex-mode is defined in the library file “tex-mode.cl” to set up a specialized collection of bindings. The file “tex-mode.cl” also defines a number of commands useful for manipulating \TeXnical text.

### 7.2 Using Scopes

- **With-View-Style-Scope** view-style (commands)
- **With-View-Scope** view (commands)
- **With-Buffer-Scope** buffer (commands)

To set options or create bindings within a given scope (other than :global), the scope must be specified before the command to set the option or create the binding is executed. The default scope for most bindings is :global.

Most commonly, the :view-style scope is used for setting non-global defaults. For example, to set up the normal key binding for "f" in the directory editor view-style, dired-view-style, you could use this lisp code:

\begin{verbatim}
(With-View-Style-Scope (view-style dired-view-style)
 (Bind-Key 'Dired-Visit "f" :view-style))
\end{verbatim}

The other two "-Scope" macros are not often used. Usually, if you want to make a change within a view, you would make the change from within that view or from within an Auto-Exec function, and then you don’t need to mention the scope explicitly.

### 7.3 Packaging

Pan’s implementation uses the Common Lisp package system, so every symbol (in particular every command, function, option, etc.) is defined in some package; almost nothing is defined in the default user package.

For convenience, all of Pan’s commands, functions, options, etc. are imported automatically into the default user package. This means that configuration files (e.g. ".panrc"), which typically are left in the user package, need not contain package qualifiers for these names (the qualifiers are optional).
7 SIMPLE CUSTOMIZATION

The help system by default does not display the package qualifiers for these names: option
Package-Symbol-Names requests that the qualifiers be displayed.

7.4 Creating Bindings

Pan provides several configuration mechanisms in the form of bindings. Keystrokes, menu items,
and generic commands may all be bound to Pan commands. All bindings are scoped, with local
bindings taking precedence over global bindings. This section describes how to find out what your
current bindings are and how to set up new default bindings with your "panrc" file.

When you shadow a binding for an object in a given scope, it effectively removes any binding
for that object inherited from an enclosing scope if any such binding exists.

It is possible to invoke a command without binding it by using the commands Execute-Lisp-Line
or Execute-Command (Section 7.8 provides more details.).

Creating Key Bindings

Display-All-Key-Bindings

Bind-Key command key-sequence [scope]

Help⇒This View⇒Keys: Display All

Key bindings associate keystroke sequences with commands. The command Display-All-Key-Bindings
prints the key bindings for the current view into the help view. Key bindings are established or
altered using the command Bind-Key. The default scope of the change is :global: specifying :view
or :view-style within the proper scope makes the change in the local environment. The syntax
for Bind-Key is

(Bind-Key 'function "key sequence" [scope])

where 'function is a quoted function name. "key sequence" is a COMMON LISP string specifying a
keystroke sequence, and scope is either :view, :view-style, or :global. The default value for
scope is :global.

Pan allows you to bind any key sequence to any command. This includes special keys, function
keys, mouse buttons, and keys modified by shift or control. Figure 6 shows some of the special
keyboard keys (and pseudo-keys, which is how mouse button events are handled in Pan) that are
known to Pan along with their names and default bindings. Most of the names for special keys
are derived from and correspond to X11 "keysyms." Not all of the keys are generated by every
keyboard. The default bindings of the "L" function keys are inspired by their labeling on Sun
keyboards.

When specifying a keystroke sequence, control characters such as "Control ?" are denoted by
the two-character sequence "~?"; the "Escape" prefix is denoted by "Esc". For example,

(Bind-Key 'Delete-Character "~D" :global)

establishes the default binding for Delete-Character.
### 7.4 Creating Bindings

<table>
<thead>
<tr>
<th>Key</th>
<th>Key Name</th>
<th>Binding</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot; &quot;</td>
<td>Space</td>
<td>Insert-Space-Fill</td>
</tr>
<tr>
<td>&quot;I&quot;</td>
<td>Tab</td>
<td>Self-Insert</td>
</tr>
<tr>
<td>&quot;J&quot;</td>
<td>Newline</td>
<td>Newline-And-Indent</td>
</tr>
<tr>
<td>&quot;M&quot;</td>
<td>Return</td>
<td>Insert-Newline-Fill</td>
</tr>
<tr>
<td>&quot;H&quot;</td>
<td>Backspace</td>
<td>Delete-Previous-Character</td>
</tr>
<tr>
<td>&quot;Escape&quot;</td>
<td></td>
<td>Not bound</td>
</tr>
<tr>
<td>&quot;W&quot;</td>
<td>Esc</td>
<td>Cursor-To-Mouse</td>
</tr>
<tr>
<td>&quot;X&quot;</td>
<td>Mouse-Left</td>
<td>Select-Region-Dot-To-Mouse</td>
</tr>
<tr>
<td>&quot;Y&quot;</td>
<td>Mouse-Middle</td>
<td>Execute-Menu</td>
</tr>
<tr>
<td>&quot;Z&quot;</td>
<td>Mouse-Right</td>
<td>Oplevel-Cursor-To-Mouse</td>
</tr>
<tr>
<td>&quot;A&quot;</td>
<td>Control-Mouse-Left</td>
<td>Oplevel-Cursor-To-Mouse</td>
</tr>
<tr>
<td>&quot;B&quot;</td>
<td>Shift-Mouse-Left</td>
<td>Oplevel-Menu</td>
</tr>
<tr>
<td>&quot;C&quot;</td>
<td>Control-Mouse-Right</td>
<td>Oplevel-Menu</td>
</tr>
<tr>
<td>&quot;D&quot;</td>
<td>Shift-Mouse-Right</td>
<td>Oplevel-Menu</td>
</tr>
<tr>
<td>&quot;E&quot;</td>
<td>Shift-Mouse-Right</td>
<td>Oplevel-Menu</td>
</tr>
<tr>
<td>&quot;F&quot;</td>
<td>Left-Arrow</td>
<td>Oplevel-Cursor-Backward</td>
</tr>
<tr>
<td>&quot;G&quot;</td>
<td>Right-Arrow</td>
<td>Oplevel-Cursor-Forward</td>
</tr>
<tr>
<td>&quot;H&quot;</td>
<td>Up-Arrow</td>
<td>Oplevel-Out</td>
</tr>
<tr>
<td>&quot;J&quot;</td>
<td>Down-Arrow</td>
<td>Oplevel-In</td>
</tr>
<tr>
<td>&quot;K&quot;</td>
<td>Begin</td>
<td>Oplevel-Cursor-To-First</td>
</tr>
<tr>
<td>&quot;L&quot;</td>
<td>End</td>
<td>Oplevel-Cursor-To-Last</td>
</tr>
<tr>
<td>&quot;M&quot;</td>
<td>Prior</td>
<td>Oplevel-Cursor-Backward</td>
</tr>
<tr>
<td>&quot;N&quot;</td>
<td>Next</td>
<td>Oplevel-Cursor-Forward</td>
</tr>
<tr>
<td>&quot;O&quot;</td>
<td>Find</td>
<td>Oplevel-Cursor-Search-Forward</td>
</tr>
<tr>
<td>&quot;P&quot;</td>
<td>L1-L3</td>
<td>Not bound</td>
</tr>
<tr>
<td>&quot;Q&quot;</td>
<td>L4</td>
<td>Undo</td>
</tr>
<tr>
<td>&quot;R&quot;</td>
<td>L5</td>
<td>Not bound</td>
</tr>
<tr>
<td>&quot;S&quot;</td>
<td>L6</td>
<td>Copy-To-Clipboard</td>
</tr>
<tr>
<td>&quot;T&quot;</td>
<td>L7</td>
<td>Not bound</td>
</tr>
<tr>
<td>&quot;U&quot;</td>
<td>L8</td>
<td>Paste-From-Clipboard</td>
</tr>
<tr>
<td>&quot;V&quot;</td>
<td>L9</td>
<td>Not bound</td>
</tr>
<tr>
<td>&quot;W&quot;</td>
<td>L10</td>
<td>Oplevel-Cursor-Search-Forward</td>
</tr>
<tr>
<td>&quot;X&quot;</td>
<td>F1-F10</td>
<td>Cut-To-Clipboard</td>
</tr>
<tr>
<td>&quot;Y&quot;</td>
<td>R1-R15</td>
<td>Not bound</td>
</tr>
<tr>
<td>&quot;Z&quot;</td>
<td>Home</td>
<td>Not bound</td>
</tr>
<tr>
<td>&quot;[&quot;</td>
<td>Help</td>
<td>Not bound</td>
</tr>
<tr>
<td>&quot;\n</td>
<td>Insert</td>
<td>Not bound</td>
</tr>
<tr>
<td>&quot;/&quot;</td>
<td>Menu</td>
<td>Not bound</td>
</tr>
<tr>
<td>&quot;#&quot;</td>
<td>Cancel</td>
<td>Not bound</td>
</tr>
<tr>
<td>&quot;=&quot;</td>
<td>Break</td>
<td>Not bound</td>
</tr>
</tbody>
</table>

Figure 6: Bindings and names for special keys
Creating Menu Bindings

Display-Default-Menu
Bind-Menu menu-name bindings
Rebind-Menu menu-name old-label new-item [new-label]
Create-Menu menu-being-added title bindings
Copy-Menu old-name new-name [new-title]

Help⇒ This View: Default Menu

A menu binding associates a menu item with a command or another menu. The command Display-Default-Menu prints the menu bindings for the current view in the help view. Menu bindings are created via the macros Bind-Menu, Copy-Menu, Create-Menu, and Rebind-Menu. All menus are global. Any change to a menu affects the global description of that menu. To establish a binding for a menu item in an existing menu, use

(Bind-Menu menu-name bindings)

where bindings is a list of menu items of the form

(item ["Selection Name"] [position])

The item being bound to the menu may be either a command or another menu. The "Selection Name" must be a COMMON LISP string. If "Selection Name" is omitted, the name of the command or menu is used. The optional argument position is an integer - the position in the menu at which to insert the item. The first item in the menu has position 1. For example,

(Bind-Menu help-view-style-menu
 (Document-Symbol "Describe" 1))

sets up the default binding for Document-Symbol as the first item in the Help menu and

(Bind-Menu text-edit-view-style-menu
 (global-help-menu "Help"))

sets up the default binding for the Help sub-menu in the Text menu.

The macro Create-Menu creates a new menu with the given name, title, and bindings. The format for Create-Menu is:

(Create-Menu menu-name menu-title bindings)

For example, the default global Help menu is created by the following COMMON LISP code:

---

9 A menu item may also be a lisp form.
7.4 Creating Bindings

(Create-Menu global-help-menu "Help"
  (this-view-help-menu "This View")
  (help:Apropos-String "Apropos")
  editor-help-menu)

Rebind-Menu allows you to change an existing menu item. Its format is:

(Rebind-Menu menu-name old-label new-label)

For example, you could change the menu item This View: Options to print the option documentation with the command Display-Option-Documentation by the following code:

(Rebind-Menu this-view-help-menu
 "Options" Display-Option-Documentation "Option Docs.")

Creating Operand Level Bindings

Display-Operand-Level-Bindings
Display-Operand-Command-Bindings
Display-Operand-Level-Menus
Display-Operand-Level-Documentation
Display-Operand-Command-Documentation
Oplevel-Menu
Execute-Oplevel-Menu
Bind-Oplevel-Command-List operand-command-command-pair-list level [scope]
Bind-Oplevel-Menu level menu-name [scope]

Operand level command bindings associate an operand level and a generic command with a Pan command. Operand level menu bindings associate an operand level with a menu. The commands Display-Operand-Level-Bindings, Display-Operand-Command-Bindings, and Display-Operand-Level-Menus print into the help view the appropriate bindings in the current view. Display-Operand-Level-Documentation and Display-Operand-Command-Documentation display documentation about the operand levels and generic “Oplevel-” commands respectively.

Bind-Oplevel-Command-List allows you to create a set of operand level command bindings at a given operand level. Its format is:
(Bind-Oplevel-Command-List
   ((operand-command1 command1)
    (operand-command2 command2)
    .
    .
    (operand-commandN commandN))
level [scope])

For example, the standard cursor-motion bindings for the operand level "Word" are established by the following code:

(Bind-Oplevel-Command-List
   ((Oplevel-Cursor-Forward text:Next-Word)
    (Oplevel-Cursor-Backward text:Previous-Word)
    (Oplevel-Cursor-To-First text:Move-To-First-Word)
    (Oplevel-Cursor-To-Last text:Move-To-Last-Word)
    (Oplevel-Cursor-To-Mouse text:Mouse-Select-Fullword))
word-level)

The operand level menu contains, by default, all of the commands which are bound at that level. Operand level menus may be created and altered in exactly the same fashion as other menus. The function Bind-Oplevel-Menu binds a menu to an operand level. Its format is:

(Bind-Oplevel-Menu level menu-name [scope])

For example, this code binds the standard word-oplevel-menu to the operand level "Word" in the global scope:

(opl:Bind-Oplevel-Menu 'word-level 'word-oplevel-menu)

The command Oplevel-Menu is a bindable generic command. It is usually bound to the command Execute-Oplevel-Menu which executes the menu that is bound to the current operand level. It is possible to rebind Oplevel-Menu to a different command for a given operand level; this allows for a highly customizable environment, used at present only for experimental features in SIMPLE2.

7.5 Getting and Setting Options

<table>
<thead>
<tr>
<th>Announcement</th>
<th>Not bound</th>
</tr>
</thead>
<tbody>
<tr>
<td>Announce-Option-Variable</td>
<td>Help⇒ This View⇒ Set Option: Global</td>
</tr>
<tr>
<td>Set-Global-Option-Variable</td>
<td>Help⇒ This View⇒ Set Option: View Style</td>
</tr>
<tr>
<td>Set-View-Style-Option-Variable</td>
<td>Help⇒ This View⇒ Set Option: View Only</td>
</tr>
<tr>
<td>Set-View-Option-Variable</td>
<td>Not bound</td>
</tr>
<tr>
<td>Set-Buffer-Option-Variable</td>
<td>Help⇒ This View: Options</td>
</tr>
<tr>
<td>Display-Option-Variables</td>
<td></td>
</tr>
</tbody>
</table>
7.5 Getting and Setting Options

Display-Option-Documentation

Much of Pan's behavior may be modified by setting options. The command Announce-Option-Variable is used to retrieve the value of an option in the current view. The command prompts for the name of the option and then announces its value in the message line.

The Set-commands listed above allow you to set the value of an option in the given scope while running Pan. They all prompt for the name of the option and then for its new value, which you must specify in a form acceptable to the Common Lisp reader.

To set an option's default value to value, use the lisp form in your ".panrc" file

\[\text{(setf (variable option-name scope) value)}\]

In this case, scope must specify either :view-style or :global. For instance, the expression

\[\text{(setf (variable Window-Close-With-Icon :global) nil)}\]

sets the value of the option Window-Close-With-Icon to be nil. To set any option in a particular view-style, use the macro With-View-Style-Scope to specify the view-style (see below). A list of all of the view styles is in Appendix G.

