

TABLEOFCONTENTS

~)19 ~'~

1. A Brief Glossary t 1.1	
2. The Mouse and the Keyboard / 2.1	
2.1. The Mouse 2.1	
2.1.1. 2and3ButtonMice 2.1	
2.2. _ The _ Keyhoard _ 2.2	
2.2.1. _ The _ 1186 _ Keyhoard _ 2.2	
2.2.2. _ The _ 1108 _ Keyhoard _ 2.2	
3. Turning On Your Lisp Machine / 3.1	
3.1. _ Turningonthello8 _ 3.1	
3.2. _ Turning on _ the1186 _ 3.2	
3.3. _ Loading _ Intertis~D from the _ Hard _ Disk _ 3.3	
3.4. _ AfterBooting _ Lisp _ 3.5	
3.5. _ Restarting _ Lisp After Logging _ Out _ 3.5	
4. If You Have a Fileserver / 4.1	
4.1. _ Turning on your 1108 _ 4.1	
4.2. _ Turningonyourll86 _ 4.1	
4.3. Location of Files 4.2	
4.4. The Timeserver 4.2	
5. Logging Out And Turning the Machine Off 51	
5.1. _ Logging Out _ 5.1	
5.2. _ Turning _ The _ MachineOff _ 5.2	
6. Typing Shortcuts 6.1	
6.1. _ If you _ makea _ Mistake _ 6.3	
7. Using Menus 7.1	
7.1. _ Making _ a _ Selection from _ a _ Menu _ 7.2	
7.2. _ Explanations _ of _ Menu _ Items _ 7.2	
7.3. Submenus 7.3	
8. How to use Files 8.1	
8.1. _ Types of Files _ 8.1	

TA8~ OF CONTENTt To'.,
1

.-----.-.-.-.-

TABLE OF CONTENTr

8.2. _ Directories _ 8.1	
8.3. _ Directory Options _ 8.2	
8.4. _ Subfile _ Directories _ 8.3	
8.5. _ To See What Filri Are _ Loaded _ 8.3	
8.6. _ Simple Commands for Manipulating _ Filu _ 8.3	
\	
8.7. _ to a _ 8.4	
- '._~y Connecting _ Directory	
\ s 8.8. _ File Vettion _ Numbers _ 8.4	
9. FileBrowser 9.1	
9.1. _ Calling the _ FileSrowser _ 9.1	
9.2. FileBrowserCommands 9.3	
10. ffose Wondertul Windows! 10.1	
10.1. _ Windows _ provided _ by Interlis~D _ 10.1	

- 10.2. _ Creating a _ window _ 10.2
- 10.3. _ The Right _ Button DefaultWindow _ Menu _ 10.2
- 10.4. _ An _ explanation of each _ menu _ item _ 10.3
- 10.5. _ krollable Windows _ 10.3
- 10.6. Other Window Functions 10.5
- 10.6.1. PROMTPRINT 10.5
- 10.6.2. _ WHICHW _ 10.6
- 11. Editing and Saving 11.1
- 11.1. _ Defining _ Functions _ 11.1
- 11.2. Simple Editing in the Interlis~D Executive Window 11.2

YA8~ OF cottnritt

- iA.3. _ Wøys to Stop Exøcution from thø Køyboard, called _ 1ørøøhng LIzp5 _ 14.3
- t4.4. _ Programming _ Brøaks and Døbugging Codø _ 14.4
- 14.5. Break Monu 14.4
- 14.6. _ Returning to Top Lovøl _ '4.5
- 15. _ On-Line _ Help _ with _ Interlisp-D: _ HELPSYS and _ DINFO _ ~ _ 15.1
- 15.1. _ HelpSys _ 15.1
- 15.2. DInfo 15.1
- 16. Floppy Disks / 16.1
- 16.1. _ Buying Floppy Disks _ 16.1
- 16.2. _ Basic Floppy Disk Information _ 16.1
- 16.3. _ Care of Floppies _ 16.2
- 16.4. _ Write Enabling and Write Protecting _ Floppies _ 16.3
- 16.4.1. _ Write Enabling an _ 1108's _ Floppy _ Disk _ 16.3
- 16.4.2. _ Write Protecting an _ 1186's Floppy Disk _ 16.3
- 16.5. _ Inserting _ Floppies _ into the _ Floppy Drive _ 16.3
- 16.6. _ Functions for Floppy Disks _ 16.4
- 16.6.1. _ Formatting Floppies _ 16.4
- 16.6.2. _ Available Space on a Floppy Disk _ 16.4
- 16.6.3. _ The Name of a Floppy Disk _ 16.4
- 16.6.4. _ FLOPPY.MODE _ 16.5

- 17. Duplicating Floppy Disks 17.1
- 17.1. _ Supplies _ 17.1
- 17.2. _ Preparabon _ 17.1
- 17.2.1. _ Handling _ Floppy _ Disks _ 17.1
- 17.2.2. _ Setup _ 17.1
- 17.3. _ Copying _ Floppy Disks _ 17.2
- 18. Sysout Files 18.1
- 18.1. _ Loading SYSOUT Filri _ 18.1
- 18.1.1. _ Loading a _ SYSOUTfile on the _ 1108 _ 18.1
- 18.1.2. _ Loading a SYSOuTfile on the _ 1186 _ 18.2
- 18.2. _ Making _ Your Own SYSOUT File _ 18.3
- 19. Using the Epson FX80 Printer ~ 19.1
- 19.1. _ Initializing the RS232 Port _ 19.1
- 19.2. _ Power up the Printer _ 19.1
- 19.3. _ to Align Top of Page _ 19.1

YA8~ OF CONTENTS TOC.3

TABLE OF CONTENTD yl

- 19.4. _ Fundions To Print Filri and _ Bitmapf _ 19.2
- 19.4.1. _ RS232.Print _ 19.2
- 19.4.2. _ FXWSTREAM _ 19.2
- 19.4.3. _ Printing a Portion of the Screen _ 19.3
- 20. RS232 File Transfer With a VAX 20.1
- 20.1. _ Prerequisites _ 20.1
- 20.2. _ Using Chat to Transfer Filri _ 20.1
- 21. Ethernet File Transfer 21.1
- 21.1. _ Prerequisites _ 21.1

21.2. File Transfer 21.1

22. WhatToDoDolf...

22.1

ø;;23. The Text žditor,TEdit 23.1

23.1. _ Using TEdit _ 23.1

23.2. _ Managing the edit Window _ 23.2

23.3. _ Seleding _ Tert _ 23.3

23.4. _ Deleting, Copying, and _ Moving Text with edit _ 23.4

23.4.1. _ Deleting Text From a File _ 23.4

23.4.2. _ Copying _ Text _ 23.4

23.4.3. _ Moving _ Text _ 23.5

23.5. _ rtdit Menus _ 23.6

23.5.1. _ Finding _ and _ Substituting Text with _ edit _ 23.7

23.5.1.1. _ Finding Text _ 23.7

23.5.1.2. _ Substituting Text _ 23.8

23.5.L _ Text Formatting _ 23.10

23.5.2.1. _ Choosing Fonts _ 23.10

23.5.2.2. _ Paragraph _ Formatting _ 23.11

23.5.3. _ Adding _ Bitmaps and Sketches to your TEdit File _ 23.13

23.5.3.t. _ Adding a _ Bitmap to your TEdit file _ 23.13

23.5.3.2. _ Adding a Sketch to your TEdit file _ 23.14

23.5.4. _ Getting and _ Including _ Filri _ 23.14

23.5.4.1. _ Get _ 23.14

23.5.4.2. _ Include _ 23.14

23.5.5. _ Saving and Printing Files _ 23.15

24. _ Records _ May _ BG _ Your _ Favorite _ Data _ Structure! _ 24.1

2&1. _ Interlisp Record ~imitlves _ 24.1

T~.. rAa~0fC0NTENrt

TABLE OF OoNTENfl

24.2. _ Exomplo _ 24.3

24.3. _ AFøwflps _ 24.4

25. Local Variables - Using LET and PROG 7 25.1

25.1. _ LET _ 25.1

25.2. _ PfloG _ 25.3

25.3. _ PorilloI _ vottus _ S~uential _ Vorioblo _ Binding _ 25.6

25.3.1. _ L~ _ 25.6

25.3.2. _ PROGø _ 25.7

26. Iterative statements 26.1

26.1. _ General _ Strudurc and _ Use _ 26.1

26.2. _ Local _ Variables _ 26.2

26.3. _ Iteration _ On _ Lists _ 26.3

26.4. _ Parallel _ Iteration _ 26.4

26.5. _ Conditional _ Iteration _ 26.5

26.6. _ More _ Iteration _ 26.6

27. Window and Regions 27.1

27.1. _ Windows _ 27.1

27.1.1. _ CREATEW _ 27.1

27.1.2. _ WINDOWPROP _ 27.2

27.1.3. _ Getting windows to do things _ 27.3

27.1.3.1. _ BUflONEVENTFN _ 27.4

27.1.4. _ Looking at a _ window's properties _ 27.5

27.2. _ Regions _ 27.5

28. What Are Menus? 28.1

28.1. _ Displaying _ Menus _ 28.1

28.2. _ Getting _ Menus to DO Stuff _ 28.2

28.2.1. The WHENHELDfH and WHENSELE~DFN fields of a

menu 28.4

28.3. _ Looking _ at a _ menu's fields _ 28.5

29. Bitmaps 29.1

30. Displaystreams 30.1

30.1. _ Drawing _ on a _ Displaystream _ 30.1

30.1.1. _ DliWUNE _ 30.1

30.1.2. _ DliWTO _ 30.2

30.1.3. _ DliWaRCLE _ 30.3

TABS OF CONTENTff TOC.5

I

TABS OF CON~Nfi

30.1.3.1. _ FILLGRCLE _ 30.3

30.2. _ Locating and _ Changing _ Your Position _ in _ a _ Displaystream _ 30.4

30.2.1. _ DSPXP0SIfION _ 30.5

30.2.2. _ DSPYPOSIBON _ 30.5

30.2.3. _ MOVETO _ 30.5

31. Fonts 31.1

31.1. _ WhatmakesupaFONn _ 31.1

31.2. _ Fontdescriptors, and _ FONTCREATE _ 31.2

31.3. _ Display Fonts-Theirfiles, and how to find them _ 31.3

31.4. _ Interpress _ Fonts- Their files, and _ how to find them _ 31.4

31.5. _ Functions for Using Fonts _ 31.4

31.5.1. _ FOHTPROP - _ Looking at Font Properties _ 31.4

31.5.2. _ SfflINGWIDTH _ 31.5

31.5.3. _ DSPFONT- Changing the Font in _ One Window _ 31.6

31.5.4. _ Glo~IlyChanging Fonts _ 31.7

31.5.5. _ Pettonalizing _ Your Font Profile _ 31.7

32. The Inspetror 32.1

32.1. _ Calling the Inspector _ 32.1

32.2. _ Using _ the _ Inspector _ 32.2

32.3. _ Inspector _ Example _ 32.2

33. Masterscope 33.1

33.1. _ The SHOW DATA command and GRAPHER _ 33.2

33.2. Databasefns: Automatic Conrtuction and Upkeep of a Mastettcope

Data~se _ 33.3

34. Where Does All the Time Go? SPY 34.1

34.1. _ How to use Spy with the SAY Window _ 34.1

34.2. _ How to use _ SPY from the _ Lisp Top Level _ 34.2

34.3. _ Interpreting _ SPY's Results _ 34.2

35. SKETCH 35.1

35.1. _ Starting _ Sketch _ 35.1

35.2. _ Selecting _ Sketch elements _ 35.1

35.3. _ Drawing _ with _ Sketch _ 35.2

35.3.1. _ Simple Shapes: _ Circles, Ellipsriø and _ Boxes _ 35.3

35.3.1.1. _ Drawing _ Circlri _ 35.3

35.3.1.1 _ Elllpsri _ 35.3

TA.G TAILE0fCONFENrt ‘/‘

----- Next Message -----

Date: 19 Dec 91 14:18 PST

From: sybalsky:PARC:Xerox

To: sybalsky

Message-ID: <<91Dec19.141853pst.43009@origami.parc.xerox.com>.?::>

<----RFC822 headers-----

Received: from origami.parc.xerox.com ([13.1.100.224]) by alpha.xerox.com with SMTP id <11640>; Thu, 19 Dec 1991 14:19:05 PST

Received: by origami.parc.xerox.com id <43009>; Thu, 19 Dec 1991 14:18:53 -0800

From: John Sybalsky <sybalsky.PARC@xerox.com>

-----RFC822 headers----->

TABLE OF CONTENTS

42. Simple Interactions with the Cursor, a Bitmap, and a Window	42.1
42.1. An Example Function Using GETHOUSESTATE	42.1
42.2. Advising GETMOUSESTATE	42.2
42.3. Changing the Cursor	42.2
42.4. Functions for "Tracing the cursor"	42.3
42.5. Running the Functions	42.6
43. Glossary of Global System Variables	43.1
43.1. Directories	43.1
43.2. Flags	43.2
43.3. History Lists	43.3
43.4. System Menus	43.3
43.5. Windows	43.4
43.6. Miscellaneous	43.4
44. Other References that will be Useful to You	44.1

TABLE OF CONTENTS

PREFACE

it was dawn and the locd told him it was down the road a piece,
left the fishing bridge in the country right at the apple
tree stump, and onto the dirt road just before the hill. At
midnight he knew he was lost.
-Anonymous

Welcome to the Interlisp-D programming environment! The
Interlisp-D environment truly must be one of the most
sophisticated and powerful tools in use by human beings.
Overall, it is flexible, well thought out, and full of pleasant
surprises: "Wow, here are exactly the set of functions I thought
I'd need to write." Unfortunately, along with the power comes
mind-numbing complexity. The Interlisp Reference Manual
describes the functions and some of the tools available in the
Interlisp-D environment. To do this takes three large volumes.
Other volumes are needed to document the library packages and
other newly written tools. Needless to say, it is very difficult to
learn such a huge amount of material when there is no way to
determine where to start!

We developed this primer to provide a starting point for new
Interlisp-D users, to enhance your excitement and challenge you
with the potential before you. We assume you know a little
about LISP, most likely received from taking a survey course in
Artificial Intelligence (AI), and have seen a demonstration of
how Interlisp-D runs on your 1186 or 1108. We further assume
that your machine is not on a network system with a file server -
though this is addressed, and that you will be working from
floppy disks and the hard disk that is part of the machine. If this
describes your situation, you are ready to sit down in front of
your machine and follow the step-by-step examples provided in
this primer.

The primer is broken into many small chapters, and these
chapters are organized into five parts. You may want to read

Parts 1 through 3 straight through, since they describe the basics of using the machine. Each chapter in Sections 4 and 5, however, can be used to learn a specific skill whenever you are ready to for it

Part one, "Introduction", includes Chapters 1 and 2. Part two, "Getting Into/Out of Interlisp", includes Chapters 3 through 5. Part three, "The Interlis~D language and Programming Environment", includes Chapters 6 through 15. These chapters discuss primary elements in Interlis~D, and orient you in relation to those elements. Part four, "Important Other Things to Know to Work Successfully", includes Chapters 16 through 31. Part five, "More Language and Environment and Packages", includes Chapters 32 through 44.

PREFAcE v

----- Next Message -----

Date: 19 Dec 91 14:20 PST
 From: sybalsky:PARC:Xerox
 To: sybalsky
 Message-ID: <<91Dec19.142054pst.43009@origami.parc.xerox.com>.?::>

<----RFC822 headers-----

Received: from origami.parc.xerox.com ([13.1.100.224]) by alpha.xerox.com with SMTP id <11636>; Thu, 19 Dec 1991 14:21:05 PST

Received: by origami.parc.xerox.com id <43009>; Thu, 19 Dec 1991 14:20:54 -0800

From: John Sybalsky <sybalsky.PARC@xerox.com>

-----RFC822 headers----->

PREFACE

Through out we make reference to the Interlis~D Reference Manual by section and page number. The material in the primer is just an introduction. When you need more depth use the detailed treatment provided in the manual.

While only you can plot your ultimate destination, you will find this primer indispensable for clearly defining and guiding you to the first landmarks on your way.

Acknowledgements The early inspiration and model for this primer came from the Intelligent Tutoring Systems group and the Learning Research and Development Center at the University of Pittsburgh. We gratefully acknowledge their pioneering contribution to more effective artificial intelligence.

This primer was developed by Computer Possibilities, a company committed to making AI technology available. Primary development and writing was done by Cynthia Cosic, with technical writing support provided by Sam Zordich.

At Xerox Artificial Intelligence Systems, John Vittal managed and directed the project. Substantial assistance was provided by many members of the AIS staff who provided both editorial and systems support.

PREFAcE ~ ø1

----- Next Message -----

Date: 19 Dec 91 14:33 PST
 From: sybalsky:PARC:Xerox
 To: sybalsky
 Message-ID: <<91Dec19.143340pst.43009@origami.parc.xerox.com>.?::>

<----RFC822 headers-----

Received: from origami.parc.xerox.com ([13.1.100.224]) by alpha.xerox.com with SMTP id <11653>; Thu, 19 Dec 1991 14:33:46 PST

Received: by origami.parc.xerox.com id <43009>; Thu, 19 Dec 1991 14:33:40 -0800

From: John Sybalsky <sybalsky.PARC@xerox.com>

-----RFC822 headers----->

1. ABRIEFGLOSSARY

The following definitions will acquaint you with general terms used throughout this primer. You will probably want to read through them now, and use this chapter as a reference while you read through the rest of the primer.

advising An Interlis~D facility for specifying function modifications without necessarily knowing how a particular function works or even what it does. Even system functions can be changed with advising.

argument An argument is a piece of information given to an Interlis~D function so that it can execute successfully. When a function is explained in the primer, the arguments that it requires will also be given. Arguments are also called Parametert.

atom The smallest rtrvcture in Lisp; like a variable in other programming languages, but can also have a property list and a function definition.