Setting Font Maps

Since the Text-Window-Fontmap option is a scoped variable, the macro With-View-Style-Scope must be used to set the font for a particular view-style as described in Section 7.2. For example, the help view uses the view-style help-view-style; thus a new font map for the help view could be specified by the following code:

\[\text{(With-View-Style-Scope (view-style help-view-style)}\]
\[\text{(setf (variable Text-Window-Fontmap :view-style)}\]
\[\text{'"-adobe-times-medium-i---12-***********"}\
\text{"-adobe-new century schoolbook-medium-r---10-***********"}\
\text{"-adobe-times-bold-r---12-***********")}\]

The X tool xfontsel is useful for determining the exact font you are looking for and getting its full name.

Setting Color Maps

Text-Window-Fg-Colormap
Text-Window-Bg-Colormap
Panel-Bg-Color
Panel-Flag-Default-Fg-Color
Panel-Flags-Bg-Color
Panel-Logo-Fg-Color
Panel-Logo-Bg-Color
Colors are represented in Pan as dotted pairs of strings which represent color names, such as ("red"."black"). The first color will be used on a color monitor if the option Window-Use-Color is set, otherwise the second color is used. Figure 5 on page 42 shows the uses and default values for the text colormaps in structure-based documents.

Each of the “Panel-” options has the value of a single pair of colors one of which is used for the appropriate area of the panel. You can set a color option by any of the methods you use to set other options. For example, this code would change the Panel-Flag-Default-Fg-Color to blue on color screens:

```
(setf (variable Panel-Flag-Default-Fg-Color :global) '("blue"."black")
```

The Text-Window-Fg-Colormap is used for displaying colored or greyed text in textual views. The Text-Window-Bg-Colormap is used for displaying colored or greyed background highlight for text in textual views. Each of these two options is an indexed list of color pairs. When a region of text is highlighted or colored in Pan, it is assigned an index into the appropriate colormap and displayed with that color.

The X tool XCOL will give you a display of the colors that are available on your system.

### 7.6 Character Sets

Display-Char-Sets  
Display-Char-Set-Documentation  

Set-Char-Set char-set-name char-list [scope] [operation]  
Set-Char-Match left-char right-char [scope] [operation]

These are the normal help commands. Display-Char-Sets lists all of the character sets and their contents in the current view. Display-Char-Set-Documentation lists the character sets with a description and their default global values.

One particular character set, match-characters, contains all of the matched characters such as right and left braces. Whenever Pan looks for a balanced expression it tries to find the match for the character under the cursor.

The functions Set-Char-Set and Set-Char-Match are useful for creating character sets and altering the contents of existing character sets. Set-Char-Set allows you to add, delete, or shadow characters in a character set. Its format is:

```
(Set-Char-Set char-set-name char-list [scope] [operation])
```

The char-list has the form of a Common Lisp string of characters. The operation may be one of :add, :delete, or :shadow: it defaults to :add. For example, here is the lisp code to add the character “-” to the character set word-characters:
7.7 Creating and Modifying Flags

(Set-Char-Set 'word-characters "-" )

Set-Char-Match allows you to add, delete, or shadow pairs of match characters to the match-character set. Its format is:

(Set-Char-Match left-char right-char [scope] [operation])

For example, here is code to add the characters ">" and "<" to the match-characters set:

(Set-Char-Match #\< #\> )

7.7 Creating and Modifying Flags

Display-Flags

Window-Modify-Flag-Collection

A flag in Pan is implemented as an option which can be true or false (in Common Lisp, t or nil) and an associated bitmap. The command Display-Flags displays all of the flags defined in the current view.

Flags may be added to or removed from any scope with the function Window-Modify-Flag-Collection. Its format is:

(Window-Modify-Flag-Collection [scope]
  [:add 'add-flag-list]
  [:delete 'delete-flag-list]
  [:shadow 'shadow-flag-list])

where delete-flag-list is a list of variables in the current collection of flags and add-flag-list is a list of flag descriptors of the form:

(variable
  [(on-bitmap [ [ on-background-color ] on-foreground-color ])
   (off-bitmap [ [ off-background-color ] off-foreground-color ])])

The bitmap variables both default to Panel-Flag-Default-Bitmap, the on-foreground-color defaults to Panel-Flag-Default-Fg-Color, and the background color variables both default to Panel-Flags-Bg-Color. The off-foreground-color defaults to Panel-Flag-Default-Fg-Color if the off-bitmap is set and Panel-Flags-Bg-Color if it is not.

All bitmaps and colors used in the descriptor must be defined as variables. For example, to create a new bitmap for the syntax error flag and assign colors and bitmaps for its on and off representations, variables representing all of these pieces of information would have to be defined and set in your "panrc" file:
(defparameter *syntax-errors-present-bitmap* "~/syntax-error.12x12")

(define-option-variable *error-on-color* ("red" "black")
  "The color to use for the error flag."
  :bindable :scopes :global)

(define-variable-cache *error-on-color* 
  'win-system-color-compute-cache)

(define-option-variable *error-off-color* ("medium-seagreen" "black")
  "The color to use for the no-error flag."
  :bindable :scopes :global)

(define-variable-cache *error-off-color* 
  'win-system-color-compute-cache)

(define-option-variable no-error-bitmap "~/no-error.12x12"
  "The bitmap which represents the no-error flag."
  :bindable :scopes :global)

(define-variable-cache no-error-bitmap 
  'win-system-bitmap-compute-cache)

See Section 7.8 for more information on defining new options. Then the following code would be included in the appropriate auto-load files:

(defun win-window-modify-flag-collection :view
  :add
  '((o-tree:syntax-errors-present?
      (syntax-errors-present-bitmap error-on-color)
      (no-error-bitmap error-off-color)))))

The X tool-bitmap is useful for designing flag bitmaps.

7.8 Lisp-Oriented Commands

| Execute-Lisp | Esc Esc |
| Execute-Command | Esc x |
| Load-File | ^X ^L |
| Load-From-Selection | Esc ^S |
| Load-From-Text-Stream | Esc ^X |
These commands are mostly 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.

Execute-Lisp prompts for a Common LISP form to evaluate: the resulting value 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-Command and respond to the prompt with the name of the command, e.g. Next-Character. Alternatively, one can invoke Execute-Lisp and respond to the prompt with the expression (Next-Character). The parentheses are required in the latter case, since commands are implemented as Common LISP functions.

Load-File loads a file of Common LISP and Pan commands into the system. Load-From-Text-Stream loads the contents of the entire text stream in the current view and evaluates the Common LISP and Pan commands. Load-From-Selection loads and evaluates the Common LISP and Pan commands in the current text selection.

Defining Variables and Options

Define-Option-Variable option [initial-value [documentation [:apropos] [:notifier]]]
Define-Variable variable [initial-value [documentation [:apropos] [:option] [:notifier]]]

You can create variables and options simply by defining them as Common LISP variables in your "panrc" file. The macro Define-Variable allows you to include documentation and apropos information, add notifiers, or make your variable into an option variable known to the help system. The macro Define-Option-Variable is the same as Define-Variable with :option set to t. As an example, here is Common LISP code to create the option variable Number-Of-Views-Created:

(Define-Option-Variable Number-Of-Views-Created 0
 "the number of views that have been created so far"
:notifier #'view-num-notifier
:bindable-scopes :global
:initial 'nil)

Hooks and Notifiers

Display-Hooks
Display-Hook-Documentation
Display-Notifiers

Define-Hook-Function hook scope function [arguments] body
Define-Hook hook hookable-scopes arguments [documentation] [:apropos]
Perform-Hook hook arguments

add-variable-notifier variable notifier

If you really want to dig into Pan's internals and change things significantly, you may be interested in using hooks and notifiers. This is for ambitious users only!
A notifier is a function which is called every time the value of a given variable is changed. Most Pan variables and options have notifiers. You can add a notifier to a variable with the function add-variable-notifier or you can list the notifier when you declare the variable. Here is COMMON LISP code for a notifier to display the value of the variable in your console window (note that this notifier was listed in the variable definition):

```
(defun view-num-notifier ( foo scope-kind bound new old )
  ( declare ( ignore scope-kind bound old ))
  ( format t ""A is "A."%" new ))
```

A hook is a place within a function where you can insert new code. The Define-Hook-Function macro allows you to add code at an existing hook. Here is an example of a function to increment the option Number-Of-Views-Created (defined above) attached to the hook %view-construction-hook% which is called immediately after a view is created:

```
(Define-Hook-Function bufview:%view-construction-hook%
 :global
 add-to-view-num
 ()
 ( incf ( variable Number-Of-Views-Created :global)))
```

8 Acknowledgments

Pan is the work of many people. Professor Susan Graham has directed the project since its inception. Robert Ballance was the original architect and continued with the project until completing his Ph.D. Michael Van De Vanter joined the project in its early stages and has been a principal member of the project through the present. Christina Black, Jacob Butcher, Bruce Forstall, Mark Hastings, Darrin Lane, Bill Maddox, Ethan Munson, Tim Wagner, and Robert Wahbe have all made substantial contributions. Finally, we wish to acknowledge the community of "editor people" with whom we have interacted over the past several years. Many of their best ideas appear somewhere within Pan.
References


9 Glossary

Active View The view that owns the active window.

Active Window The active window is the window in which the most recent keystroke or mouse action occurred, although the active window may change as the result of a command, for example Visit-File.

Apropos A Help command for browsing documentation gathered during definition of Pan's commands, flags, and options. For instance, (Apropos "Cursor") lists all of the commands dealing with the cursor.

Auto Load The Auto-Load command instructs Pan to ensure that a specified file has been loaded whenever a view is created onto a file whose name matches a specified UNIX file expression.

Auto Exec The command Auto-Exec instructs Pan to execute a specified function whenever creating a view having a name that matches a specified pattern.

Backup A copy of the file being visited that records the state of the file before any changes are made using Pan. You may visit the backup of a file with the command Visit-Backup. You may restore a view of a file to its backed-up state with the command Revert-Object-To-Backup.

Binding An association between an event (key-sequence, mouse action, menu selection) and a Pan command, used by Pan's command dispatcher to determine the appropriate response to each user event. Every binding is made at a scope: global, for a specified view style, or for a specific view instance.

Character Set A named set of ASCII characters, scoped as with all other Pan bindings. Character sets are used to define simple structural attributes of ordinary text: words, white-space, matching brackets, and the like.

Checkpoint A copy of a file being visited that records an intermediate, modified state of the file. Pan makes a checkpoint periodically as the text in a view is modified; each checkpoint replaces the previous one. You may visit separately the most recent checkpoint of a file with the command Visit-Checkpoint. You may restore a view of a file to its most recently checkpointed state with the command Revert-Object-To-Checkpoint.

Clipboard A ring, shared by all views, that contains one or more regions of text. The clipboard supports copy, cut, and paste among views, or between Pan and other processes.

Command An operation that may be invoked directly by users through bindings to keystroke sequences and menus. Pan commands are defined using Define-Command. They are implemented as and may also be called as Common LISP functions.
Current Operand Level  See Operand Level.

Cut  Remove a region of text from the active text stream and push it onto the clipboard.

Cycle  Make the current top item in a ring become the oldest item, moving all other values up—the second youngest becoming the new top item.

Default Font  The first element in the font list specified by option Text-Window-Fontmap. When any other slot in a map is unspecified, the default font is used.

Delete  Remove a region of text from the active text stream, keeping no copies other than possibly in undo history.

Deselect  Ensure that there is no selection in the active view, without affecting the view’s contents.

Directory Editor  A Pan view style for browsing and operating upon UNIX file system directories.

Dot  An anachronistic internal name for Pan’s text cursor. The function Dot returns the integer offset (in characters, zero-based) of the text cursor from the beginning of the file.

Edit Object  Whatever can be displayed and possibly manipulated by a Pan view, closely related to the Emacs notion of “buffer.” A Pan session instantiates an edit object for each text file, program or directory being viewed; each edit object comprises a copy of the data being edited, one or more views, and other information.

Edit Window  That area of a Pan window in which part of the object being edited is displayed to the user.

Expression  A balanced bracket expression is a string of characters enclosed within matched left and right braces, brackets, parentheses, or other matched characters, and which may contain nested pairs of balanced match characters. Character set declarations specify which characters are to be treated as matching characters.

Fill  Rearrange lines of text so that all are of approximately the same length, as controlled by option Text-Line-Length.

Flag  A Pan variable that has been configured specially in some scope so that its (boolean) value appears on the information panel of all windows in that scope. At the flag’s panel location will appear one specified graphical appearance when the variable’s (scoped) value is on (non-nil) and another specified graphical appearance when the variable’s value is off (nil).
Font Map  A zero-indexed list of font names, containing from 1 to 8 names, that specifies how font codes stored in Pan’s internal text representation are to be interpreted during display operations.

Frame  The outer surrounding edge of a window that responds to X window protocols.

Generic Operation  A Pan command, such as Cursor-Forward or Delete, whose behavior is determined dynamically by dispatch on the current operand level of the active window and by operand bindings.

Global Image  A small inset within a Pan graphical window that summarizes the entire graph being viewed, regardless of zoom and scroll state. It also marks specially that portion of the entire graph that is visible in the rest of the window. The global image can be resized by the user and can be toggled on or off.

Graphical View  A Pan view that displays information in terms of boxes (possibly containing text) and lines that connect them.

Help View  A special Pan view used for displaying help information.

Hook  A place within Pan’s thread of control where new code can be inserted conveniently. The Define-Hook creates a hook, Define-Hook-Function adds code to the hook, and Perform-Hook executes the hook’s code.

Implicit Selection  The text region between the text cursor and the top mark of the mark stack. This is distinct from and secondary to the visible selection in each view.

Information Panel  The upper portion of every Pan window, which contains information about the current view, such as the language, the operand level, and displayed flags.

Internal Tree  Pan’s internal representation for programs and other language-structured objects. It is generated by the syntactic analyzer (parser) from the lexical stream. The details are generally hidden from users and are revealed only through the operand level mechanisms.

Key Binding  An association between a sequence of keyboard (or mouse button) events and a Pan command, used by Pan’s command dispatcher to determine the appropriate response to each user event. In some cases (prefix keys) the response may be to wait for another keyboard event. Every key binding is made at a scope: global, for a specified view style, or for a specific view instance.

Kill  Remove a region of text from the active text stream and push it onto the active view’s kill ring (or to the clipboard, depending on option Kills-To-Clipboard).
**Kill Ring** A ring, local to each view, that contains one or more regions of text. The kill ring supports copy, cut, and paste within a single textual view.