Background Menu The menu that appears when the mouse is not in any window and the right mouse button is pressed. A typical background menu is shown in Figure 1.1.

Loops Icon
 FileBrowser

Figure 1.1. The Menu that appears when the mouse is not in any window, and the right mouse button is pressed. Your background menu may have some different items in it

binding The value of a variable. It could be either a local or a global variable. See unbound.

bitmap A rectangular array of '0 pixels, '0 each of which is on or off representing one point in the bitmap image.

BREAK An Interlisp function that causes a function to stop executing, open a Break window, and allow the user to find out what is happening while the function is halted.

Break Window A window that opens when an error is encountered while running your program (i.e., when your program has broken).

There are tools to help you debug your program from this window. This is explained further in Chapter 14, Page 14.1.

browse To examine a data structure by use of a display that allows the user to "move" around within the data structure.

button

A BRIEF GLOSSARY 1.1

1

A BRIEF GLOSSARY

(1) (n.) A key on a mouse.

(2) (v.t.) To depress one of the mouse keys when making a selection.

EAR A function that returns the head or first element of a list. See CDR.

caret The small blinking arrowhead that marks where text will appear when it is typed in from the keyboard. An example of the caret in the Interlisp-D Executive Window is shown in Figure 1.2.

NIL

B6+(PLUS 3A

Figure 1.2. The caret is to the right of the number 3. When a character is typed at the keyboard, it will appear at the caret.

CDR A function that returns the tail (that is, everything but the first element) of a list. See EAR.

CLISP A mechanism for augmenting the standard Lisp syntax. One such augmentation included in Interlisp is the iterative statement. See Section 13.1.

cr Please press your carriage return key.
datatype

(1) The kind of a datum. In Interlisp, there are many system-defined datatypes e.g. Floating Point, Integer, Atom, etc.

(2) A datatype can also be user-defined. In this case it is like a record made up from system types and other user-defined datatypes.

DWIM Do-what-I-mean. Many errors made by Interlisp users could be corrected without any information about the purpose of the program or expression in question (e.g. misspellings, certain kinds of parenthesis error). The DWIM facility is called automatically whenever an error occurs in the evaluation of an Interlisp expression. If DWIM is able to make a correction, the computation continues as though no error had occurred; otherwise, the standard error mechanism is invoked.

error Occasionally, while a program is running, an error may occur which will stop the computation. Interlisp provides extensive facilities for detecting and handling error conditions, to enable the testing, debugging, and revising of imperfect programs.

evaluate or EVAL Means to find the value of a form. For example, if the variable X is bound to 5, we get 5 by evaluating X. Evaluation of an Interlisp function involves evaluating the arguments and then applying the function.

file package A set of functions and conventions that facilitate the bookkeeping involved with working in a large system consisting of many source code files and their compiled counterparts.

Essentially, the file package keeps track of where things are and

1,

A BRIEF GLOSSARY

1

A BRIEF GLOSSARY

When things have changed. In 4150 kbps track of which files have been modified and need to be updated and recompiled.
 form Another way of saying ~xpression. An Interlisp-D expression can be evaluated.

function A Lisp function is a piece of Lisp code that executes and returns a value.

history The programmer's assistant is built around a memory structure called the history list. The history functions (e.g. FIX, UNDO, REDO) are part of this assistant. These operations allow you to conveniently rework previously specified operations.

History List As you type on the screen, you will notice a number followed by a prompt arrow. Each number, and the information on that line, is sequentially stored as the History List Using the History List, you can easily reexecute lines typed earlier in a worksession. See Chapter 6.

icon A pictorial representation, usually of a shrunken window.

Interlisp-D Executive Window This is your main window, where you will run functions and develop your programs. See Figure 1.3. This is the window that the caret is in when you turn on your machine and load Interlisp-D.

NIL

8~#iPRO*PTPRIHT "HELLO" A

Fourth window

inspector An interactive display program for examining and changing the parts of a data structure. Interlisp-D has inspectors for lists and other data types.

iterative statement (also called i.s.) A statement in Interlisp that repetitively executes a body of code. (E.g. (forx from to do (PRINT x)) is an i.s.)

iterative variable (also called i.v.) Usually, an iterative statement is controlled by the value that the i.v. takes on. In the iterative statement example above,
 x

is the iterative variable because its value is being changed by each cycle through the loop. All iterative variables are local to the iterative statement where they are defined.

LISP Family of languages invented for "list processing." These languages have in common a set of basic primitives for creating and manipulating symbol structures. Interlisp-D is an implementation of the LISP language together with an environment (set of tools) for programming, and a set of packages that extend the functionality of the system.

list A collection of atoms and lists; a list is denoted by surrounding its contents with a pair of parentheses.

A BRIEF GLOSSARY II

1

A BRIEF GLOSSARY

Loading LJSP This is the process of bringing Interlis~D from floppy disks, hard disks, or some other secondary storage into your main, or working, memory. You will need to load (i.e., install, and boot) Interlis~D if you have not logged off the machine at the end of a session. The process of loading Interlis~D is explained in Chapter3.

Maintenance Panel Codes Should you have a problem with your equipment, these codes will indicate the status of your processor. On the 1108, these are the red LED numbers under the floppy drive door. There is a cover over these numbers. Pull down the cover located immediately under the floppy door button. The code numbers are defined for the 1108 in the 1108 User Guide, in the MP Codes chapter.

If there is a problem with the 1186, the mouse cursor will change from its normal arrow to the code number that describes the problem. The code numbers are defined for the 1186 in the 1166 User's Guide in the Cursor Codes subsection of the Diagnostics Chapter.

Masterscope A program analysis tool. When told to analyze a program, Masterscope creates a data base of information about the program. In particular, Masterscope knows which functions call other functions and which functions use which variables. Masterscope can then answer questions about the program and display the information with a browser.

menu A way of graphically presenting the user with a set of options. There are two kinds of menus: pop-up menus are created when needed and disappear after an item has been selected; permanent menus remain on the screen after use.

mouse The Mouse is the box to the right of your keyboard. It controls the movement of the cursor on your screen. As you become familiar with the mouse, you will find it much quicker to use the mouse than the keyboard. See Figure 1.4. (Note: Some mice have three buttons; the button in the center is known as the middle mouse button. If your mouse has only two buttons, you can simulate a middle button by pressing the left and right buttons simultaneously.).

Mouse 1. & Mouse

Mouse Cursor The small arrow on the screen that points to the northwest. See Figure 1.5.

Mouse Cursor Icons

Mouse Cursor Icons

A BRIEF GLOSSARY

I Wait The processor is busy.

The processor is saving a snapshot of your current system session. This is usually done when the processor has been idle for a while.

The "Mouse Confirm Cursor". It appears when you have to confirm that the choice you just made was correct. If it was, press the left button. If the choice was not right, press the right button to abort.

F=*x This means "sweep out" the shape of the window. To do this, move the mouse to a position where you want a corner. Press the left mouse button, and hold it down. Move the mouse diagonally to sketch a rectangle. When the rectangle is the desired size and shape, release the left button.

r-'l

l

l

l

l

- This is the "move window" prompt. Move the mouse so that the large "ghost" rectangle is in the position where you want the window. When you click the left mouse button, the window will appear at this new location.

NIL NIL is the Interlis~D symbol for the empty list. It can also be represented by a left paren followed by a right paren: (). It is the only expression in Interlis~D that is both an atom and a list. Pixel stands for Picture Element. The screen of your Lisp Machine is made up of a rectangular array of pixels. Each pixel corresponds to one bit. When a bit is turned on, i.e. set to 1, the pixel on the screen represented by this bit is black.

pretty printing Pretty printing refers to the way Interlis~D functions are printed with special indentation, to make them easier to read. Functions are pretty printed in the structure editor, DEdit (See Section 11.3, Page 11.4). You can pretty print uncompiled functions by calling the function PP with the function you would like to see as an argument, i.e. (PP function-name). For an example of this, see Figure 1.6.

96.(PP HEAD)

```
[LAMBDA (LST) <lambda>G; 'lambda~JlinH13;3&lambda>
(CAR LST)
(HEAD)
97.'
```

Figure 1.6. An example of the pretty printing 'unmon FP

A BRIEF GLOSSARY 1.5

I

A BRIEF GLOSSARY

Programmer's Assistant The programmer's assistant accesses the History List to allow you to FIX, UNDO, and/or REDO your previous expressions typed to the Interlis~D executive window. (See Chapter 6.)

Promptwindow The skinny black window at the top of the screen. It displays system prompts, or prompts you have developed. (See Figure 1.7.)

Figure 1.7. Prompt window

property list A list of the form (<property-name1> <property-value1> <property-name2> <property-value2>) associated with an atom. It is accessed by the functions GETPROP and PUTPROP.

record A record is a data-structure that consists of named "fields". Accessing elements of a record can be separated from the details of how the data structure is actually stored. This eliminates many programming details. A record definition establishes a record template, describing the form of a record. A record instance is an actual record storing data according to a particular record template. (See datatype, second definition.)