**Load** Read into Pan's execution image a collection of Common Lisp and Pan commands, executing top level forms.

**Lexeme** The smallest unit known to the syntactic analyzer. Lexemes include variable names, arithmetic and assignment operators, and keywords.

**Lexical Analysis** The process of reading a text stream and constructing an appropriate stream of lexemes according to an underlying language description. Lexical analysis in Pan is incremental, only making changes to the lexical stream where needed because of changes to the text stream since the previous analysis.

**Mark** A character position in a text stream that is guaranteed to remain in the stream, even when an underlying character is deleted (in which case the mark is moved rightward). A mark is attached to a character, not an offset, so deletions before a mark do not affect the character to which a mark refers.

**Mark Stack** Each Pan text stream provides for a stack of marks, the top element of which is known as simply "the mark". These values may be popped and pushed as in any stack.

**Menu Binding** An association between a menu or submenu title and a Pan command, used by Pan's command dispatcher to determine the appropriate response to each menu operation by the user. Every key binding is made at a scope: global, for a specified view style, or for a specific view instance.

**Message Line** The lowermost part of the information panel in each Pan window, used to display a single line of text containing various information: incomplete key sequences, information about time consuming operations being performed, and the output of some commands.

**Minor Mode** A collection of bindings that may be added to particular view instances, providing a slight variation on one of Pan's established view styles.

**Mouse Icon** A graphical token that shows the location of the mouse on the workstation screen. The mouse icon is normally an arrow when the mouse is positioned within a Pan window, as specified by option Window-Default-Mouse-Icon. Different mouse icons appear when Pan is busy working (not responding to user events) or to suggest a different function for the mouse in a region.

**Node** A structural component of Pan's internal tree representation, used to represent a meaningful part of a program or other structured document.
**Notifier** A function that may be associated with a Pan variable, using either the Lisp: notifier keyword with Define-Variable and Define-Option-Variable or the function Lispadd-variable-notifier. Whenever the value of a Pan variable in a particular scope changes, all associated notifiers are called with information about the old and new values.

**Numeric Prefix** An optional (default 1) integer argument to each Pan command invocation, supplied by special sequences of prefix keyboard events. Not all commands use the numeric prefix, and some check only for its existence and not its value.

**Operand Level** An element of Pan window state, always visible in the panel, which modulates the behavior of certain generic (or level-sensitive) operations like Cursor-Forward or Delete. The current level is selected by the user, from among those specified by option Operand-Level-Choices, using either the panel menu or by invoking level-specific commands such as Set-Oplevel-To-Character and Set-Oplevel-To-Declaration.

**Operand Level Binding** A three-way association of an operand level (for example Character at level "") , a generic (or level-sensitive) operation (for example Cursor-Forward) , and a Pan command (for example Next-Character). These bindings are used by Pan's command dispatcher to determine the appropriate response to each user invocation of a generic command. Every binding is made at a scope: global, for a specified view style, or for a specific view instance.

**Operand Level Menu** A menu associated with a particular operand level; the command Oplevel-Menu by default presents the user with the menu appropriate to the current level. These menus normally display all of the operand level bindings for the generic commands at the given level.

**Option** A user-definable, user-modifiable, typed Pan variable. Every value is assigned at a scope: global, for a specified view style, or for a specific view instance. Many of the customizations available to a user are provide via predefined options. Option values may be made visible in Pan's windows: see flags. Options may be made "active values" by the addition of notifiers.

**Parsing** See syntactic analysis.

**Paste** Copy a region of text from the top of the clipboard into the current text stream.

**Pop-Up Window** A small window which appears on the screen to accept a textual argument to a command or to obtain a response to a question.

**Prefix Keys** Keyboard events that are not bound to a command but which form prefixes to sequences which are bound. Any key may be a prefix in Pan: the keys "Escape", "\-X", "\-C", and "\-Z" are used as prefix keys for in Pan's default key bindings.

**Protected** The contents of a file, directory, or other editable object which is protected may not be altered.
Region  A contiguous sequence of characters in a Pan text stream. Most text operations involve regions either as source, destination, or both.

Replace  1. A “Replace-” command searches for instances of a given pattern and replaces them with a given string. 2. A region of text may be replaced by another region of text from the kill ring or clipboard.

Ring  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.

Scope  The scope of an option or binding determines the set of Pan views in which the option or binding is effective. Normal configurations occur in one of three scopes: global, for a specified view style, or for a specific view instance.

Scroll Bars  The scroll bars are found on the borders of a Pan window and allow you to scroll the window with the mouse.

Scrolling  Scrolling a window moves the region of the view which is currently visible in the window vertically or horizontally.

Search  Pan provides commands which search for strings that match specified regular expressions.

Selection  A specially designated region in textual views, distinguished by underlining in all text windows. There is at most one selection per view.

Semantic Analysis  The process of analyzing a program and its associated database of derived information, according to the Colander portion of an underlying language description. Results of semantic analysis include facts asserted in the database and diagnosis of program components at which Colander constraints (typically including, but not limited to, the static semantics of the language) are not satisfied. Semantic analysis in Pan is incremental, only making changes to the database where needed because of changes to the internal tree since the previous analysis. The Database-Updated? is on in a view if and only if semantic analysis has been performed since the most recent textual or database change.

Shadow  When you shadow a binding in a given scope, it effectively removes any binding inherited from an enclosing scope if any such binding exists.

Structural Cursor  A reference to some component in the internal tree representation of a structured object, shared by all views on that object.
Syntactic Analysis  The process of reading a lexical stream and constructing an appropriate internal tree, according to an underlying language description (also known as parsing). Syntactic analysis in Pan is incremental, only making changes to the internal tree where needed because of changes to the lexical stream since the previous analysis. The flag Parse-Is-Current? is on in a view if and only if lexical and syntactic analysis have been performed since the most recent textual change.

Text Cursor  The location where alterations to a text stream can occur, displayed as an inverted box in all text windows. Characters are inserted or deleted at the character position to the left of the character designated by the cursor.

Undo  Return the state of an edit object to a state it was in before a recent modification.

Unsatisfied Constraints  Contextual constraints from the Colander description of the underlying language which are not met in the program.

View  A view is a format for viewing an object. Any object may have several views. For example, a program may have a text view, a tree view, a subtree view, and a database view. A view may have more than one window. Each view has associated state information which is shared by all windows on that view.

View List  A Pan view style for browsing and operating on the list of all view instances in a Pan session. Quitting the view list terminates Pan.

View-Style  A configurable editing context. Pan has several different view-styles and every view must belong so some view-style.

Window  The equivalent of a window in Emacs, a window displays a view. Each text window has its own text cursor but all other state is shared by other windows on the same view.

Yank  Copy a region of text from the top of the kill ring into the current text stream.
A Default Key Bindings

A.1 Bindings by Command Name in a Text View

- ^X = Announce-Text-Cursor-Info
- Esc ^B Backward-Expr
- Esc v Backward-Vscroll-Current-Window
- Esc c Backward-Vscroll-Current-Window
- Capitalize-Selection
- Capitalize-Word
- Close-Current-Window
- Copy-Selection-As-kill
- Copy-Selection-As-Kill
- Copy-Selection-To-Cursor
- Copy-To-Clipboard
- Cursor-To-Mouse
- Cut-To-Clipboard
- Cycle-Paste
- Cycle-Show-Clipboard
- Cycle-Show-Kill
- Cycle-Yank
- Cycle-Yank
- Delete-Blank-Lines
- Delete-Character
- Delete-Horizontal-Space
- Delete-Indentation
- Delete-Previous-Character
- Delete-Previous-Character
- Deselect-Region
- Display-Objects
- Editor-Error
- Editor-Error
- Editor-Error
- Editor-Error
- Editor-Error
- Execute-Command
- Execute-Command
- Execute-Lisp
- Execute-Menu
- Exit-Editor
- First-Non-Blank
- First-Non-Blank
- Forward-Expr
- Forward-Vscroll-Current-Window
- Goto-Line
- Goto-Line
- Indent-Like-Previous-Line
- Insert-File
- Insert-Newline-Fill
- Insert-Parentheses
- Insert-Space-Fill
- Interrupt-Editor
- Esc Tab
- ^X 1
- Return
- Esc ( Space
- Esc ^Z
- Esc $K
- Esc Delete
- ^W
- ^K
- Esc D
- Esc d
- Esc <
- Esc ^L
- Esc ^S
- Esc X
- Esc L
- Esc l
- Esc Return
- ^A
- Esc <
- Esc E
- Esc >
- Newline
- ^F
- ^N
- Esc N
- Esc n
- Esc F
- Esc f
- ^X 2
- ^C 0
- Left-Arrow
- ^C ^B
- Prior
- Right-Arrow
- ^C ^F
- Next
- ^C ^I
- Down-Arrow
- ^C ^D
- Up-Arrow
- ^C ^R
- ^C ^S
- Find
A DEFAULT KEY BINDINGS

Oplevel-Cursor-Search-Forward | L9  | Self-Insert
Oplevel-Cursor-To-First      | "C Esc < | Self-Insert
Oplevel-Cursor-To-First      | "C <  | Set-Mark
Oplevel-Cursor-To-First      | Begin | Set-Oplevel-To-Character
Oplevel-Cursor-To-Last       | "C Esc > | Set-Oplevel-To-Line
Oplevel-Cursor-To-Last       | "C >  | Set-Oplevel-To-Word
Oplevel-Cursor-To-Last       | End   | Set-Text-Fill-Prefix
Oplevel-Cursor-To-Mouse      | Control-Mouse-Left | Set-Text-Line-Length
Oplevel-Cursor-To-Mouse      | Shift-Mouse-Left | Set-Text-Line-Length
Oplevel-Cursor-To-Mouse      | "C Mouse-Left | Show-Clipboard
Oplevel-Delete               | "C D  | Show-Kill
Oplevel-Menu                 | Control-Mouse-Right | Show-Match
Oplevel-Menu                 | Shift-Mouse-Right | Split-Line
Oplevel-Menu                 | "C Mouse-Right | Swap-Dot-And-Mark
Oplevel-Mouse-Extend         | Control-Mouse-Middle | Text-Fill-To-Blank-Line
Oplevel-Mouse-Extend         | Shift-Mouse-Middle | Text-Fill-To-Blank-Line
Oplevel-Mouse-Extend         | "C Mouse-Middle | Toggle-Object-Protection
Oplevel-Query-Replace-All    | "C Esc "R | Toggle-Text-Fill
Oplevel-Query-Replace-All    | "C Esc R  | Transpose-Characters
Oplevel-Query-Replace-All    | "C Esc r  | Transpose-Lines
Oplevel-Select               | "C Backspace | Undo
Paste-From-Clipboard        | L8    | Undo
Prefix                      | "U    | Undo
Preserve-All-Objects        | "X Return | Uppercase-Selection
Prev-Message-Current-Window | Esc P  | Uppercase-Word
Prev-Message-Current-Window | Esc p  | Uppercase-Word
Previous-Character           | "B    | Visit-Directory
Previous-Character           | "P    | Visit-Directory
Previous-Word                | Esc B  | Visit-Directory
Protected-Visit-File         | Esc b  | Visit-File
Query-Replace-All            | "X "R | Visit-Help-View
Quote-Character              | Esc "R | Visit-Symbol-Definition
Re-Search-Backward           | "Q    | Visit-View
Re-Search-Forward            | "R    | Visit-View
Redo                         | "S    | Visit-View-List
Redo                         | "X R  | Visit-View-List
Redo                         | "X r  | Yank-From-Kill-Ring
Redraw-Current-Window        | "L    |
Remove-View                  | "X K  |
Remove-View                  | "X k  |
Replace-All                  | Esc R  |
Replace-All                  | Esc r  |
Revert-Object                | "X Backspace |
Right-Hscroll-Current-Window | "X >  |
Save-Object-Copy-As          | "X "W |
Save-Object                  | "X "S |
Select-Expr                  | Esc "Q |
Select-Region-Dot-To-Mark    | Esc "W |
Select-Region-Dot-To-Mouse   | Mouse-Middle |
Select-Text-Stream           | "X H  |
Select-Text-Stream           | "X h  |
Select-Word                  | Esc Q  |
A.2 Bindings by Key in a Text View

<table>
<thead>
<tr>
<th>Key</th>
<th>Command</th>
<th>1.0</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>A</code></td>
<td>Set-Mark</td>
<td><code>G</code></td>
</tr>
<tr>
<td><code>A</code></td>
<td>Move-To-Bo</td>
<td><code>I</code></td>
</tr>
<tr>
<td><code>B</code></td>
<td>Previous-Character</td>
<td><code>L</code></td>
</tr>
<tr>
<td><code>B</code></td>
<td>Oplevel-Delete</td>
<td><code>Q</code></td>
</tr>
<tr>
<td><code>B</code></td>
<td>Oplevel-Cursor-Backward</td>
<td><code>R</code></td>
</tr>
<tr>
<td><code>C</code></td>
<td>Oplevel-Cursor-Forward</td>
<td><code>S</code></td>
</tr>
<tr>
<td><code>C</code></td>
<td>Oplevel-Select</td>
<td><code>T</code></td>
</tr>
<tr>
<td><code>C</code></td>
<td>Oplevel-Cursor-Out</td>
<td><code>W</code></td>
</tr>
<tr>
<td><code>C</code></td>
<td>Oplevel-Cursor-In</td>
<td><code>X</code></td>
</tr>
<tr>
<td><code>C</code></td>
<td>Oplevel-Cursor-Search-Backward</td>
<td><code>Y</code></td>
</tr>
<tr>
<td><code>C</code></td>
<td>Oplevel-Cursor-Search-Forward</td>
<td><code>Z</code></td>
</tr>
<tr>
<td><code>C</code></td>
<td>Oplevel-Query-Replace-All</td>
<td><code>!</code></td>
</tr>
<tr>
<td><code>C</code></td>
<td>Oplevel-Cursor-To-First</td>
<td><code>0</code></td>
</tr>
<tr>
<td><code>C</code></td>
<td>Oplevel-Cursor-To-Last</td>
<td><code>2</code></td>
</tr>
<tr>
<td><code>C</code></td>
<td>Oplevel-Replace-All</td>
<td><code>&lt;</code></td>
</tr>
<tr>
<td><code>C</code></td>
<td>Oplevel-Replace-All</td>
<td><code>=</code></td>
</tr>
<tr>
<td><code>C</code></td>
<td>Oplevel-Cursor-To-First</td>
<td><code>&gt;</code></td>
</tr>
<tr>
<td><code>C</code></td>
<td>Oplevel-Cursor-To-Last</td>
<td><code>?</code></td>
</tr>
<tr>
<td><code>C</code></td>
<td>Set-Current-To-Character</td>
<td><code>B</code></td>
</tr>
<tr>
<td><code>C</code></td>
<td>Set-Current-To-Line</td>
<td><code>D</code></td>
</tr>
<tr>
<td><code>C</code></td>
<td>Set-Current-To-Word</td>
<td><code>F</code></td>
</tr>
<tr>
<td><code>C</code></td>
<td>Oplevel-Cursor-To-Mouse</td>
<td><code>H</code></td>
</tr>
<tr>
<td><code>C</code></td>
<td>Oplevel-Cursor-Extend</td>
<td><code>K</code></td>
</tr>
<tr>
<td><code>C</code></td>
<td>Oplevel-Menu</td>
<td><code>L</code></td>
</tr>
<tr>
<td><code>C</code></td>
<td>Editor-Error</td>
<td><code>R</code></td>
</tr>
<tr>
<td><code>D</code></td>
<td>Delete-Character</td>
<td><code>U</code></td>
</tr>
<tr>
<td><code>E</code></td>
<td>Move-To-Eol</td>
<td><code>b</code></td>
</tr>
<tr>
<td><code>F</code></td>
<td>Next-Character</td>
<td><code>d</code></td>
</tr>
<tr>
<td><code>G</code></td>
<td>Editor-Error</td>
<td><code>f</code></td>
</tr>
<tr>
<td><code>H</code></td>
<td>Delete-Previous-Character</td>
<td><code>h</code></td>
</tr>
<tr>
<td><code>I</code></td>
<td>Self-Insert</td>
<td><code>k</code></td>
</tr>
<tr>
<td><code>K</code></td>
<td>Newline-And-Indent</td>
<td><code>l</code></td>
</tr>
<tr>
<td><code>L</code></td>
<td>Kill-To-Eol</td>
<td><code>r</code></td>
</tr>
<tr>
<td><code>L</code></td>
<td>Redraw-Current-Window</td>
<td><code>u</code></td>
</tr>
<tr>
<td><code>L</code></td>
<td>Insert-Newline-Fill</td>
<td><code>X</code></td>
</tr>
<tr>
<td><code>N</code></td>
<td>Next-Line</td>
<td><code>Y</code></td>
</tr>
<tr>
<td><code>O</code></td>
<td>Open-Line</td>
<td><code>Z</code></td>
</tr>
<tr>
<td><code>P</code></td>
<td>Previous-Line</td>
<td><code>G</code></td>
</tr>
<tr>
<td><code>Q</code></td>
<td>Quote-Character</td>
<td><code>0</code></td>
</tr>
<tr>
<td><code>R</code></td>
<td>Re-Search-Backward</td>
<td><code>B</code></td>
</tr>
<tr>
<td><code>S</code></td>
<td>Re-Search-Forward</td>
<td><code>C</code></td>
</tr>
<tr>
<td><code>T</code></td>
<td>Transpose-Characters</td>
<td><code>D</code></td>
</tr>
<tr>
<td><code>U</code></td>
<td>Prefix</td>
<td><code>F</code></td>
</tr>
<tr>
<td><code>V</code></td>
<td>Forward-Current-Window</td>
<td><code>G</code></td>
</tr>
<tr>
<td><code>W</code></td>
<td>Kill-Selected-Region</td>
<td><code>Tab</code></td>
</tr>
<tr>
<td><code>X</code></td>
<td>Kill-Selected-Region</td>
<td><code>K</code></td>
</tr>
<tr>
<td><code>Y</code></td>
<td>Toggle-Text-Fill</td>
<td><code>L</code></td>
</tr>
<tr>
<td><code>Z</code></td>
<td>Display-Objects</td>
<td><code>Return</code></td>
</tr>
<tr>
<td><code>A</code></td>
<td>Exit-Editor</td>
<td><code>0</code></td>
</tr>
<tr>
<td><code>B</code></td>
<td>Exit-Editor</td>
<td><code>1</code></td>
</tr>
<tr>
<td><code>C</code></td>
<td>Exit-Editor</td>
<td><code>2</code></td>
</tr>
<tr>
<td><code>D</code></td>
<td>Exit-Editor</td>
<td><code>3</code></td>
</tr>
<tr>
<td><code>F</code></td>
<td>Exit-Editor</td>
<td><code>4</code></td>
</tr>
</tbody>
</table>

Additional bindings:

<table>
<thead>
<tr>
<th>Esc \r</th>
<th>Query-Replace-All</th>
<th>Esc \r8</th>
<th>Cycle-Paste</th>
</tr>
</thead>
<tbody>
<tr>
<td>Esc \s</td>
<td>Load-From-Selection</td>
<td>Space</td>
<td>Insert-Space-Fill</td>
</tr>
<tr>
<td>Esc \u</td>
<td>Uppercase-Selection</td>
<td>!to`</td>
<td>Self-Insert</td>
</tr>
<tr>
<td>Esc \w</td>
<td>Select-Region-Dot-To-Mark</td>
<td>Delete</td>
<td>Delete-Previous-Character</td>
</tr>
<tr>
<td>Esc \x</td>
<td>Load-From-Text-Stream</td>
<td>Mouse-Left</td>
<td>Cursor-To-Mouse</td>
</tr>
<tr>
<td>Esc \y</td>
<td>Copy-Selection-To-Cursor</td>
<td>Mouse-Middle</td>
<td>Select-Region-Dot-To-Mouse</td>
</tr>
<tr>
<td>Esc \esc</td>
<td>Execute-Lisp</td>
<td>Mouse-Right</td>
<td>Execute-Menu</td>
</tr>
<tr>
<td>Esc Space</td>
<td>Just-One-Space</td>
<td>Control-Mouse-Left</td>
<td>Oplevel-Cursor-To-Mouse</td>
</tr>
<tr>
<td>Esc !</td>
<td>Cycle-Show-Clipboard</td>
<td>Control-Mouse-Middle</td>
<td>Oplevel-Mouse-Extend</td>
</tr>
<tr>
<td>Esc %</td>
<td>Show-Match</td>
<td>Control-Mouse-Right</td>
<td>Oplevel-Menu</td>
</tr>
<tr>
<td>Esc ()</td>
<td>Insert-Parentheses</td>
<td>Shift-Mouse-Left</td>
<td>Oplevel-Cursor-To-Menu</td>
</tr>
<tr>
<td>Esc .</td>
<td>Visit-Symbol-Definition</td>
<td>Shift-Mouse-Middle</td>
<td>Oplevel-Cursor-Extend</td>
</tr>
<tr>
<td>Esc &lt;</td>
<td>Move-To-Bos</td>
<td>Shift-Mouse-Right</td>
<td>Oplevel-Menu</td>
</tr>
<tr>
<td>Esc &gt;</td>
<td>Move-To-Eos</td>
<td>Home</td>
<td>Visit-View-List</td>
</tr>
<tr>
<td>Esc ?</td>
<td>Show-Clipboard</td>
<td>Left-Arrow</td>
<td>Oplevel-Cursor-Backward</td>
</tr>
<tr>
<td>Esc Q</td>
<td>Select-Word</td>
<td>Up-Arrow</td>
<td>Oplevel-Cursor-Out</td>
</tr>
<tr>
<td>Esc B</td>
<td>Previous-Word</td>
<td>Right-Arrow</td>
<td>Oplevel-Cursor-Forward</td>
</tr>
<tr>
<td>Esc C</td>
<td>Capitalize-Word</td>
<td>Down-Arrow</td>
<td>Oplevel-Cursor-In</td>
</tr>
<tr>
<td>Esc D</td>
<td>Kill-Word</td>
<td>Prior</td>
<td>Oplevel-Cursor-Backward</td>
</tr>
<tr>
<td>Esc F</td>
<td>Next-Word</td>
<td>Next</td>
<td>Oplevel-Cursor-Forward</td>
</tr>
<tr>
<td>Esc L</td>
<td>Lowercase-Word</td>
<td>End</td>
<td>Oplevel-Cursor-To-Last</td>
</tr>
<tr>
<td>Esc M</td>
<td>First-Non-Blank</td>
<td>Find</td>
<td>Oplevel-Cursor-To-First</td>
</tr>
<tr>
<td>Esc N</td>
<td>Next-Message-Current-Window</td>
<td>Help</td>
<td>Oplevel-Cursor-Search-Forward</td>
</tr>
<tr>
<td>Esc P</td>
<td>Prev-Message-Current-Window</td>
<td>Cancel</td>
<td>Editor-Error</td>
</tr>
<tr>
<td>Esc Q</td>
<td>Text-Fill-To-Blank-Line</td>
<td>L4</td>
<td>Visit-Help-View</td>
</tr>
<tr>
<td>Esc R</td>
<td>Replace-All</td>
<td>L6</td>
<td>Undo</td>
</tr>
<tr>
<td>Esc U</td>
<td>Uppercase-Word</td>
<td>L8</td>
<td>Copy-To-Clipboard</td>
</tr>
<tr>
<td>Esc V</td>
<td>Backward-Vscroll-Current-Window</td>
<td>L9</td>
<td>Paste-From-Clipbord</td>
</tr>
<tr>
<td>Esc W</td>
<td>Copy-Selection-As-Kill</td>
<td>L10</td>
<td>Oplevel-Cursor-Search-Forward</td>
</tr>
<tr>
<td>Esc X</td>
<td>Execute-Command</td>
<td>R7</td>
<td>Cut-To-Clipbord</td>
</tr>
<tr>
<td>Esc Y</td>
<td>Cycle-Yank</td>
<td></td>
<td>Visit-View-List</td>
</tr>
</tbody>
</table>

| Esc \ \ | Delete-Horizontal-Space | | |
| Esc ` | Delete-Indentation | | |
| Esc b | Previous-Word | | |
| Esc c | Capitalize-Word | | |
| Esc d | Kill-Word | | |
| Esc f | Next-Word | | |
| Esc l | Lowercase-Word | | |
| Esc m | First-Non-Blank | | |
| Esc n | Next-Message-Current-Window | | |
| Esc p | Prev-Message-Current-Window | | |
| Esc q | Text-Fill-To-Blank-Line | | |
| Esc r | Replace-All | | |
| Esc u | Uppercase-Word | | |
| Esc v | Backward-Vscroll-Current-Window | | |
| Esc w | Copy-Selection-As-Kill | | |
| Esc x | Execute-Command | | |
| Esc y | Cycle-Yank | | |
| Esc Del | Kill-Previous-Word | | |
| Esc Mouse-Left | | | Mouse-Select-Fullword |
| Esc Mouse-Middle | Mouse-Extend-Selection-Fullword | | |
| Esc Cancel | Editor-Error | | |
A.3  Special Key Bindings in the View List

`B` Previous-Line-Select
`F` Next-Line-Select
`N` Previous-Line-Select
`P` Next-Line-Select
`i` Visit-Selected-View
`x` Remove-Selected-View
Mouse-Left Mouse-Select-Full-Line
Mouse-Middle Mouse-Select-Full-Line
Left-Arrow Previous-Line-Select
Up-Arrow Previous-Line-Select
Right-Arrow Next-Line-Select
Down-Arrow Next-Line-Select

A.4  Special Key Bindings in the Help View

Mouse-Left Mouse-Select-Fullword
Mouse-Middle Mouse-Extend-Selection-Fullword
Left-Arrow Previous-Character
Right-Arrow Next-Character

A.5  Special Key Bindings in a Directory Editor

`B` Previous-Line-Select
`D` Dired-Mark-Selection
`F` Next-Line-Select
`N` Next-Line-Select
`P` Previous-Line-Select
`R` Dired-Visit-Read-Only
Escape Mouse-Left Mouse-Select-Full-Line
Escape Mouse-Middle Mouse-Extend-Selection-Full-Line
# Dired-Mark-Editor-Files
`D` Dired-Mark-Selection
F Dired-Visit
G Revert-Object
Q Remove-Object
U Dired-Unmark-Selection
X Dired-Delete-Files
`d` Dired-Mark-Selection
f Dired-Visit
`g` Revert-Object
q Remove-Object
u Dired-Unmark-Selection
x Dired-Delete-Files
Mouse-Left Mouse-Select-Full-Line
Mouse-Middle Mouse-Extend-Selection-Full-Line

A.6  Special Key Bindings in a Language-Based View (SIMPLE2)

`c -x` Announce-Structure-Cursor-Info
`c x` Set-OpLevel-To-Syntactic-Error
`c #` Set-OpLevel-To-Semantic-Error
`c =` Announce-Structure-Cursor-Node-Name
`c q` Set-OpLevel-To-Simple2-Placeholder
`c a` Analyze-Changes
`c d` Set-OpLevel-To-Simple2-Declaration
`c e` Set-OpLevel-To-Simple2-Expression
`c p` Set-OpLevel-To-Simple2-Placeholder
`c g` Set-OpLevel-To-Query
`c s` Set-OpLevel-To-Simple2-Statement
`c x` Clear-Structure-Cursor
Escape `=d`

A.7  Special Key Bindings in a Graphical View

a Global-Image-Enlarge
e Global-Image-Toggle
f Zoom-In
i Zoom-Out
o Global-Image-Shrink
s
# B Default Menu Bindings