Right Button Default Window Menu This is the menu that appears when the mouse is in a window, and the right mouse button is pressed. It looks like the menu in Figure 1.8. If this menu does not appear when you depress the right button of the mouse and the mouse is in the window, move the mouse so that it is pointing to the title bar of the window, and press the right button.

Clone
Snap
Paint
Clear
Bury

Redisplay
Hardcopy~
Move
Shapo
Shrink

Figure 1.1. Right Button Default Window Menu

S-expression Short for "symbolic expression." In Lisp, this refers to any well-formed collection of left parens, atoms, and right parens.

stack A pushdown list Whenever a function is entered, information about that specific function call is pushed onto (i.e. added to the front of) the stack; this information includes the variable names and their values associated with the function call. When the function is exited, that data is popped off the stack.

storage devices Information is stored for your Lisp machine on floppy disks, or on the hard disk. They are referred to as (FLOPPY) and (DSK) respectively.

sysout A file containing the Lisp environment: namely, Interlisp-O, everything the user defines or loaded into the environment, the

1.6 A ~N:EF GL0SsARY
I

A BRIEF GLOSSARY

windows that appeared on the screen, the amount of memory used, and so on. Everything is stored in the sysout file exactly as it was when the function SYSWT was called).

TRACE A function that creates a trace of the execution of another function. Each time the traced function is called, it prints out the values of the arguments it was called with, and prints out the value it returns upon completion.

Unbound Without value; an atom is unbound if a value has never been assigned to it

window A rectangular area of the screen that acts as the main display area for some Interlisp process,

A BRIEF GLOSSARY 1.7

1

----- Next Message -----

Date: 19 Dec 91 14:42 PST
 From: sybalsky:PARC:Xerox
 To: sybalsky
 Message-ID: <<91Dec19.144256pst.43009@origami.parc.xerox.com>.?::>

<-----RFC822 headers----->

Received: from origami.parc.xerox.com ([13.1.100.224]) by alpha.xerox.com with SMTP id <11642>; Thu, 19 Dec 1991 14:43:07 PST

Received: by origami.parc.xerox.com id <43009>; Thu, 19 Dec 1991 14:42:56 -0800

From: John Sybalsky <sybalsky.PARC@xerox.com>

<-----RFC822 headers----->

6. TYPING SHORTCUTS

Once you have logged it, as per Chapters 3 or 4, you are in Interlis~D. The functions you type into the Interlisp-D executive window will now execute, that is, perform the designated task. Please note that Interlisp-D is case-sensitive; often it matters whether text is typed in capital- or lower-case letters. The shiflock key is above the left shift key; when it is pressed (on the 1186, the red LED will be on; on the 1108, the key will be depressed), everything typed is in capital letters.

You must type all Interlisp-D functions in parentheses. The Interlis~D interpreter will read from the left parenthesis to the closing right parenthesis to determine both the function you want to execute, and the arguments to that function. Executing this function is called evaluation. When the function is evaluated it returns a value, which is then printed in the Interlis~D executive window. This entire process is called the read-eval-print loop, and is how most LISP interpreters, including the one for Interlis~D, run.

The prompt in Interlis~D is a number followed by a left pointing arrow (see Figure 6.3). This number is the function's position on the History List -- a list that stores your interactions with the Interlis~D interpreter. Type the function (PLUS 3 4), and notice the number the History List assigns to the function (the number immediately to the left of the arrow). Interlis~D reads in the function and its arguments, evaluates the function, then prints the number 7.

In addition to this read-eval-print loop, there is also a programmer's assistant. It is the programmer's assistant that prints the number as part of the prompt in the Interlis~D executive window, and uses these numbers to reference the function calls typed after them.

When you issue commands to the programmer's assistant, you will not use parentheses as you do with ordinary function calls. You simply type the command, and some specification that indicates which item on the history list the command refers to. Some programmer's assistant commands are FIX, REDO, and UNDO. They are explained in detail below.

Programmer's assistant commands are useful only at the Interlis~D top level, that is, when you are typing into the

Here are a few more examples of using the programmer's assistant:

G.a TYPING SHORrCUff
1

TYPING SHORTCUTS
NIL

54k[PLUS 4 5)
9

55~REDO
9

56#??

54 +(PLUC~ 4 5)
9

56~(SETQ B '80Y)
BOY
5'~B
BOY

59" UNDO cETQ
SETQ undone.
59'.B

UN8OUND nTOM
B

SBkREDO 56
BOY
6lkB
BOY
62#

Fqøurø 6.3. Some Applications of the Programmer's Assistant

6.1 If you make a Mistake

Editing in the Interlisp-D Executive Window is explained in Section 11.2, Page 11.2. In this section, only a few of the most useful commands will be repeated.

To move the caret to a new place in the command being typed, point the mouse cursor at the appropriate position, and press the left mouse button.

To move the caret back to the end of the command being typed, press CONTROL-X. (Hold the CONTROL key down, and type ø.X.'..) The way you choose to delete an error may depend on the amount you need to remove. To delete:

The character behind the caret simply press the backspace key
The word behind the caret press CONTROL-W. (Hold the CONTROL key down, and type 'øW'ø.)

Any part of the command, first move the caret to the appropriate place in the command. Hold the right mouse button down and move the mouse cursor over the text. All of the blackened text between the caret and mouse cursor is deleted when you release the right mouse button.

TYPING SHORTCUTS 63

IF YOU MAKE A MISTAKE

The entire command press CONTROL-U. (Hold the CONTROL key down, and type in".)
 Deletions can be undone. Just press the UNDO key.
 To add more text to the line, move the caret to the appropriate position, and just type. Whatever you type will appear at the caret.

6.4 TYPING SHORTcUTS

----- Next Message -----

Date: 19 Dec 91 14:48 PST
 From: sybalsky:PARC:Xerox
 To: sybalsky
 Message-ID: <<91Dec19.144827pst.43009@origami.parc.xerox.com>.?::>

<-----RFC822 headers-----

Received: from origami.parc.xerox.com ([13.1.100.224]) by alpha.xerox.com with SMTP id <11544>; Thu, 19 Dec 1991 14:48:38 PST

Received: by origami.parc.xerox.com id <43009>; Thu, 19 Dec 1991 14:48:27 -0800

From: John Sybalsky <sybalsky.PARC@xerox.com>

-----RFC822 headers----->

7. USING MENUS

The purpose of this chapter is to show you how to use menus. Many things can be done more easily using menus, and there are many different menus provided in the Interlisp-D environment. Some are "po~up" menus, that are only available until a selection is made, then disappear until they are needed again. An example of one of these is the "background menu", that appears when the mouse is not in any window and the right mouse button is pressed. A background menu is shown in Figure 7.1. Yours may have different items in it.

```
SkGtL'h
LUop3 Icon
CHAT
F.lle0ro~er
```

```
sav"VM
5nap
```

Figure 7.1. A hackground menu.

Another common pop-up menu is the right button default window menu. This menu is explained more in Section 10.4, Page 10.3.

Other menus are more permanent, such as the menu that is always available for use with the Interlisp-D Filebrowser. This menu is shown in figure Figure 7.2, and the specifics of its use with the filebrowser is explained in Chapter 9).

```
Dnjelsta
Rcname
Hor~'UpJ
-='ffl.e
```


Compil'.
E~prnge

Recrjm Ut.fl,L'

Figure 7.2. The menu that is available when using the Filehrowser

USING MENUS 71

I

MAKING A SELEcTION FROM A MENU

7.1 Making a Selection from a Menu

To make a selection from a menu, point with the inouse to the item you would like to selert If one of the moU5e buttons is already pressed, the menu item 5hould blacken. If it is a permanent menu, you must press the leff mouse button to blacken the item. When you release the button, the item will be chosen. Figure 7.3 shows a menu with the item "Undo" chosen.

ø ø1
.lffer
Bpfor',
GeIer~,
Replace
'witch
(

'3 LIt.

Find
'=w~
pcpfInt
Edt

Edfl-Um

0~ik

Eva
E.xit

Figure 73. A menu with the item "Undo" chosen

7.2 Explanations of Men.u Items

Many menu items have explanations associated with them. If you are not sure what the consequences of choosing a particular menu item will be, blacken the menu item, and do not release the leff button. If the menu item has an explanation associated with it, the explanation will be printed in the prompt window. Figure 7.4 shows the explanation associated with the item "Snap" from the background menu.

ile0row~or

Flguvø 7.& The explanation associated with the cliosen item, Snip, is displayed in the prom pt window.

7.2 USING NENuS

I

SUBMENUS

7.3 Submenus

Some menu items have submenus associated with them. This means that, for these items, you can make even more precise choices if you would like to.