## B.1 Bindings in an Ordinary Text View by Command Name

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Append-Selection-To-File</td>
<td>Text Edit ⇒ File: Append Selection To...</td>
</tr>
<tr>
<td>Apropos-String</td>
<td>Help: Apropos</td>
</tr>
<tr>
<td>Close-Current-Window</td>
<td>Window: Close</td>
</tr>
<tr>
<td>Copy-Selection-To-Cursor</td>
<td>Text Edit: Copy To Cursor</td>
</tr>
<tr>
<td>Copy-To-Clipboard</td>
<td>Clipboard: Copy</td>
</tr>
<tr>
<td>Cut-To-Clipboard</td>
<td>Clipboard: Cut</td>
</tr>
<tr>
<td>Cycle-Show-Clipboard</td>
<td>Clipboard: Cycle and Show</td>
</tr>
<tr>
<td>Display-All-Key-Bindings</td>
<td>Help⇒ This View⇒ Keys: Display All</td>
</tr>
<tr>
<td>Display-Auto-Exec</td>
<td>Help⇒ This View⇒Filename Auto-Exec Map</td>
</tr>
<tr>
<td>Display-Auto-Load</td>
<td>Help⇒ This View⇒Filename Auto-Load Map</td>
</tr>
<tr>
<td>Display-Bug-Report-Info</td>
<td>Help⇒ This View⇒ Programs: Bug Report Errors</td>
</tr>
<tr>
<td>Display-Char-Sets</td>
<td>Help⇒ This View⇒ Char Sets</td>
</tr>
<tr>
<td>Display-Commands</td>
<td>Help⇒ This View⇒ Commands Available</td>
</tr>
<tr>
<td>Display-Default-Menu</td>
<td>Help⇒ This View⇒ Default Menu</td>
</tr>
<tr>
<td>Display-Flags</td>
<td>Help⇒ This View⇒ Flags</td>
</tr>
<tr>
<td>Display-Key-Bindings-For-Command</td>
<td>Help⇒ This View⇒ Keys: Command - Key&gt;</td>
</tr>
<tr>
<td>Display-Key-Binding</td>
<td>Help⇒ This View⇒ Keys: Key - Command&gt;</td>
</tr>
<tr>
<td>Display-Language-List</td>
<td>Help⇒ This View⇒ Language Descriptions Loaded</td>
</tr>
<tr>
<td>Display-Objects</td>
<td>Help⇒ This View⇒ Objects Being Viewed</td>
</tr>
<tr>
<td>Display-Operand-Command-Bindings</td>
<td>Help⇒ This View⇒ Operand Cmds.: By Command</td>
</tr>
<tr>
<td>Display-Operand-Level-Bindings</td>
<td>Help⇒ This View⇒ Operand Cmds.: By Level</td>
</tr>
<tr>
<td>Display-Operand-Level-Menus</td>
<td>Help⇒ This View⇒ Operand Menus</td>
</tr>
<tr>
<td>Display-Option-Variables</td>
<td>Help⇒ This View⇒ Options</td>
</tr>
<tr>
<td>Display-Version</td>
<td>Help⇒ This View⇒ Version</td>
</tr>
<tr>
<td>Insert-File</td>
<td>Text Edit⇒ File: Insert Text From...</td>
</tr>
<tr>
<td>Kill-Selected-Region</td>
<td>Text Edit: Kill</td>
</tr>
<tr>
<td>Move-Selection-To-Cursor</td>
<td>Text Edit: Move To Cursor</td>
</tr>
<tr>
<td>Open-Another-Window-In-View</td>
<td>Window: Open Another</td>
</tr>
<tr>
<td>Paste-From-Clipboard</td>
<td>Clipboard: Paste</td>
</tr>
<tr>
<td>Redo</td>
<td>Undo: Redo</td>
</tr>
<tr>
<td>Redraw-Current-Window</td>
<td>Window: Redraw</td>
</tr>
<tr>
<td>Remove-View</td>
<td>Undo: Remove</td>
</tr>
<tr>
<td>Revert-Object</td>
<td>Store: Revert</td>
</tr>
<tr>
<td>Save-Object-Copy-As</td>
<td>Store: Save Copy As</td>
</tr>
<tr>
<td>Save-Object</td>
<td>Store: Save</td>
</tr>
<tr>
<td>Scroll-To-Cursor-Current-Window</td>
<td>Window: Scroll to Cursor</td>
</tr>
<tr>
<td>Set-Global-Option-Variable</td>
<td>Help⇒ This View⇒ Set Option: Global</td>
</tr>
<tr>
<td>Set-View-Option-Variable</td>
<td>Help⇒ This View⇒ Set Option: View Only</td>
</tr>
<tr>
<td>Set-View-Style-Option-Variable</td>
<td>Help⇒ This View⇒ Set Option: View Style</td>
</tr>
<tr>
<td>Show-Clipboard</td>
<td>Clipboard: Show Contents</td>
</tr>
<tr>
<td>Sub-Redo</td>
<td>Undo: Sub-Redo</td>
</tr>
<tr>
<td>Sub-Undo</td>
<td>Undo: Sub-Undo</td>
</tr>
<tr>
<td>Toggle-Object-Protection</td>
<td>Text Edit: Toggle Protection</td>
</tr>
<tr>
<td>Undo</td>
<td>Undo: Undo</td>
</tr>
<tr>
<td>Visit-File</td>
<td>Store: Visit New</td>
</tr>
<tr>
<td>Write-Selection-To-File</td>
<td>Text Edit⇒ File: Write Selection To...</td>
</tr>
<tr>
<td>Yank-From-Kill-Ring</td>
<td>Text Edit: Yank</td>
</tr>
</tbody>
</table>
### B.2 Bindings in an Ordinary Text View by Menu

<table>
<thead>
<tr>
<th>Command</th>
<th>Binding</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clipboard: Cut</td>
<td>Cut-To-Clipboard</td>
</tr>
<tr>
<td>Clipboard: Copy</td>
<td>Copy-To-Clipboard</td>
</tr>
<tr>
<td>Clipboard: Paste</td>
<td>Paste-From-Clipboard</td>
</tr>
<tr>
<td>Clipboard: Show Contents</td>
<td>Show-Clipboard</td>
</tr>
<tr>
<td>Clipboard: Cycle and Show</td>
<td>Cycle-Show-Clipboard</td>
</tr>
<tr>
<td>Text Edit: Kill</td>
<td>Kill-Selected-Region</td>
</tr>
<tr>
<td>Text Edit: Yank</td>
<td>Yank-From-Kill-Ring</td>
</tr>
<tr>
<td>Text Edit: Copy To Cursor</td>
<td>Copy-Selection-To-Cursor</td>
</tr>
<tr>
<td>Text Edit: Move To Cursor</td>
<td>Move-Selection-To-Cursor</td>
</tr>
<tr>
<td>Text Edit: File</td>
<td>Insert-File</td>
</tr>
<tr>
<td>Insert Text From... Text Edit</td>
<td>Write-Selection-To-File</td>
</tr>
<tr>
<td>Edit: File: Append Selection</td>
<td>Append-Selection-To-File</td>
</tr>
<tr>
<td>Text Edit: Toggle Protection</td>
<td>Toggle-Object-Protection</td>
</tr>
<tr>
<td>Undo</td>
<td>Undo</td>
</tr>
<tr>
<td>Undo: Undo</td>
<td>Undo</td>
</tr>
<tr>
<td>Undo: Redo</td>
<td>Redo</td>
</tr>
<tr>
<td>Undo: Sub-Undo</td>
<td>Sub-Undo</td>
</tr>
<tr>
<td>Undo: Sub-Redo</td>
<td>Sub-Redo</td>
</tr>
<tr>
<td>Store: Save</td>
<td>Save-Object</td>
</tr>
<tr>
<td>Store: Save Copy As</td>
<td>Save-Object-Copy-As</td>
</tr>
<tr>
<td>Store: Revert</td>
<td>Revert-Object</td>
</tr>
<tr>
<td>Store: Visit New</td>
<td>Visit-File</td>
</tr>
<tr>
<td>View: Remove</td>
<td>Remove-View</td>
</tr>
<tr>
<td>Window: Close</td>
<td>Close-Current-Window</td>
</tr>
<tr>
<td>Window: Open Another</td>
<td>Open-Another-Window-In-View</td>
</tr>
<tr>
<td>Window: Scroll to Cursor</td>
<td>Scroll-To-Cursor-Current-Window</td>
</tr>
<tr>
<td>Window: Redraw</td>
<td>Redraw-Current-Window</td>
</tr>
<tr>
<td>Help⇒This View⇒Keys: Display</td>
<td>Display-All-Key-Bindings</td>
</tr>
<tr>
<td>Help⇒This View⇒Keys: Key - Command&gt;</td>
<td>Display-Key-Binding</td>
</tr>
<tr>
<td>Help⇒This View⇒Keys: Command - Key&gt;</td>
<td>Display-Key-Bindings-For-Command</td>
</tr>
<tr>
<td>Help⇒This View: Default Menu</td>
<td>Display-Default-Menu</td>
</tr>
<tr>
<td>Help⇒This View⇒Operand Cmds.: By Level</td>
<td>Display-Operand-Level-Bindings</td>
</tr>
<tr>
<td>Help⇒This View⇒Operand Cmds.: By Command</td>
<td>Display-Operand-Command-Bindings</td>
</tr>
<tr>
<td>Help⇒This View: Operand Menus</td>
<td>Display-Operand-Level-Menus</td>
</tr>
<tr>
<td>Help⇒This View: Options</td>
<td>Display-Option-Variables</td>
</tr>
<tr>
<td>Help⇒This View⇒Set Option: View Only</td>
<td>Set-View-Option-Variable</td>
</tr>
<tr>
<td>Help⇒This View⇒Set Option: View Style</td>
<td>Set-View-Style-Option-Variable</td>
</tr>
<tr>
<td>Help⇒This View⇒Set Option: Global</td>
<td>Set-Global-Option-Variable</td>
</tr>
<tr>
<td>Help⇒This View: Flags</td>
<td>Display-Flags</td>
</tr>
<tr>
<td>Help⇒This View: Char. Sets</td>
<td>Display-Char-Sets</td>
</tr>
<tr>
<td>Help⇒Apropos</td>
<td>Apropos-String</td>
</tr>
<tr>
<td>Help⇒Pan: Objects Being Viewed</td>
<td>Display-Objects</td>
</tr>
<tr>
<td>Help⇒Pan: Language Descriptions Loaded</td>
<td>Display-Language-List</td>
</tr>
<tr>
<td>Help⇒Pan: Commands Available</td>
<td>Display-Commands</td>
</tr>
<tr>
<td>Help⇒Pan: Filename Auto-Load Map</td>
<td>Display-Auto-Load</td>
</tr>
<tr>
<td>Help⇒Pan: Filename Auto-Exec Map</td>
<td>Display-Auto-Exec</td>
</tr>
<tr>
<td>Help⇒Pan: Version</td>
<td>Display-Version</td>
</tr>
<tr>
<td>Help⇒Pan: How To Report Bugs</td>
<td>Display-Bug-Report-Info</td>
</tr>
</tbody>
</table>
B.3 Additional Menu Bindings in the View List

View List: Visit View .................................................. Visit-Selected-View
View List: Remove View .............................................. Remove-Selected-View
View List: Visit New .................................................. Visit-File
View List: New Scratch ............................................... Scratch-View
View List: Exit-Editor .................................................. Quit

B.4 Additional Menu Bindings in the Help View

Help View: Describe .................................................. Document-Symbol
Help View: Visit Definition ......................................... Visit-Symbol-Definition
Help View: Apropos .................................................... Apropos-String
Help View: Reset ...................................................... Reset-Help

B.5 Additional Menu Bindings in a Directory Editor

Dired: Visit ............................................................ Dired-Visit
Dired: Visit Read Only ................................................. Dired-Visit-Read-Only
Dired: Mark ............................................................. Dired-Mark-Selection
Dired: Unmark .......................................................... Dired-Unmark-Selection
Dired: Delete marked files ........................................... Dired-Delete-Files
Dired: Mark backup files .............................................. Dired-Mark-Editor-Files
Dired: Reread directory ............................................... Revert-Object
Dired: Undo .............................................................. Undo

B.6 Additional Menu Bindings in a Language-Based View (SIMPLE2)

Language: Analyze .................................................... Analyze-Changes
Variables: Show Declaration and Uses ................................ Query-Simple2-Variable
Variables: Goto Declaration .......................................... Goto-Simple2-Variable-Declaration
Variables: Rename Variable .......................................... Rename-Simple2-Variable
View⇒Program Graph: With Node Names ......................... Visit-Program-Graph-Name-View
View⇒Program Graph: With Operators ............................ Visit-Program-Graph-Operator-View
View⇒Subgraph At Cursor: With Node Names .................... Visit-Program-Subgraph-Name-View
View⇒Subgraph At Cursor: With Operators ....................... Visit-Program-Subgraph-Operator-View

B.7 Additional Menu Bindings in a Graphical View

Window: Toggle Global Image ....................................... Global-Image-Toggle
Window: Enlarge Global Image ...................................... Global-Image-Enlarge
Window: Shrink Global Image ........................................ Global-Image-Shrink
Window: Zoom In ...................................................... Zoom-In
Window: Zoom Out ..................................................... Zoom-Out
Window: Zoom All ...................................................... Zoom-All
C  Operand Level Bindings in a Language-Based View (SILENCE2)

Level: "Character"

Operand Level Bindings

- Previous-Character
- Next-Character
- Next-Line
- Previous-Line
- Re-Search-Backward
- Re-Search-Forward
- Move-To-Bos
- Move-To-Eos
- Cursor-To-Mouse
- Delete-Character
- Execute-Operand-Menu
- Select-Region-Dot-To-Mouse
- Query-Replace-All
- Replace-All
- Deselect-Region

Level: "Word"

Operand Level Bindings

- Previous-Word
- Next-Word
- Next-Line
- Previous-Line
- Move-To-First-Word
- Move-To-Last-Word
- Mouse-Select-Fullword
- Delete-Fullword
- Execute-Operand-Menu
- Mouse-Extend-Operand-Text
- Select-Fullword

Level: "Line"

Operand Level Bindings

- Previous-Line
- Next-Line
- Next-Line
- Previous-Line
- Re-Search-Backward
- Re-Search-Forward
- Move-To-Bos
- Move-To-Last-Line
- Mouse-Select-Line
- Delete-Line
- Execute-Operand-Menu
- Mouse-Extend-Selection-Full-Line
- Query-Replace-All
- Replace-All
- Select-Line
<table>
<thead>
<tr>
<th>Level: &quot;Lexeme&quot;</th>
<th>Operand Level Bindings</th>
</tr>
</thead>
<tbody>
<tr>
<td>Oplevel-Cursor-Backward:</td>
<td>Prev-Lexeme</td>
</tr>
<tr>
<td>Oplevel-Cursor-Forward:</td>
<td>Next-Lexeme</td>
</tr>
<tr>
<td>Oplevel-Cursor-Search-Forward:</td>
<td>Lexeme-Search-Forward</td>
</tr>
<tr>
<td>Oplevel-Cursor-To-First:</td>
<td>First-Lexeme</td>
</tr>
<tr>
<td>Oplevel-Cursor-To-Last:</td>
<td>Last-Lexeme</td>
</tr>
<tr>
<td>Oplevel-Cursor-To-Mouse:</td>
<td>Mouse-Select-Lexeme</td>
</tr>
<tr>
<td>Oplevel-Delete:</td>
<td>Delete-Lexeme</td>
</tr>
<tr>
<td>Oplevel-Menu:</td>
<td>Execute-Oplevel-Menu</td>
</tr>
<tr>
<td>Oplevel-Mouse-Extend:</td>
<td>Mouse-Extend-Lexeme</td>
</tr>
<tr>
<td>Oplevel-Query-Replace-All:</td>
<td>Query-Replace-Lexemes</td>
</tr>
<tr>
<td>Oplevel-Replace-All:</td>
<td>Replace-Lexemes</td>
</tr>
<tr>
<td>Oplevel-Select:</td>
<td>Select-Lexeme</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Level: &quot;Expression&quot;</th>
<th>Operand Level Bindings</th>
</tr>
</thead>
<tbody>
<tr>
<td>Oplevel-Cursor-Backward:</td>
<td>Prev-Oplevel-Node</td>
</tr>
<tr>
<td>Oplevel-Cursor-Forward:</td>
<td>Next-Oplevel-Node</td>
</tr>
<tr>
<td>Oplevel-Cursor-In:</td>
<td>In-Oplevel-Node</td>
</tr>
<tr>
<td>Oplevel-Cursor-Out:</td>
<td>Out-Oplevel-Node</td>
</tr>
<tr>
<td>Oplevel-Cursor-To-First:</td>
<td>First-Oplevel-Node</td>
</tr>
<tr>
<td>Oplevel-Cursor-To-Last:</td>
<td>Last-Oplevel-Node</td>
</tr>
<tr>
<td>Oplevel-Cursor-To-Mouse:</td>
<td>Mouse-Select-Oplevel-Node</td>
</tr>
<tr>
<td>Oplevel-Delete:</td>
<td>Delete-Oplevel-Node</td>
</tr>
<tr>
<td>Oplevel-Menu:</td>
<td>Execute-Oplevel-Menu</td>
</tr>
<tr>
<td>Oplevel-Mouse-Extend:</td>
<td>Mouse-Extend-Oplevel-Node</td>
</tr>
<tr>
<td>Oplevel-Select:</td>
<td>Select-Oplevel-Node</td>
</tr>
</tbody>
</table>
D  Options Defined in Pan

Apropos-Documentation-Sub-Type-List
Type: list
Default Value: (:option-variable :command :view-style)
Scopes: global
If non-nil, the apropos commands list only symbols defined with a documentation sub-type in this list.