A submenu can also be found in one of two ways. One is to point to the item with the mouse cursor, and press the middle mouse button. If there is a submenu associated with that item, it will appear. (See Figure 7.5.)

```

ø
Atter
8e?are
DoloCe
Replace
Yvitch

'ut
l)nda
Find
cap
Repnnt
Edit

EditL'om
Break

Eva OK

```

TOP

Figure 7.5. The submenu associated with the menu item Exit It appeared when the mouse cursor pointed to the menu item, and the middle mouse button was pressed.

A submenu can be indicated by a gray arrow to the right of the menu item, like the one to the right of the "Hardcopy" choice in Figure 7.1. To see the submenu, blacken the menu item, and move the mouse to follow the arrow. An example of this is shown in Figure 7.6. Choosing an item from a submenu is done in the same way as choosing an item from the menu. Any submenus that might be associated with the items in the submenu are indicated in the same way as the submenus associated with the items in the menu.

```

Dnclete .~ .
Copy

Rename
Harjcopy

.=.ee ~e~;

Loa.d c'.Ee!T.:
E,puni~e

P',com Ll!eø

```

Figure 7.6. The submenu associated with the menu item Edit - It appeared when the menu item was blackened, and the mouse was moved (to follow the gray arrow).

In summary, here are a few rules of thumb to remember about the interactions of the mouse, and system menus:

- ø Press the left mouse button to select an item of a menu

Ø Press the middle mouse button to get more options - one of the ways to find a submenu

USING MENUS 73

SUBMENU5

Ø Press the right mouse button to see the default right button window menu, and the background menu

7.4 USING MENUS

----- Next Message -----

Date: 19 Dec 91 14:56 PST
 From: sybalsky:PARC:Xerox
 To: sybalsky
 Message-ID: <<91Dec19.145658pst.43009@origami.parc.xerox.com>.:>

<-----RFC822 headers-----

Received: from origami.parc.xerox.com ([13.1.100.224]) by alpha.xerox.com with SMTP id <11659>; Thu, 19 Dec 1991 14:57:09 PST

Received: by origami.parc.xerox.com id <43009>; Thu, 19 Dec 1991 14:56:58 -0800

From: John Sybalsky <sybalsky.PARC@xerox.com>

-----RFC822 headers----->

8. MOW TO USE FILES

8.1 Types of Files

A program file, or lisp file, contains a series of expressions that can be read and evaluated by the Interlis~D interpreter. These expressions can include function or macro definitions, variables and their values, properties of variables, and soon. How to save Interlis~D expressions on these files is explained in Section 11.6, Page 11.7. Loading a file is explained below, in Section 8.6, Page 8.4.

Not all files, however, have Interlis~D expressions stored on them. For example, TEdit files (see Chapter 23) store text; sketches are stored on files made with the package Sketch (see Chapter 35), or can be incorporated into TEdit files. These files are not loaded directly into the environment, but are accessed with the package used to create them, such as TEdit or Sketch. When you name a file, there are conventions that you should follow. These conventions allow you to tell the type of a file by the extension to its name. If a file contains:

Interlis~D expressions, it should not have an extension. For example, a file called "MYCODE" should contain Interlis~D expressions;

compiled code, it should have the extension ".DCOM". For example, a file called "MYCODE.DCOM" should contain compiled code;

a Sketch, then its extension should be ".SKETCH". For example, a file called "MOUNTAINS.SKETCH" should contain a Sketch;

text, it should have the extension ".TEDIT". For example, a file called "REPORT.TEDIT" should contain text that can be edited with the editor TEDIT.

8.2 Directories

This section focuses on how you can find files, and how you can easily manipulate files. To see all the files listed on a device, use the function DIR. For example, to see what files are stored on the hard disk, type
(DIR (DSK))

HOW TO USE FILES B1

DIRŽG0R1E5

To see what files are stored on the floppy disk inside of the floppy drive, type
(DIR (FLOPPY))

Partial directory listings can be gotten by specifying a file name, rather than just a device name. The wildcard "*" can be used to match any number of unknown characters. For example, the command
(DIR (DSK)T*)

will list the names of all files stored on the hard disk that begin with the letter T. An example using the wildcard is shown in Figure 8.1

```
'DIR '(P'h',(LI'.\PFILž.' 'PRIMER~T';1
'LPQh'''.LI.'1 FILE.C\,'PRIMER\
Tsi'REF.>.TEP[1]2
T6LICINT.TEDIT,1
```

FigurG 8.1. Using the function DIR with a wildcard

8.3 Directory Options

Various words can appear as extra arguments to the DIR command. These words give you extra information about the files.

(1) SIZE displays the size of each file in the directory. For example, type

(DIR (DSK) SIZE)

(2) DATE displays the creation date of each file in the directory. An example of this is shown in Figure 8.2

```
35~(DIR (DsxJ.<LI$PF1LEs>PRIMER~T* DATE)
CREATIDNOATE
```

```
(øDSK)'LI5PFILES~PRIfIER?
```

```
TA'1"REF~TEPIT;2 26-lun-R5 19:A,O:R2
TBLnmNT.TEDIT;1 26-lun'66 ja:4R~ø?
```

```
3Lq~
```

```
.....:.....
```

Figure 8.2. An eximpie using thø dirøctory option DATE
(3) DEL deletes all tho files found by the directory command

G.a HOW TO USE FILES

SUøFILE DIREO0RIES

8.4 Subfile Directories

Subfile directories are very helpful for organizing files. A set of files that have a single purpose, for example all the external documentation files for a system, can be grouped together into a subfile directory.

To associate a subfile directory with a filename, simply include the desired subfile directory as part of the name of the file. Subfile directories are specified after the device name and before the simple filename. The first subfile directory should be between less-than and greater-than signs < >, with nested subdirectory names only followed by a greater-than sign >. For example:

```
[DSK]<D1røctory>SubOlmctory>SrnSubDirøctor-y>.. >fi1on~
```

8.5 To See What Files Are Loaded

If you type FILELST<CR>, the names of all the files you loaded will display.

Type SYSFILES<CR>, to see what files are loaded to create the SYSOUT.

8.6 Simple Commands for Manipulating Files

The following commands will work with the (FLOPPY) and other devices, but have been shown with (DSK) for simplicity.

To have the contents of a file displayed in a window:

(SEE '[DSK]f11ønrn)

To copy a file: (coPYFILE '[~)o1dfi1ønrn~ '[DSF)ne,r,,ilonrn)

An example of this is shown in Figure 8.3

```
(sOPvFILE 'T~Or,RžFc.TEDIT 'PF;IMEFRøEFOø.TžDITJ  
'Dcxl,(.LIsPFILEs.PRIMžP.;ø.PRIMEP.fiEFs.TEDIT;1
```

Figure 8.3. An example of the use of the function COPYFILE

To delete a file: (DEL.FILž '[~)fi1on~)

An example of this is shown in Figure 8.4.

```
ø ,, OELFILž 'L'AMPLE.TEPITJ
```

```
ø . \. I'PFILE;.'PRIMER?>AnPLE.TžPIT;1
```

Figure 8.5. The function DELFILE

To rename a file: (RENIEFILE ø(osK)oldftlønrn '(rSF)ner,r11ønrn~)

HOW TO USE FILES 83

1

SIMPLE COMMANDS FOR MANIPULATING FILES

"LOAD" a file: Files that contain Interlisp-D expressions can be loaded into the environment. That means that the information on them is read, evaluated, and incorporated into the Interlisp-D environment.

To load a file, type:

```
(LUG '[DSff)filenm)
```

When using these functions, always be sure to specify the full filename, including subfile directories if appropriate.

8.7 Connecting to a Directory

Often, each person or project has a subdirectory where their files are stored. If this is your situation, you will want any files you

create to be put into this directory automatically. This means you should "connect" to the directory.

CONK is the Interlisp-D form that connects you to a directory. For example, COILK in the following figure:

- 1 1 11

```
29#(L'OtJN "CDv~K1,.LIv" PFILES~\PP,IMER7IM\!,I
t'OS'Y96)cLIy'PFILZCv;~PRIh1ZR.~IM>
30#
```

Figure 8.5. COILK connecting to the subdirectory "PRIMERs srnbsu~i'edory ,.IM" connects you to the subsubdirectory iM, in the subdirectory PRIMER, in the directory LISPFILES, on the device D5K. This information, the device and the directory names down to the subdirectory you want to be connected to, is called the "path" to that subdirectory. co:: expects the path to a directory as an argument.

Once you are connected to a directory, the command DIR will assume that you want to see the files in that directory, or any of its subdirectories.

Other commands that require a filename as an argument (e.g., SEE, above) will assume, if there is no path specified with the filename, that the file is in the connected directory. This will often save you typing.

8.8 File Version Numbers

When stored, each file name is followed by a semicolon and a number.

```
ffFILE.TEOIY;1
```

The number is the version number of the file. This is the system's way of protecting your files from being overwritten. Each time the file is written, a new file is created with a version number one

8.1 HOW TO use FILES

FILE VERSION NUMBERS

greater than the last. This new file will have everything from your previous file, plus all of your changes.

In most cases, you can exclude the version number when referencing the file. When the version is not specified, and there is more than one version of the file on that particular directory, the System generally uses your most recent version. An exception is the function DELFILE, which deletes the oldest version (the one with the lowest version number) if none is specified.

HOW TO USE FILES as

----- Next Message -----

Date: 19 Dec 91 15:03 PST
 From: sybalsky:PARC:Xerox
 To: sybalsky
 Message-ID: <<91Dec19.150359pst.43009@origami.parc.xerox.com>.?::>

<----RFC822 headers-----

Received: from origami.parc.xerox.com ([13.1.100.224]) by alpha.xerox.com with SMTP id <11664>; Thu, 19 Dec 1991 15:04:10 PST

Received: by origami.parc.xerox.com id <43009>; Thu, 19 Dec 1991 15:03:59 -0800

From: John Sybalsky <sybalsky.PARC@xerox.com>

-----RFC822 headers----->

10. THOSE WONDERFUL WINDOWS!

A window is a designated area on the screen. Every rectangular box on the screen is a window. While Interlisp-D supplies many of the windows (such as the Interlisp-D executive window), you may also create your own. Among other things, you will type, draw pictures, and save portions of your screen with windows.

10.1 Windows provided by Interlisp-D

Two important windows are available as soon as you enter the Interlisp-D environment. One is the Interlisp-D executive window, the main window where you will run your functions. It is the window that the caret is in when you turn on your machine, and load Interlisp-D. It is shown in Figure 10.1.

Figure 10.1. Interlisp-D Executive Window

The other window that is open when you enter Interlisp-D is the "Prompt Window". It is the long thin black window at the top of the screen. It displays system prompts, or prompts you have associated with your programs. (See Figure 10.2.)

Figure 10.2. Prompt Window

Other programs, such as the editors, also use windows. These windows appear when the program starts to run, and close (no longer appear on the screen) when the program is done running.

THOSE WONDERFUL WINDOWS' 101

CREATING A WINDOW

10.2 Creating a window

To create a new window, type: (CREATEil). The mouse cursor will change, and have a small square attached to it. (See Figure 10.3.)

Figure 10.3. The mouse cursor asking you to sweep out a window
There may be a prompt in the prompt window to create a window. Press and hold the left mouse button. Move the mouse around, and notice that it sweeps out a rectangle. When the rectangle is the size that you'd like your window to be, release the left mouse button. More specific information about the creation of windows, such as giving them titles and specifying their size and position on the screen when they are created, is given in Section 27.1.2, Page 27.2.

10.3 The Right Button Default Window Menu

Position the cursor inside the window you just created, and press and hold the right mouse button. A menu of commands should appear (do not release the right button!), like the one in figure 10.4. To execute one of the commands on this menu, choose the item. Making a choice from a menu is explained in Section 7.1, Page 7.2.

clQ1,0/

Pant

Boar
Bury

RoJisplay
Hardsopy~
Movc
Shape
shrink

Figure 10.6 The Right Button Default Window Menu

As an example, select "Move" from this menu. The mouse cursor will become a ghost window (just an outline of a window, the same size as the one you are moving), with a square attached to one corner, like the one shown in Figure 10.5.

~1

~1

Figure 10.1 The mouse cursor for moving a window

Move the mouse around. The ghost window will follow. Click the left mouse button to place the window in a new location.

10.1 THE RIGHT BUTTON DEFAULT WINDOW MENU

f

THE RIGHT BUTTON DEFAULT WINDOW MENU

Choose "Shape", and notice that you are prompted to sweep out another window. Your original window will have the shape of the window you sketch out.

10.4 An explanation of each menu item

The meaning of each right button default window menu item is explained below:

Close removes the window from the screen;
Snap copies a portion of the screen into a new window;
Paint allows drawing in a window;

Clear clears the window by erasing everything within the window boundaries;

Bury puts the window beneath all other windows that overlap it;
Redisplay redisplay the window contents;

Hardcopy sends the contents of the window to a printer or to a file;
Move allows the window to be moved to a new spot on the screen;
Shape repositions and/or reshapes the window;
Shrink reduces the window to a small black rectangle called an icon.
(See Figure 10.6.)

Figure 10.7 An example icon

Expand changes an icon back to its original window. Position the mouse cursor on the icon, depress the right button, and select Expand.

Or, just button the icon with the middle mouse button.

These right-button default window menu selections are available in most windows, including the Interlis-D Executive window. When the right button has other functions in a window (as in an editor window), the right button default window menu should be accessible by pressing the Right button

in the black border at the top of the window.

10.5 Scrollable Windows

Some windows in Interlisp-D are "scrollable". This means that you can move the contents of the window up and down, or side to side, to see anything that doesn't fit in the window.

Point the mouse cursor to the left or bottom border of a window. If the window is scrollable, a "scroll bar" will appear.

THOSE WONDERFUL WINDOWS' 103

SCROLLABLE WINDOWS

The mouse cursor will change to a double headed arrow. (See Figure 10.7.)

. 1 , 1

Figure 10.7. The scroll bar of a scrollable window. The mouse cursor changes to a double headed arrow.

The scroll bar represents the full contents of the window. The example scroll bar is completely white because the window has nothing in it. When a part of the scroll bar is shaded, the amount shaded represents the amount of the window's contents currently shown. If everything is showing, the scroll bar will be fully shaded. (See Figure 10.8.) The position of the shading is also important. It represents the relationship of the section currently displayed to the full contents of the window. For example, if the shaded section is at the bottom of the scroll bar, you are looking at the end of the file.

1 0 .

The amount of shading in the scroll bar represents the amount of the file shown in the window. Most of the file is visible. Because the shading is at the top of the scroll bar, you know you are looking at the top of the file.

Figure 10.1. The amount of shading in the scroll bar represents the amount of the file shown in the window. Most of the file is visible. Because the shading is at the top of the scroll bar, you know you are looking at the top of the file.

When the scroll bar is visible, you can control the section of the window's contents displayed:

To move the contents higher in the window (scroll the contents up in the window), press the left button of the mouse, the mouse cursor changes to look like this:

Figure 10.1. upward scrolling cursor.

The contents of the window will scroll up, making the line that the cursor is beside the topmost line in the window.

10.4 THOSE WONDERFUL WINDOWS' 103

SCROLLASLE MN00~S

ø To move the contents lower in the window (scroll the contents
øødown" in the window), press the right button of the mouse,
and the mouse cursor changes to look like this:

Flgrnro 10.10. Oownwørd scrolling curtor

The contents of the window scroll down, moving the line that is
the topmost line in the window to beside the curtor.

ø To show a specific section of the window's contents, remember
that the scroll bar represents the full contents of the window.
Move the mouse curtor to the relative position of the section you
want to see (e.g., to the top of the scroll bar if you want to see
the top of the window's contents.). Press the middle button of
the mouse. The mouse cursor will look like this:

f'9ure 10.11. Proportional scrolling crnrtror.

When you release the middle mouse button, the window's
contents at that relative position will be displayed.

10.6 Other Window Functions

10.6.1 PROMTPRINT

Prints an expression to the black prompt window.
For example, type

(P~PTPRIKT øTNIS SILL BE PRIKTED I* THE PAT UIKOoS')
The message will appear in the prompt window. (See Figure
10.12.)

1 . ø1 II

43 lpROMTPRINT 'THIS WILL BE PRINTED IN THE
PROMPT WINDOW')

Flurf 10.12. PROMTPRINTing

THOSE WONDERFUL WINDOWS' 10.5

OTHER WINDOW FUNCTIONS

10.6.2 WHICNW

Returns as a value the name of the window that the mouse
cursor IS in.

(WHICHW) can be used as an argument to any function
expecting a window, or to reclaim a window that has no name
(that is not attached to some particular part of the program.).

10.6 THOil wONOERFUL~N00vn'

----- Next Message -----

Date: 19 Dec 91 15:18 PST
From: sybalsky:PARC:Xerox
To: sybalsky
Message-ID: <<91Dec19.151815pst.43009@origami.parc.xerox.com>.?::>

<----RFC822 headers-----

Received: from origami.parc.xerox.com ([13.1.100.224]) by alpha.xerox.com with SMTP id <11655>; Thu, 19 Dec 1991 15:18:21 PST

Received: by origami.parc.xerox.com id <43009>; Thu, 19 Dec 1991 15:18:15 -0800

From: John Sybalsky <sybalsky.PARC@xerox.com>

-----RFC822 headers----->

tl. ŽDITUNG AND SAVING

This chapter explains how to define functions, how to edit them, and how to save your work.

11.1 Defining Functions

DEFINEQ can be used to define new functions. The syntax for it is:

```
(HFIffEQ (<tunctionname> (<parameterlist↓
c~y-offrnction>j>
```

New functions can be created with DEFINEQ by ryping directly into the Interlis~D executive window. Once defined, a function is a part of the Interlis~D environment. For example, the function EXANPLE-ADDER is defined in Figure 11.1.