Arrow-Mouse-Icon
Type: dotted pair of strings
Default Value: ("left ptr" , "left ptr")
Scopes: view object view-style global
Arrow mouse icon.

Auto-Show-Match
Type: boolean
Default Value: t
Scopes: view object view-style global
If T. insert of a right bracket shows matching left bracket.

Auto-Show-Match-Limit
Type: integer
Default Value: 2000
Scopes: view object view-style global
Bound on number of characters to scan when Auto-Show-Match is on.

Autowrap-Search
Type: boolean
Default Value: t
Scopes: view object view-style global
If T. search wraps around at Eos back to cursor; else to Eos only.

Backup-Object?
Type: boolean
Default Value: t
Scopes: object global
When true, the first time an object is saved, the original is backed up.

Case-Sensitive-Lexeme-Search
Type: boolean
Default Value: nil
Scopes: view object view-style global
When attempting to perform regular expression matching on lexemes, should comparisons be case sensitive?

Checkpoint-Modification-Interval
Type: integer
Default Value: 250
Scopes: object global
Checkpoint after this many modifications without saving or checkpointing.
Clipboard-Max-Size
Type: integer
Default Value: 8
Scopes: global
Maximum number of clips in clipboard.

Default-Line-Style
Type: list
Default Value: ("line-solid" "line-solid")
Scopes: view object view-style global
Default line style for objects displayed in graph window. Possible values are line-solid, line-on-off-dash or line-double-dash. All values are expressed as strings.

Demo
Type: boolean
Default Value: nil
Scopes: global
Controls special configuration modes suitable for demos.

Dired-Ls-Flags
Type: string
Default Value: "-a"
Scopes: global
String containing flags for /bin/ls command used in Directory Editor. The string must begin with the "-" prefix. The flag -l is supplied automatically, so need not be included. Any of the flags -acrFt may be used. Don’t use any of the flags -dfgIsR.

Editor-File-Search-List
Type: list
Default Value: ("." "--" "/usr/local/lib/pan/"")
Scopes: global
A list of directory names, specified as strings, to search when loading and requiring editor files. May include standard directory naming conventions like ".", "--", and "-user" where user must be a valid user name in the system. Used by Load-File, editor-load, and editor-require.

Editor-Files-Pattern
Type: string
Default Value: ".#Pan."
Scopes: global
Regexp pattern matching files created by Editor for backup and checkpointing.

Fast-Draw-Color
Type: dotted pair of strings
Default Value: ("midnight blue" "black")
Scopes: view object view-style global
Color to use when fast drawing outlines.
Global-Image-Initial-Height
Type: real
Default Value: 0.2
Scopes: view object view-style global
Initial height of global image as percentage of the graph window height.

Global-Image-Initial-On?
Type: boolean
Default Value: t
Scopes: view object view-style global
Determines whether global image is initially displayed.

Global-Image-Initial-Width
Type: real
Default Value: 0.2
Scopes: view object view-style global
Initial width of global image as percentage of the graph window width.

Global-Image-Outline-Color
Type: dotted pair of strings
Default Value: ("forest green" . "black")
Scopes: view object view-style global
Color to use for box around global image.

Graph-Window-Zoom-Factor
Type: real
Default Value: 0.1
Scopes: view object view-style global
Multiplier for zooming operations.

Graphical-Tree-Hbox-Background-Color
Type: dotted pair of strings
Default Value: ("white" . "white")
Scopes: view object view-style global
Background color for a horizontal box

Graphical-Tree-Hbox-Border-Color
Type: dotted pair of strings
Default Value: ("midnight blue" . "black")
Scopes: view object view-style global
Line color for a horizontal box

Graphical-Tree-Hbox-Style
Type: dotted pair of strings
Default Value: ("line-solid" . "line-solid")
Scopes: view object view-style global
Line style for a horizontal box
Graphical-Tree-Hbox-Border-Thickness
Type: integer
Default Value: 3
Scopes: view object view-style global
Line border for a horizontal box

Graphical-Tree-Hbox-Text-Color
Type: dotted pair of strings
Default Value: ("black", "black")
Scopes: view object view-style global
Text color for a horizontal box

Graphical-Tree-Horiz-Node-Spacing
Type: integer
Default Value: 15
Scopes: view object view-style global
Horizontal spacing between parent and child nodes

Graphical-Tree-Line-Arrow-Filled?
Type: integer
Default Value: 1
Scopes: view object view-style global
Directed edge arrow filled in?

Graphical-Tree-Line-Arrow-Length
Type: integer
Default Value: 6
Scopes: view object view-style global
Directed edge arrow length

Graphical-Tree-Line-Arrow-Width
Type: integer
Default Value: 4
Scopes: view object view-style global
Directed edge arrow width

Graphical-Tree-Line-Color
Type: dotted pair of strings
Default Value: ("midnight blue", "black")
Scopes: view object view-style global
Edge line color

Graphical-Tree-Line-Style
Type: dotted pair of strings
Default Value: ("line-solid", "line-solid")
Scopes: view object view-style global
Edge line style
Graphical-Tree-Line-Thickness
Type: integer
Default Value: 2
Scopes: view object view-style global
Edge line thickness

Graphical-Tree-Node-Border
Type: integer
Default Value: 4
Scopes: view object view-style global
Border ('whitespace') around a node

Graphical-Tree-Node-Type
Type: integer
Default Value: 0
Scopes: view object view-style global
Symbolic constant for the drawing figure for nodes

Help-View-At-Startup
Type: boolean
Default Value: t
Scopes: global
If non-nil, create an empty help view at startup.

Highlight-Query-Results
Type: boolean
Default Value: t
Scopes: view object view-style global
When t, visually highlight text associated with all components resulting from the most recent structural query.

Highlight-Semantic-Errors
Type: boolean
Default Value: t
Scopes: view object view-style global
When t, visually highlight the text associated with all components having unsatisfied semantic constraints.

Highlight-Structure-Cursor
Type: boolean
Default Value: t
Scopes: view object view-style global
When t, visually highlight text associated with the structure cursor's location.

Highlight-Syntax-Errors
Type: boolean
Default Value: t
Scopes: view object view-style global
When t, visually highlight the text associated with all components resulting from syntactic error recovery.
Indent-With-Tabs
Type: boolean
Default Value: nil
Scopes: view object view-style global
If T, use tabs when performing computed indentations; else just spaces.

Killring-Max-Size
Type: integer
Default Value: 16
Scopes: view object view-style global
Maximum number of elements retained in kill-ring.

Kills-To-Clipboard
Type: boolean
Default Value: nil
Scopes: view object view-style global
If true, kill commands use the global clipboard instead of the local kill-ring.

Lisp-Mouse-Icon
Type: dotted pair of strings
Default Value: ("star" . "star")
Scopes: view object view-style global
Lisp mouse icon.

Load-Verbose
Type: boolean
Default Value: t
Scopes: global
When non-nil, Load-File, editor-load, and editor-require write messages to standard output.

Modula2-Src-Dir
Type: boolean
Default Value: nil
Scopes: global
Directory containing source files for Modula2 standard modules. When present, semantic analysis imports information from them by visiting and analyzing imported modules.

Object-Modified?
Type: boolean
Default Value: nil
Scopes: object
True when the current object is modified.

Object-Protected?
Type: boolean
Default Value: nil
Scopes: object
True when the current object is protected.
Operand-Level-Choices
Type: boolean
Default Value: nil
Scopes: view object view-style global
A list of currently visible operand levels from which the user may select one to be "current". The default is the first in the list.

Package-Symbol-Names
Type: boolean
Default Value: nil
Scopes: global
If non-nil, help and other tools will attempt to print symbol names with their packages when that package is not user. Otherwise, tools will usually not print package names.

Panel-Bg-Color
Type: dotted pair of strings
Default Value: ("white" , "white")
Scopes: view object view-style global
Background color for window's panel.

Panel-Flag-Default-Bitmap
Type: string
Default Value: "asterisk.12x12"
Scopes: view object view-style global
The default bitmap for panel flags. Assigning this variable does not affect current flags.

Panel-Flag-Default-Fg-Color
Type: dotted pair of strings
Default Value: ("black" , "black")
Scopes: view object view-style global
The default foreground color for panel flags. Assigning this variable does not affect current flags.

Panel-Flags-Bg-Color
Type: dotted pair of strings
Default Value: ("white" , "white")
Scopes: view object view-style global
Background color behind panel flags.

Panel-Logo-Bg-Color
Type: dotted pair of strings
Default Value: ("white" , "white")
Scopes: view object view-style global
Background color for editor logo in panel.

Panel-Logo-Bitmap
Type: string
Default Value: "pan-edit.31x22"
Scopes: view object view-style global
Bitmap that specifies the editor logo; appears at left of every panel.
Panel-Logo-Fg-Color
Type: dotted pair of strings
Default Value: ("black", "black")
Scopes: view object view-style global
Foreground color for editor logo in panel.

Panel-View-Logo-Bg-Color
Type: dotted pair of strings
Default Value: ("white", "white")
Scopes: view object view-style global
Background color for view style logo in panel.

Panel-View-Logo-Bitmap
Type: string
Default Value: "graph-logo.32x32"
Scopes: view object view-style global
Bitmap that specifies the view style logo: appears at next to panel logo.

Panel-View-logo-Fg-Color
Type: dotted pair of strings
Default Value: ("black", "black")
Scopes: view object view-style global
Foreground color for view style logo in panel.

Pause-Ticks
Type: integer
Default Value: 50000
Scopes: global
Constant multiplier for Pause command.

Prefix-Arg-Multiplier
Type: integer
Default Value: 4
Scopes: view object view-style global
The multiplier value for prefix arguments.

Print-Stats-On-Termination
Type: boolean
Default Value: nil
Scopes: global
Print statistics information on termination of Pan.

Gc-Mouse-Icon
Type: dotted pair of strings
Default Value: ("box_spiral", "box_spiral")
Scopes: view object view-style global
The mouse icon displayed during garbage collection.
Reticle-Outline-Color
Type: dotted pair of strings
Default Value: ("dark orange" . "black")
Scopes: view object view-style global
Color to use to show position of reticle.

Scratch-View-At-Startup
Type: boolean
Default Value: nil
Scopes: global
If non-nil, create an empty scratch view at editor startup.

Text-Fill
Type: boolean
Default Value: nil
Scopes: view object view-style global
Controls automatic line filling in text.

Text-Line-Length
Type: integer
Default Value: 65
Scopes: view object view-style global
Max line length for text filling.

Text-Window-Bg-Colormap
Type: dotted pair of strings
Default Value: ("white" . "white")
Scopes: view object view-style global
Non-empty list of color specifications for text rendering background shading, where each specification is a dotted pair of color names, the first to be used when performing color rendering, the second to be used when performing monochrome rendering. Slot positions specify colors for (0) background 1, (1) background 2, (2) backgrounds 1 and 2 together. Due to a text viewport implementation restriction, text with null background is always white. Unspecified slots default to the first.

Text-Window-Col-Popover
Type: integer
Default Value: 16
Scopes: view object view-style global
Minimum characters to move window when scrolling left/right.

Text-Window-Empty-Line-Char
Type: character
Default Value: \.
Scopes: view object view-style global
Character prefix for displaying empty lines in a text window.
Text-Window-Fg-Colormap
Type: dotted pair of strings
Default Value: ("black", "black")
Scopes: view object view-style global
Non-empty list of color specifications for text rendering ink colors, where each specification is a dotted pair of color names, the first to be used when performing color rendering, the second to be used when performing monochrome rendering. Slot positions specify ink colors 0 through 3. Unspecified slots default to the first.

Text-Window-Fontmap
Type: list
Default Value: ("fixed")
Scopes: view object view-style global
List of font specifications, where each font specification is either a single font name or a dotted pair of font names.

Text-Window-Init-Cols
Type: integer
Default Value: 78
Scopes: view object view-style global
Number of columns in a text window when created.

Text-Window-Init-Rows
Type: integer
Default Value: 30
Scopes: view object view-style global
Number of rows in a text window when created.

Text-Window-Proportional-Scroll
Type: boolean
Default Value: t
Scopes: view object view-style global
If T, vertical scroll moves proportional to distance between mouse and top of scrollbar, else fixed screenful at a time.

Text-Window-Row-Popup
Type: integer
Default Value: 4
Scopes: view object view-style global
Minimum number rows to move window when scrolling up/down.

Text-Window-Tabwidth
Type: integer
Default Value: 8
Scopes: view object view-style global
Number of characters to display for a tab character.
Think-Mouse-Icon
Type: dotted pair of strings
Default Value: ("watch", "watch")
Scopes: view object view-style global
Think mouse icon.

Undo-Size
Type: integer
Default Value: 20
Scopes: object global
The size of the undo history.

Window-Alert-Style
Type:
Default Value: :bell
Scopes: view object view-style global
Style for alerting user. The only style currently supported is :bell.

Window-Bg-Color
Type: dotted pair of strings
Default Value: ("alice blue", "white")
Scopes: view object view-style global
Default background color of window.

Window-Close-With-Icon
Type: boolean
Default Value: t
Scopes: view object view-style global
Determines if an icon is created when the window is closed.

Window-Default-Mouse-Icon
Type: dotted pair of strings
Default Value: ("left.ptr", "left.ptr")
Scopes: view object view-style global
Specifies the default mouse icon.

Window-Icon-Name
Type: string
Default Value: "Untitled"
Scopes: view object view-style global
Title of icon representing the window.

Window-Initial-Pixel-Height
Type: integer
Default Value: 500
Scopes: view object view-style global
Initial height, in pixels, of window.
Window-Initial-Pixel-Width
Type: integer
Default Value: 500
Scopes: view object view-style global
Initial width, in pixels, of window.