-

HIL

```
46=(OEFINEQ (E.~AMPLE-rt"D&ER (~" B cJ
(PRINT 'THE SUM OF THE
THREE NUMBERS Is ")
(IPLUS n" B CJJJ
(EXn~MPLE~~&DERj
47-
```

Flure 11.1. Defining the function EXAMPLEøADDER

Now that the function is defined, it can be called from the Interlis~D executive window:

ø . -

NIL

```
49'. cEX~~MPLE-ffD&ER 3 4 'J;
"THE SUM OF THE THREE NIjMBERS 15
12
```

c~g

Fq'mre III. After EXAMPLžøADDER is defined, it can he executed. The function returns 12, after printing out the message. Functions can also be defined using the editor DEdit described above. To do this, simply type

```
(DF furttiornvnamej
```

EDITING AND SAVING 111

1

DEFINING FUNCTIONS

You will be asked whether you would like to edit a Dummy definition. A dummy definition is a standard template for your

function definition. Answer by typing Y for Yes, and you will be able to define the function in the editor. (See Figure 11.3. The use of the editor is explained in Section 11.3, Page 11.4.)

```

ø h.'1,fIF PJž~HOT'E'['øTi
ø Ho FH~, dean ;or oil NflT-Eøøø 1:-7, on øoU Ui 'øh ro ?dlr a 'lu
60A "Fløø;:øl

```

Figure 11.1 Using DEdit to define a function

II _ 2 _ Simple _ Editing _ in _ the _ Interlisp-D _ Executive Window

First, type in an example function to edit:

```

51~(oEFIEQ (Y~R-FIRST-fuKTIrn (A B)
(if (GREATERP A B

```

```

thøñ TNE FIR T IS GREATER
elsø THE SECO*O IS 6REATE )))

```

To run the function, type (YOUR-FIRST-FUflcTloa 3 5).

```
52~(Y~R-FIRST-Fu~TI: 3 5)
```

```
(TNE SEc~ Is GREATER)
```

Now, let's alter this. Type:

```
53~FIZ 51 cr
```

Notf that your original function is redisplayed, and ready to edit. (See Figure 11.4.)

```
llJ EO1Y1~ AHO SAVING
```

```
r,
```

```
SIMPLE EDITING IN TNE INTERLISPøD EXECUTIVE WINDOW
```

```
NIL
```

```
53~FI~ 51
+(DEFINEQ
```

```
[YOUR-FIRST-FUNCTION
(A B) (ø edited;
```

```
"~1-Dec-GB 19;"8")
(IF (GREaTERPøA B)
THEN (QUOTE (THE FiRST Is
UREATERj)
```

```
ELSE (QUOTE (THE SECOND IS
u'RE~~TER] 1A
```

f~urø11.& Using FIX to editafundion

Move the tert cursor to the appropriate place in the function by positioning the mouse cursor and pressing the Jeff mouse button.

Delete text by moving the caret to the beginning of the section to be deleted. Hold the right mouse button down and move the mouse cursor over the text. All of the blackened text between the caret and mouse cursor is deleted when you release the right mouse button.

If you make a mistake deletions can be undone. On an 1108, press the OPEN key to. UNDO the deletion. On an 1108, press the UNDO key on the

keypad to the left of the keyboard.
Now change GREATER to BIGGER:

(1) Position the mouse cursor on the G of GREATER, and click the left mouse button. The text cursor is now where the mouse cursor is.

(2) Next, press the right mouse button and hold it down. Notice that if you move the mouse cursor around, it will blacken the characters from the text cursor to the mouse cursor. Move the mouse so that the word "GREATER" is blackened.

(3) Release the right mouse button and GREATER is deleted.

(4) Without moving the cursor, type in BIGGER.

(5) There are two ways to end the editing session and run the function. One is to type CONTROL-X. (Hold the CONTROL key down, and type "X".) Another is to move the text cursor to the end of the line and press CR. In both cases, the function has been edited!

Try the new version of the function by typing:

```
58~(Y~FZIRST-F~Tzrn 8 9)
```

```
(TNz sEc~ Is BIKER)
```

and get the new result, or you can type:

```
5~RE00 52cr
```

```
(TNE SEc~ Is BIKER)
```

EDITING AND SAVING 11.3

USING THE LIST STRUCTURE EDITOR

11.3 Using The List Structure Editor

If the function you want to edit is not readily available (i.e. the function is not in the Interlisp-D Executive window, and you can't remember the history list number, or you simply have a lot of editing), use the List Structure Editor, often called DEdit. This editor is evoked with a call to OF:

```
81~(DF YWR-FIRST-f~Tla)
```

Your function will be displayed in an edit window, as in Figure 11.5.

If there is no edit window on the screen, you will be prompted to create a window. As before, hold the left mouse button down, move the mouse until it forms a rectangle of an acceptable size and shape, then release the button. Your function definition will automatically appear in this edit window.

```
!L~nb&A IA Bj (* OJtfJ' 00:010O:cw '~;'~'0 .~.tr~r
(IF 113'REATEPP A B'i EqV;r~
THEN iO0UUTE "THE ::.p0'.T f ~Ir,GER),l cl,,t'
ELSE 1~UUTE THE .:=0E.u'N& j:. eluh'ER;J)) 4ep~:c
/'~tC.h
```

```
.
Un~io
Find
```

```
Rcorint
cit.
```

```
EOIf/C T7~
Sr:ok
E0.. y
```

E..t.

Figurø II.L An Edit Window

Many changes are easily done with the structure editor. Notice that by pressing the left mouse button, different expressions are underlined. Underline BIGGER as in Figure 11.5. Release the left mouse button.

To add an expression that doesn't appear in the edit window, (i.e. it can't simply be underlined), just type it in. Doing this will create an edit buffer below the DEdit window. For example,

type LARGER and hit cr (Remember to cr! You won't be able to do anything in the editor until you cr - this can fool you at first, so beware.) A new window opens up at the bottom for the new expression. (See Figure 11.6.)

LARGER now has the bold line underneath it, while BIGGER has a dotted line.

A

11.4 EDITING ~O ~VING

USING THE LIST STRUCTURE EDITOR

```
, LAMDOA VA B\ ~ø ødltød 'ø3' øOøc 00 l F;3Q'ø) ArtOr
VV (OREATERP A B) Befom
~ VQUOTE -THE FIRST Is 816OER)) cOIOT
(15 (QUOTE VTHE SEL'ONO IS BIW~Ry,\.i Ropl&ce
witch
()
```

```
y)out
Unoo
Find
wap
```

FQum II.L Edit Window with Edit Buffer

DEdit keeps track of items you have chosen by Using a stack. The underlines tell you the order of the items on the stack. The solid underline indicates the item on the top of the stack; the dotted underline indicates the second to the top. (liIGGER was pushed on first. When LARGER was pushed on, BIGGER became the second element in the "stack", and LARGER the first.)

Many commands operate with two items on the stack. Some of them are listed below:

Atter pops the stack, and adds this top item (in this example, LARGER) to the edit window after the second item on the stack (in this example, BIGGER). The item that was at the top of the stack, LARGER, will now appear in both the original and the new position.

Before pops the stack, and adds this top item (in this example, LARGER) to the edit window before the second item on the stack. (See Figure 11.7.)

```
(LAKBDA VA 8' C' oJ'lfG '3,-Oocø~ ~F;l.Oøø ,~rtOr
(IF VGREATERp A 8J E~'inre
~ (QUOTE (THE FIRST IS ~'R ,8øIUGEP); cOIgTe
```

ELI (QUOTE (THE SECOND IS 8I,øb'E .j! ,1J F!Gplace
itch
r

tJut
Undo
Find
,=-.i,lr
P.O~rir,t
Eda

fiUre 11.7. The command Before is chosen; the word LARGER appear
before the word BIGGER

Replace pops the stack, and substitutes this top item for the second item
on the stack.

Sat tch changes the position of the first and second items on the stack in
the edit window.

Find pops the stack, and searches this top expression for an occurrence
of the second item on the stack. If the item is found, it is
underlined with a solid line, that is, pushed on the stack. To find
the next occurrence, simply choose "Find" again. If the
expression is not found, the prompt window will blink, and a

EDITING AND SAVING 115
1

USING ~E LIST STRUCTURE EDITOR

øspøc1ø11~ asøfal If yri ant to &pøcø ~r coøants)
There are other editor commands which can be very UsefUl. To
learn about them, read to the Interlis~D Refrence Manual,
Volume 2, Section 16, on DEDIT.

it .4 _ File _ Functions and _ Variables - _ How to _ See Them _ and _ Save Them
With Interlis~D, all work is done inside the "Lisp Environment".
There is no "Operating System" or "Command Level" other than
the Interlis~D Executive Window. All functions and data
strUctures are defined and edited using normal Interlisp-D
commands. This section describes tools in the Interlisp-D
environment that will keep track of any changes that you make
in the environment that you have not yet saved on files, such as
defining new functions, changing the values of variables, or
adding new variables. And it then has you save the changes in a
file you specify.

11.5 File Variables

Certain system-defined global variables are used by the file
package to keep track of the environment as it stands. You can
get system information by checking the values of these variables.
Two important variables follow.

ø FILELST evaluates to a list, all files that yoU have loaded into
the Interis~D environment.

ø filenameC0liS (Each file loaded into the Lisp environment has
associated with it a global variable, whose name is formed by
appending "COMS" to the end of the filename.) This variable
evaluates to a list of all the functions, variables, bitmaps,
windows, and soon, that are stored on that particular file.
For example, if you type:

~FILEC0*s

the system will respond with something like:
FKS YouR-FZRST-Fu*CTil)
VARS))

11.6 Saving Interlisp-D on Files

The functions (FILES?) and (NAKEFILE 'filename) are useful when it is time to save function, variables, windows, bitmaps, records and whatever else to files.

EDITING AND SAVING 117

I

USING THE LIST STRUCTURE EDITOR

message that the item was not found will appear. (See Figure 11.8 for an example of an item, the atom THIRD, not appearing in the function, YOUR-FIRST-FUKCTION.

ø1

```
L.fIFBo~P~ø~T\P _ B! (, '-J'l-.J. _ .z' _ P...,n _ 1-
THEN 'c1.lcTE _ 'THE _ FIPT _ ~I.'i'.EP']
ELSE 1øtIJITE _ HE _ '/E/I)MID
```

TrtI,,,v Sr.i
El

ET.

Figø 11.& The atom THIRD is not in the fundion being edited
Saap changes places, on the stack, of the first and second items on the stack. The edit window does not change, except that the expression that had a solid underline now has a dotted underline, and vice versa.

Delete works on only the top item of the stack. Delete removes the solid underlined expression from the edit window.
Undo undoes the last editor command.

Completing the example begun earlier, here's how to have the word LARGER that you typed into the edit buffer appear in place of the BIGGER that you selected from the DEdit window: select the SWITCH command. Notice that the two items are switched, and the stack is popped. Now select EXIT and to leave the editor, and your function will again be redefined.

11.3.1 Commenting Fundions

Tert can be marked as a comment by nesting it in a set of parentheses with a star immediately after the left parenthesis.
(ø This ii thø Von of ø c~rtt)

Inside an editor window, the comment will be printed in a smaller font and may be moved to the far right of the code. Sometimes, however, centered comments are more appropriate. To center a comment, type ,, after the left parenthesis.
ø This co.oortt ø111 rtot bø rnd to thø ?ør ri9ht of thø
co5oø but ø111 bø cørttørd)

It is also possible to insert linebreaks within a comment. A dash should be placed in the comment whcrevør A carriagø return is needed. Thii featurø allows several commønt1 to bø placed

inside one set of parentheses.

(This is the first time at the top of the page.)

11.6 FINDING AND LAYING

SAVING INTERLISP-D ON FILES

(FILES?) displays a list of variables that have values and are not already a part of any file, and then the functions that are not already a part of any file.

Type:

(FILES?)

the system will respond with something like:

the variables: ~.VARIABLE CURRENT.turtle.. to be dumped.

the functions: RIGHT LEFT FOIAff lick*Aa CLEAR-uREEl.. to be dumped.

start to say where the above should go?

If you type Y, the system will prompt with each item. There are three options:

(1) To save the item, type the filename (unquoted) of the file where the item should be placed. (This can be a brand new file or an existing file.)

(2) To skip the item, without removing it from consideration the next time (FILES?) is called, type cr This will allow you to postpone the decision about where to save the item.

(3) If the item should not be saved at all, type J. NoilhQ re will appear after the item.

Part of an example interaction is shown in the following figure:

HIL

u31~(FILES,)

The variables: MY-'y'AR. To be dumped.

the functions: MY-SECUINO-FUNCTION,

YJUP-FIPOT'FUNI)TION

to be dumped.

want to say where the above 30 of 'ye'

(variables)

NY-VAR Nowhere

(functions)

NY-SELrNO-FUN&'TION File name: E;~AMPL~

F~11.9. Part of an interaction using the function FILES?

(FILES?) assembles the items by adding them to the appropriate file's COMS variable. (See Section 11.5, Page 11.7.)

(FILES?) does NOT write the file to secondary storage (disks or floppies). It only updates the global variables discussed in Section 11.5.

(NAKEFILE 'TI lena) actually writes the file to secondary storage. Files should only be written when the time is set. If the time is not set, you will run into problems, such as not being able to copy your file. To check

the time, type
(rite)

If the date is correct, you can safely use IRE FILE. If it is not correct, set the time with the function SETTIME. To use it, type (SETTIME date), where date is a string such as the one shown in Figure 11.10.

it a Eomlfill ANC SAVING
I

SAVING INTERUSP~ ON FILES

NIL

```
97;k(SETTIME "10-Jul-86 15:08 2<8)
"16-Jul-86 15:08:22 EDT"
98+
```

Figure 11.10. Using the SETTIME function to set the date and time. Once the time is set correctly, use the function MAKEFILE. Type: (MAKEFILE 'P.FILE.~)

and the system will create the file. The function returns the full name of the file created. (i.e. (DSK)MY.FILE.NAME.; 1). Note: Files written to (DSK) are permanent files. They can be removed only by the user deleting them or by reformatting the disk.

Other file manipulation functions can be found in Section 8.6, Page 8.3.

EDITING AND SAVING 119
I

----- Next Message -----

Date: 19 Dec 91 15:20 PST
From: sybalsky:PARC:Xerox
To: sybalsky
Message-ID: <<91Dec19.152031pst.43009@origami.parc.xerox.com>.:>

<-----RFC822 headers-----

Received: from origami.parc.xerox.com ([13.1.100.224]) by alpha.xerox.com with SMTP id <11670>; Thu, 19 Dec 1991 15:20:42 PST

Received: by origami.parc.xerox.com id <43009>; Thu, 19 Dec 1991 15:20:31 -0800

From: John Sybalsky <sybalsky.PARC@xerox.com>

<-----RFC822 headers----->

13. FLEXIBILITY AND FORGIVENESS: CLISP AND DWIM

CLISP, (Conversational Lisp), and DWIM, (Do What I Mean), are two Interlisp utilities that make life easier.

13.1 CLISP

CLISP allows the machine to understand and execute commands given in a non-standard way. For example, Figure 13.1 contains an example expression (4 + 5).

NIL

```
b'4-iJ + 5;
9
```

```
85'
```

Figure 13.1. cLisp allows the use of infix notation

Without CLISP, you would need to type this using the notation (PLUS 4 5). CLISP allows you to use expressions such as (4 + 5) for all arithmetic expressions.

CLISP also allows you to use more readable forms instead of standard Lisp control structures. Expressions like IF-THEN-ELSE statements can replace COND statements. For example, instead of:

```
(CIO 1J6RE(APLTUESRPBA B (PLUS A 10))
10
```

the following can be used:

```
(if (A ~ B) then (A + 10) else (B + 10))
```

The system translates this CLISP code into Interlisp-D code. Setting flags will allow you to either save the CLISP code, or save the translation. One such flag is CLISPIFTRANFLG; if it is set to t, all the IF statements will be replaced with the equivalent CORD statements. This means that when you edit the function, the IF will be removed and replaced with the CORD. Typically, flags such as this one are set in your INIT file. These flags are discussed in the Interlisp-D Reference Manual in Volume 2, Section 21.

FLEXIBILITY AND FORGIVENESS. cLisp AND DWIM 13 I

OWIM

13.2 DWIM

DWIM tries to match unrecognized variable and function names to known ones. This allows Lisp to interpret minor typing errors or misspellings in a function, without causing a break. Line 87 of Figure 13.2 illustrates how the misspelled 0ANNANNA was replaced by 8ANANA before the expression was evaluated.

```
NIL
```

```
a7(8ETQø 8~øN.HA 'FRUITj
FRUIT
```

```
38'8nNN,,~NNA
=8,, 'H,,NA
FRUIT
39'
```

Figure 13.2. Examples of CLISP and DWIM features

Sometimes DWIM may alter an expression you didn't want it to. This may occur if, for example, a hyphenated function name (eg. (NY-FUNCTION)) is misused. If the system doesn't recognize it, it may think you are trying to subtract "FUN~ION" from "MY". DWIM also takes the liberty of updating the function, so it will

have to be fixed. However, this is as much a blessing as a curse, since it points out the misused expression!

```
13.2 F~11UM AND ~ROVENES$: cub AND OWN
I
```

----- End Forwarded Messages -----

Figure 13.2. Examples of CLISP and DWIM features

Sometimes DWIM may alter an expression you didn't want it to. This may occur if, for example, a hyphenated function name (eg. (NY-FUNCTION)) is misused. If the system doesn't recognize it, it may think you are trying to subtract "FUN~ION" from "MY". DWIM also takes the liberty of updating the function, so it will have to be fixed. However, this is as much a blessing as a curse, since it points out the misused expression!

```
13.2 F~11UM AND ~ROVENES$: cub AND OWN
I
```

----- End Forwarded Messages -----