Window-Initial-X-Position
Type: integer
Default Value: 0
Scopes: view object view-style global
Initial X position of top left hand corner of window.

Window-Initial-Y-Position
Type: integer
Default Value: 0
Scopes: view object view-style global
Initial Y position of top left hand corner of window.

Window-Message-History-Size
Type: integer
Default Value: 32
Scopes: view object view-style global
The number of window messages saved for reviewing.

Window-Name
Type: string
Default Value: "Untitled"
Scopes: view object view-style global
Title of window.

Window-Use-Color
Type: boolean
Default Value: t
Scopes: view object view-style global
If operating on a color monitor, then setting Window-Use-Color to t specifies that Pan is to use color settings. If a monochrome monitor is being used, this option variable has no effect.

Zero-Index-Lines
Type: boolean
Default Value: nil
Scopes: view object view-style global
If T, arguments to Goto-Line are interpreted as 0-indexed, else 1-indexed.
E  Flags Defined in *Pan*

- **Database-Updated?**
  - Presentation: "D"
  - Behavior: *gray when cleared*
  - Maps: On: `D.12x12`
    - Off: `gray-D.12x12`
  - Set when semantics of tree have been analyzed and database updated.

- **Object-Modified?**
  - Presentation: "x"
  - Behavior: *invisible when cleared*
  - Maps: On: `asterisk.12x12[firebrick]`
    - Off: `no map`
  - True when the current object is modified.

- **Object-Protected?**
  - Presentation: "Θ"
  - Behavior: *invisible when cleared*
  - Maps: On: `read-only.12x12`
    - Off: `no map`
  - True when the current object is protected.

- **Parse-Is-Current?**
  - Presentation: "\""
  - Behavior: *gray when cleared*
  - Maps: On: `tree.12x12`
    - Off: `gray-tree.12x12`
  - Set when language file has been parsed.

- **Semantic-Errors-Present?**
  - Presentation: "#"
  - Behavior: *invisible when cleared*
  - Maps: On: `crosshatch.12x12`
    - Off: `no map`
  - Nil if there are no semantic errors, the number of unsatisfied constraints if there are.

- **Syntax-Errors-Present?**
  - Presentation: "!"
  - Behavior: *invisible when cleared*
  - Maps: On: `exclamation-point.12x12`
    - Off: `no map`
  - Nil if there are no parse errors, the number of error nodes if there are.

**Text-Fill**
- Presentation: "\""
- Behavior: *invisible when cleared*
- Maps: On: `text-fill.12x12`
  - Off: `no map`
- Controls automatic line filling in text.
F  Character Sets in a Text View

filename-terminators
Elements: (Tab)(Newline)(Space)\nCharacters that terminate filenames in output from "ls".

identifier-characters
Elements: 0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz
Identifier characters.

indentation-characters
Elements: (Tab)(Space)
Indentation characters, a subset of whitespace-characters.

match-characters
Elements: ()[]{}
The characters that are part of match pairs.

whitespace-characters
Elements: (Tab)(Newline)(Space)
Whitespace characters.

word-characters
Elements: 0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz
Word characters.
G View-Styles

base-view-style
The View List style.

database-view-style
The Colander database view style.

dired-view-style
The Directory Editor view style.

help-view-style
The Help View style.

lex-data-view-style
The Lexical stream internal data view

node-sem-view-style
The node semantics view style.

null-view-style
A null view style for the View List class.

program-subgraph-style
Graphically presents a subtree of the internal tree representation for programs

program-graph-style
Graphically presents the internal tree representation of programs

subtree-data-view-style
The internal data for the subtree representation view

text-edit-view-style
The view style for ordinary text editing.

tree-data-view-style
The internal data for the tree representation view

worklist-view-style
The Colander worklist view style.
Index

activate view 2
active window 2
:add 54
add-variable-notifier 57, 58
analysis
incremental 12, 36
Analyze 36, 74
Analyze-Changes 36
Announce-Variable-Option 10, 52, 53
Announce-Structure-Cursor-Info 38, 40
Announce-Structure-Cursor-Node-Rule 38
Announce-Structure-Cursor-Node-Rule 40
Announce-Text-Cursor-Info 22
Any Structure Level 40, 41
Append Selection To ... 19, 72, 73
Append-Selection-To-File 19
Apropos- 16
Apropos 16
Apropos 16, 72, 73, 74
Apropos-Documentation-Sub-Type-List 77
Apropos-String 16
argument
numeric 13
text 13, 14
tree node 39
Arrow-Mouse-Icon 77
Auto-Exec 17, 38, 46, 47, 60
Auto-Load 17, 38, 46, 47, 60
Auto-Show-Match 33, 77
Auto-Show-Match-Limit 77
Autowrap-Search 32, 77
backup
revert 20
backup 19, 20
Backup-Object 19
Backup-Object? 20, 77
Backward-Expr 33
Backward-Vscroll-Current-Window 20, 21
balanced bracket expression 33
base-view-style 91
binding 2, 48, 7
bindings
scope 48
typographic format 12
Bind-Key 48
Bind-Menu 50
Bind-Oplevel-Command-List 51
Bind-Oplevel-Menu 51, 52
break loop 13, 22
:buffer 7
By Command 16, 51, 72, 73
By Level 16, 51, 72, 73
cancel 14
Capitalize-Selection 31
Capitalize-Word 31
case
change 31
case sensitive 43
Case-Sensitive-Lexeme-Search 43, 77
center 27
Center-Line 27
Char. Sets 16, 72, 73
character set
alter 54
create 54
character set 10, 33, 54
character sets 2
Character Sets 54
Character operand level 9, 22-26, 28, 29, 32, 40, 64, 75
checkpoint
revert 20
checkpoint 20
Checkpoint-Modification-Interval 20, 77
Checkpoint-Object 19
Clear-Structure-Cursor 38, 40
clipboard
copy 30
cut 30
numeric argument 30
paste 30
replace 30
size 30
viewing 30
clipboard 16
clipboard 30
Clipboard 30
clipboard 31, 43, 6
Clipboard 72, 73
Clipboard-Max-Size 6, 30, 78
Close 20, 72, 73
Close-Current-Window 20
INDEX

c-mode 46

c-mode.cl 46

c-mode.fasl 46
.
col 11

Colander 1, 11, 12, 34, 36, 37, 44, 45, 46, 65, 66

color representation 54

command 13

cancel 14

effect 8, 13

by name 13

by operand level 9

package 47

undo 15

Command 16, 72, 73

command arguments 13

Command 12

Commands Available 16, 72, 73

comment 37

configuration 4, 8, 11, 45

language-based 46

consistent information 37

constant 37

:cont 13

contextual constraints 12, 37

control character insertion 27

Copy To Cursor 31, 72, 73

Copy 30, 72, 73

Copy-Menu 50

Copy-Selection-As-Kill 29, 30

Copy-Selection-To-Cursor 30, 31

Copy-To-Clipboard 30, 49

Create-Menu 50

current operand level 9, 25

cursor 22

effects of scrolling 21

information 22

location 24

motion 24, 33, 41

move to line 24

cursor motion 5, 25

numeric argument 23

Cursor-Backward 9, 23, 41

Cursor-Forward 8, 9, 23, 41, 62, 64

Cursor-In 9, 23, 41

Cursor-Out 9, 23, 41

Cursor-Search-Backward 9, 32

Cursor-Search-Forward 9, 32, 43

Cursor-To-First 9, 23, 41

Cursor-To-Last 9, 24, 41

Cursor-To-Mouse

Cursor-To-Mouse 9, 22, 26, 40, 49, 75

Cursor-To-Token 38, 40

customization 45

Cut 30, 72, 73

cut-and-paste operations 6

Cut-To-Clipboard 30, 49

Cycle and Show 30, 72, 73

Cycle-Clipboard 30

Cycle-Kill 29, 30

Cycle-Paste 30

Cycle-Paste-Clipboard 30

Cycle-Show-Clipboard 30

Cycle-Show-Kill 29, 30

Cycle-Yank 29, 30

Database-Updated? 36, 37, 65, 89

database-view-style 91

Debug 16

debugging

   graphical views 44

   navigation commands 45

textual views 44

Declaring 10, 38

default configuration 7

default font 11, 37

Default Menu 16, 50, 72, 73

default menu 8

default scope 47

Default-Line-Style 78

Define-Command 60

Define-Hook 57, 62

Define-Hook-Function 57, 58, 62

Define-Option-Variable 57, 64

Define-Variable 57, 64

Delete marked files 18, 74

delete 25, 29

Delete 28

:delete 54

Delete 8, 9, 62, 64

Delete-Blank-Lines 28, 29

Delete-Character 28, 29, 48, 75

Delete-Fullword 28, 29, 75

Delete-Horizontal-Space 28, 29

Delete-Indentation 28, 29

Delete-Lexeme 76

Delete-Line 28, 75

Delete-Op-level-Node 76
Delete-Previous-Character 28, 29, 49
Delete-Previous-Word 28, 29
Delete-Region-Dot-To-Mark 28, 29
Delete-Selected-Region 28, 29
Delete-Word 28, 29
deletion
recover 29
Demo 78
derived information 37
Describe 16, 74
Describe-Selection 16
deselect 5
Deselect-Region 26, 27, 75
directory
edit 18, 19
directory editor 18, 19
directory 18, 19
dired 18, 19
Dired 18, 19, 74
Dired-Delete-Files 18, 19
Dired-Ls-Flags 78
Dired-Mark-Editor-Files 19
Dired-Mark-Selection 18, 19
Dired-Unmark-Selection 18
dired-view-style 4, 47, 91
Dired-Visit 18, 19
Dired-Visit-Read-Only 18, 19
Display All 16, 48, 72, 73
Display 17
Display-All-Key-Bindings 16, 48
Display-Apropos-Index 16
Display-Auto-Exec 16, 46
Display-Auto-Load 16, 46
Display-Bug-Report-Info 17
Display-Char-Set-Documentation 17, 54
Display-Char-Sets 10, 16, 54
Display-Command-Documentation 17
Display-Commands 16
Display-Configuration 17
Display-Constant-Documentation 17
Display-Default-Menu 16, 50, 8
Display-File-Ids 17
Display-Flags 16, 55
Display-Function-Documentation 17
Display-Hook-Documentation 17, 57
Display-Hooks 17, 57
Display-Key-Binding 16
Display-Key-Bindings-For-Command 16
Display-Language-List 16, 46
Display-Macro-Documentation 17
Display-Notifiers 17, 57
Display-Objects 16
Display-Operand-Command-Bindings 16, 51
Display-Operand-Command-Documentation 17, 51
Display-Operand-Level-Bindings 16, 51
Display-Operand-Level-Documentation 17, 51
Display-Operand-Level-Menus 16, 51
Display-Option-Documentation 17, 51, 53
Display-Option-Variables 10
Display-Option-Variables 16, 52, 17
Display-Version 16
Display-View-Class-Key-Bindings 17
-Documentation 17
Document-Symbol 16, 50
Dot 61
Dot-To-Mark 24
edit
protected 19, 20
edit window 4
editing area 3, 4
editing context 7
editing interface 11
Editor-Error 14
Editor-File-Search-List 46, 78
Editor-Files-Pattern 78
End-Of-Word 24
Enlarge Global Image 33, 74
error
highlighting 37, 42
lexical 42
semantic 42
syntax 42
error handling 12
Execute-Command 13, 48, 56, 57
Execute-Lisp 56, 57
Execute-Lisp-Line 48
Execute-Menu 13, 49, 8
Execute-Oplevel-Menu 51, 52, 75, 76
exit Pan 13, 22
Exit-Editor 13, 19, 74
expression
kill 29
selection 27
Expression 10, 38, 39, 40, 41, 76
expression 54
Fast-Draw-Color 78
file
auto-load 46
backup 19. 20
delete 19
edit 18. 19
insert 27
load 46, 57
mark 19
revert 20
File 19. 27. 72. 73. 73
Filename Auto-Exec Map 16, 46. 72. 73
Filename Auto-Load Map 16, 46. 72. 73
filename-terminators 90
fill 28
multiple lines 28
numeric argument 28
prefix 28
filling text 28
Fill-Selected-Lines 28
First-Lexeme 41. 76
First-Non-Blank 24
First-Oplevel-Node 41. 76
flag is set 10
flag 2. 10
Flags 16. 55. 72. 73
font 37
font code 11. 37
font map 11. 53
scope 11
formal language 34
Forward-Expr 33
Forward-Vscroll-Current-Window 20. 21
-Full-Line 27
-Fullword 27
function keys 8
Gc-Mouse-Icon 84
generic command 25
generic operations 8
global bindings 7
global image
   resize 6. 34
toggle 6. 34
global image 6. 34
global values 17
:global 7. 8. 10. 47. 48. 53
Global 52. 72. 73
Global-Image-Enlarge 33. 34
Global-Image-Initial-Height 79
Global-Image-Initial-On? 79
Global-Image-Initial-Width 79
Global-Image-Outline-Color 79
Global-Image-Shrink 33. 34
Global-Image-Toggle 33. 34
Goto Declaration 74
Goto-Line 24
graph node
   contents 33
graphical representation 44
graphical view
   mouse buttons 34
   navigation 6
   scroll 34
   subgraph 6
   zoom 34
graphical view 1. 6. 33. 44
graphical window 4
Graphical-Tree-Hbox-Background-Color 79
Graphical-Tree-Hbox-Background-Color 79
Graphical-Tree-Hbox-Border-Color 79
Graphical-Tree-Hbox-Border-Style 79
Graphical-Tree-Hbox-Border-Thickness 80
Graphical-Tree-Hbox-Text-Color 80
Graphical-Tree-Horiz-Node-Spacing 80
Graphical-Tree-Line-Arrow-Filled? 80
Graphical-Tree-Line-Arrow-Length 80
Graphical-Tree-Line-Arrow-Width 80
Graphical-Tree-Line-Color 80
Graphical-Tree-Line-Style 80
Graphical-Tree-Line-Thickness 81
Graphical-Tree-Node-Border 81
Graphical-Tree-Node-Type 81
Graph-Window-Zoom-Factor 33. 79
*: [hc] 46
help 13
Help 15, 16. 17. 46. 48. 50. 51. 52. 54. 55. 72. 73
help commands 16
help view 2. 15. 30
Help View 15. 16. 74
Help-View-At-Startup 81
help-view-style 4. 53. 91
highlight color 39. 42
Highlight-Query-Results 42. 81
Highlight-Semantic-Errors 42. 81
Highlight-Structure-Cursor 39. 42. 81
Highlight-Syntax-Errors 42. 81
hook 58
How To Report Bugs 17. 72. 73
icon 3. 20
identifier 37
identifier-characters 90
implicit selection 5. 26
INDEX

delete 29
kill 29
inconsistent 37
inconsistent information 37
indent 27
indentation
  delete 29
  indentation-characters 10
  indentation-characters 90
Inden-Like-Previous-Line 27
Indent-With-Tabs 82
information panel 3, 4, 22, 25, 40, 42
ln-Oplevel-Node 41, 76
Insert Text from... 27, 72
Insert-File 27
Insert-Newline 27
Insert-Newline-Fill 28, 49
Insert-Parentheses 27
Insert-Space-Fill 28, 49
internal state 20
internal structural representation 12
internal tree representation 6, 33, 39, 44, 45
Interrupt-Editor 13
Just-One-Space 28, 29
Key -> Command 16, 72, 73
key binding
  create 48
  scope 8
key binding 7, 48
typographic format 12
key sequence 7, 48
Keys 16, 48, 72, 73
keyword 37
kill 6, 29
  expression 33
  restore 30
Kill- 6
Kill 29
Kill 29, 72, 73
kill ring 6, 29, 43
cycle 30
  size 30
  view 30
Kill-Expr 29, 33
Kill-Previous-Word 29
Kill-Region-Dot-To-Mark 29
Killring-Max-Size 6, 30, 82
Kill-Selected-Region 29
Kills-To-Clipboard 6, 30, 62, 82

Kill-To-Eol 29
Kill-Word 29
Ladic 1, 11, 12, 34, 38, 42, 44, 45, 46, 58
language
  load 46
  operand levels 38
language definition 11, 45
language description 38
Language Descriptions Loaded 16, 46, 72, 73
language descriptions 11
language view
  flags 36
Language 35, 36, 74
language-based editing
text 35
languages
  and font use 37
Last-Lexeme 41, 76
Last-Oplevel-Node 41, 76
Left-Hscroll-Current-Window 20, 21
lex-data-view-style 91
Lexeme 38, 39, 40, 41, 43, 76
lexemes 37
Lexeme-Search-Forward 43, 76
lexical analysis 37, 42
lexical errors 42
line length 28
line number 24
line prefix 28
Line operand level 22-24, 26, 28, 32, 75
-Lisp 57
Lisp-Mouse-Icon 82
Load-File 45, 46, 56, 57
Load-From-Selection 45, 46, 56, 57
Load-From-Text-Stream 45, 46, 56, 57
Load-Verbose 82
local bindings 7
LowerCase-Selection 31
LowerCase-Word 31
mark
  set 24
Mark backup files 19, 74
mark stack 5, 6, 24, 26
Mark 18
mark 6, 24, 26
Mark 74
match characters 27, 32, 33, 54, 55, 90
menu
  scope 8
INDEX

menu bindings 8, 50
  typographic format 12
menu description 50
menu item 8, 50
  binding 8
menu selection 4, 8
Menu 51
menus 4
Menus 8
MenuTitle 12, 8
message
  review 15
message history 15
message line 2, 4, 15, 21
minor mode 47
modifier keys 7
Modula-2 24
Modula2-Src-Dir 82
Most Operand Levels 51
mouse buttons
  selection 4, 5
mouse buttons 4, 8, 21, 22, 25, 26, 34, 41
mouse 22
Mouse-Extend 9, 26, 40
Mouse-Extend-Lexeme 40, 76
Mouse-Extend-Oplevel-Node 40, 76
Mouse-Extend-Oplevel-Text 26, 75
Mouse-Extend-Selection-Full-Line 26, 75
Mouse-Extend-Selection-Fullword 26, 27
Mouse-Select-Full-Line 26
Mouse-Select-Fullword 22, 26, 27, 75
Mouse-Select-Lexeme 40, 76
Mouse-Select-Line 22, 26, 75
Mouse-Select-Oplevel-Node 40, 76
Mouse-Select-Word 26
Move To Cursor 31, 72, 73
Move-Selection-To-Cursor 31
Move-To-Bol 24
Move-To-Bos 23, 24, 75
Move-To-Eol 24
Move-To-Eos 24, 75
Move-To-First-Word 23, 75
Move-To-Last-Line 24, 75
Move-To-Last-Word 24, 75
naming conventions 12, 37
New Scratch 18, 74
Newline-And-Indent 27, 49
Next 23
Next-Character 23, 57, 64, 75, 9
Next-Lexeme 41, 76
Next-Line 14, 23, 75
Next-Line-Select 26, 27
Next-Message-Current-Window 14
Next-Oplevel-Node 41, 76
Next-Word 23, 75
node 44
Node 41, 45
node-sem-view-style 91
notifier 58
Null-Select 26, 27
null-view-style 91
Number-Of-Views-Created 57, 58
numeric prefix 13, 14
Object-Modified? 4, 10, 20, 82, 89
Object-Protected? 4, 20, 82, 89
Objects Being Viewed 16, 72, 73
Open Another 20, 72, 73
Open-Another-Window-In-View 20
Open-A-Window-In-View 20
Open-Line 27
Operand Cmds. 16, 51, 72, 73
operand level 2, 9, 22, 25, 39, 40, 41, 43
  and selection 26
  commands 25
  current 25
  delete 25, 29
  error levels 41
  language 38, 38
  menu 38, 41
  navigation 41
  search 32, 43
  select 22, 25, 26
  selection by menu 25
  selection 40
  set 25
operand level bindings 8, 25, 51
  scope 10
  command 51
  menu 52
  typographic format 12
operand level menu 25, 52
Operand Menus 16, 51, 72, 73
OperandLevel 10, 12
operand-level 2
Operand-Level-Choices 38, 64, 83
Operation 10
Olevel- 10, 41, 51
Olevel 38
Oplevel-Cursor-Backward 23, 25, 41, 45, 49
Oplevel-Cursor-Forward 9, 22, 23, 25, 40, 41, 45, 49
Oplevel-Cursor-In 23, 25, 41, 45
Oplevel-Cursor-Out 23, 25, 41, 45
Oplevel-Cursor-Search 32
Oplevel-Cursor-Search-Backward 25, 32
Oplevel-Cursor-Search-Forward 25, 31, 49
Oplevel-Cursor-To-First 23, 25, 41, 49
Oplevel-Cursor-To-Last 23, 25, 41, 49
Oplevel-Cursor-To-Mouse 22, 25, 26, 40, 41, 49
Oplevel-Delete 25, 28, 29, 43
Oplevel-In 41, 49
Oplevel-Menu 9, 25, 38, 41, 49, 51, 52, 64
Oplevel-Mouse-Extend 25, 26, 40
Oplevel-Out 41, 49
Oplevel-Query-Replace-All 25, 32
Oplevel-Replace 32
Oplevel-Replace-All 25, 32
Oplevel-Select 25, 26, 40
option
   announce 53
   define 57
   set 53
Option variables 10
option 2, 10, 53
:option 57
Options 16, 51, 52, 72, 73
Out-Oplevel-Node 41, 76
package 47
Package-Symbol-Names 48, 83
Pan 16, 17, 46, 72, 73
Panel 54
Panel-Bg-Color 53, 83
Panel-Flag-Default-Bitmap 55, 83
Panel-Flag-Default-Fg-Color 53-55, 83
Panel-Flags-Bg-Color 53, 55, 83
Panel-Logo-Bg-Color 53, 83
Panel-Logo-Bitmap 83
Panel-Logo-Fg-Color 53, 84
Panel-View-Logo-Bg-Color 54, 84
Panel-View-Logo-Bitmap 84
Panel-View-Logo-Fg-Color 54, 84
.panrc 1, 45-48, 53, 55, 57
parentheses 27
Parse-Is-Current? 36, 37, 66, 89
Paste 30, 72, 73
Paste-From-Clipboard 30, 49
pattern match 32, 43
Pause-Ticks 33, 84
Perform-Hook 57, 62
Pan 21
Placeholder 38
Pop-Mark 24
pop-up window 14
prefix
   repetition 14
prefix keys 7
Prefix 13, 14
Prefix-Arg-Multiplier 14, 84
Preserve-All-Objects 19
Previous 23
Previous-Character 23, 75
Previous-Line 23, 75
Previous-Line-Select 26, 27
Previous-Word 23, 75
Prev-Lexeme 41, 76
Prev-Message-Current-Window 14
Prev-Oplevel-Node 41, 76
Print-Stats-On-Termination 84
program
   editing
      language-based editing 34
      operand levels 38
      Program Graph 35, 44, 74
      program-graph-style 91
      program-subgraph-style 91
      prompt 13, 14
      protected 19
      Protected-Visit-File 17
      < language >-mode 46
      < language >-mode.cl 12, 46
      Push-Mark 24
      query 43
      Query 38, 43
      Query-Replace-All 9, 15, 32, 43, 75
      Query-Replace-Lexemes 43, 76
      Quit 13
      quitting Pan 13
      Quote-Character 27
      Quote-Insert 27
      reanalysis 37, 44
      Rebind-Menu 50, 51
      Redo 15
      Redo 15, 72, 73
      Redraw 20, 72, 73
      Redraw-Current-Window 20
      reduction rule 40
      region 5
semantic information 12
Semantic-Errors-Present? 36, 42, 89
Set Option 52, 72, 73
Set- 53
Set-Buffer-Option-Variable 52
Set-Char-Match 54, 55
Set-Char-Set 54
Set-Global-Option-Variable 52
Set-Mark 24
Set-Oplevel- 38
Set-Oplevel-To-Character 24, 64
Set-Oplevel-To-Declaration 64
Set-Oplevel-To-Lexeme 38
Set-Oplevel-To-Line 24
Set-Oplevel-To-Node 38
Set-Oplevel-To-Query 38
Set-Oplevel-To-Semantic-Error 38
Set-Oplevel-To-Syntactic-Error 38
Set-Oplevel-To-Word 24
Set-Text-Fill-Prefix 28
Set-Text-Line-Length 28
Set-View-Option-Prefix 52
Set-View-Style-Option-Variable 52
Set-Working-Directory 18
shadow 48
:shadow 54
Show Contents 30, 72, 73
Show Declaration and Uses 74
Show-Clipboard 30
Show-Kill 29, 30
Show-Match 33
Shrink Global Image 33, 74
Split-Line 27
starting Pan 13
start-up 45
Statement 10, 38, 40
Store 17, 19, 72, 73
structural annotations 12
structural navigation 41
structural cursor 6, 39, 40, 41
and selection 39
clear 40
set 39
set 40
structure node 40
Subgraph At Cursor 35, 44, 74
subgraph 44
Sub-Redo 15
Sub-Redo 15, 72, 73
subtree 6
subtree-data-view-style 91
Sub-Undo 15, 32
Sub-Undo 15, 72, 73
Swap-Dot-And-Mark 24
Syntactic analysis 37
Syntactic Error 38
syntactic errors 12, 42
syntax analysis 42
syntax description 11
Syntax Error 41
Syntax-Errors-Present? 36, 42, 89
.tex 18
tex-mode 47
tex-mode-cl 47
text
  copy 30, 31
delete 29, 30
editing 22
fill 28, 3
kill 29
move 30, 31
replace 30
transposing 31
yank 30
Text 50
text color 42, 43
text cursor 2, 3, 5, 26
Text Edit 19, 27, 29, 31, 72, 73
text insertion
  numeric prefix 27
text insertion 27
text stream
  load 57
text view 1
text window 4
text-based editing 4
text-edit-view-style 4, 46, 91
Text-Fill 3, 4, 28, 85, 89
Text-Fill-To-Blank-Line 28
Text-Line-Length 27, 28, 61, 85
textual argument 13
textual insertion 5
textual selection 26
textual view 4, 22, 26, 46
Text-Window-Bg-Colormap 39, 42, 53, 54, 85
Text-Window-Col-Popover 85
Text-Window-Empty-Line-Char 85
Text-Window-Fg-Colormap 42, 43, 53, 54, 86
subgraph 44
  text 4, 22
  visit 18
  write 19
View 18, 35, 44, 72, 73, 74
view list 2, 3, 13, 18, 20
View List 13, 17, 18, 74
view name 2
View Only 52, 72, 73
View Style 52, 72, 73
view style 4, 8
:view-style 7, 8, 10, 47, 48, 53
visit 19
Visit 18, 74
Visit Definition 16, 74
Visit New... 17, 72, 73, 74
Visit Read Only 18, 74
Visit View 17, 18, 74
Visit-And-Check-File 36, 37
Visit-Backup 17, 60
Visit-Checkpoint 17, 20, 60
Visit-Database-View 44, 45
Visit-Directory 18, 19
Visit-File 17, 18, 37, 60
Visit-Help-View 15, 16, 49
Visit-Lexeme-Internal-Data-View 44
Visit-Node-Semantics-View 44, 45
Visit-Program-Graph- 44
Visit-Program-Graph-Address-View 44
Visit-Program-Graph-Context-View 44
Visit-Program-Graph-Operator-View 44
Visit-Program-Graph-Parse-Change-View 44
Visit-Program-Graph-Parse-Status-View 44
Visit-Program-Graph-View 44
Visit-Program-Subgraph- 39, 44
Visit-Program-Subgraph-Address-View 44
Visit-Program-Subgraph-Context-View 44
Visit-Program-Subgraph-Operator-View 44
Visit-Program-Subgraph-Parse-Change-View 44
Visit-Program-Subgraph-Parse-Status-View 44
Visit-Program-Subgraph-View 44
Visit-Selected-View 17, 18
Visit-Subtree-Internal-Data-View 44, 45
Visit-Symbol-Definition 16
Visit-Tree-Internal-Data-View 44, 45
Visit-View 18
Visit-View-List 18, 49
Visit-Worklist-View 44, 45
white space
delete 29

**whitespace-characters** 90

window 2, 17, 20
close 3, 20
redraw 20, 21
reopen 3, 20

Window 5, 20, 21, 33, 72, 73, 74
window manager 3, 20
Window-Alert-Style 87
Window-Bg-Color 87
Window-Close-Icon 3, 20, 53, 87
Window-Default-Mouse-Icon 63, 87
Window-Icon-Name 87
Window-Initial-Height 87, 88
Window-Initial-Position 88
Window-Initial-Y-Position 88
Window-Message-History-Size 15, 88
Window-Modify-Flag-Collection 55
Window-Name 88

windows
  multiple 20
Window-Use-Color 54, 88
With Node Names 44, 74
With Operators 44, 74
With-Buffer-Style 47
With-View-Icon 47
With-View-Style-Icon 47, 53

Word operand level 22-24, 26, 28, 32, 52, 75

word-characters 10, 54, 90
word-oplevel-menu 52
worklist-view-style 91
write protection 20
Write Selection To ... 19, 72, 73
Write-Selection-To-File 19

yank 6

Yank 29, 72, 73
Yank-From-Kill-Ring 29, 30
Zero-Index-Lines 24, 88
zoom 33
  numeric argument 34

**Zoom All** 33, 74
Zoom In 33, 74
Zoom Out 33, 74
Zoom-6 33
Zoom-In 33
Zoom-Out 33
Zoom-Reset 33