TABLEOFCONTENTS

~)19 ~'~

TA8~ OF CONTENtt To'.,
1

- - - -

. - - - - . - - , -

TABLE OF CONTENtr

YA8~ OF CONTENTS TOC.3

TABLE OF CONTEND yl

----- 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 CONTENff

TA.G TAaU0FC0NTENff

PREFACE

it wos dawn and the locd told him it was down the road a p;ece, lefl &t the hst fishing bridge in the counvy right at the apple tree stump, and onto the d;rt roadjust before the hill. At m;dnight he knew he was lo¤t.
-Anonymous

Welcome to the Interlisp-D programming environment! The Interlis~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 l thought I'd need to write." Unfortunately, along with the power comes mind-numbing complexity. The Intedisp Refertnce 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 rtart!

We developed this primer to provide a starting point for new lnterlis~D usert, to enhanceø 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 courte in Artificial Intelligence (Al), and have seen a demonrtration of how lnterlis~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 ste~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 lnterlis~D language and Programming Environment", includes Chapters 6 through IS. These chapters discuss primary elements in lnterIis~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 lnterlis~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 flnd 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 Al 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 AlS staff who provided both editorial and systems support.

PREFA~ 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. ABRlEFGLOSSARY

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 lnterlis~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 lnterlis~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 lirt 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 I.I.

Loops Icon
FileB'owser

Figuro 1.1. The Menu that appeort 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 'ø pixels, 'ø 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 strvcture by use of a display that allows the user to "move" around within the data rtructure.
button

A BRIEF GL0SSARY 1.1
1

A BRIEF GLOssAny

(1) (n.) Akeyonamouse.

(2) (v.t.) To depress one of the mouse keys when making a selection.
EAR A function that returns the head or firrt element of a list. See CDR.

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

NIL

B6+(PLus 3A

Figurø lJ. Me caret is to the right of the numher 3. When a characters frped atthe keylsoard. it will ap~ar at the caret

CDR A function that returns the tail (that is, everything but the first element) ofa list SeeEAR.

CLlSP A mechanism for augmenting the standard Lisp syntax. One such augmentation included in Interlisp is the iterative rtatement. SeeSection 13.1.

cr Please press your carriage return key.
datatype

(1) The kind of a datum. In Interlisp, there are many SystemAefine~ datatypes e.g. Floating Point, Integer, Atom, etc.
(2) A datatype can also be user~efined. In this case it is like a record made up from system types and other user-defined datatypes.
DWlM D~whatøl-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 errort). The DWlM facility is called automatically whenever an error occurt in the evaluation of an Interlisp expression. If DWlM 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 ertensive facilities for detecting and handling error conditions, to enable the testing, debugging, and revising of impertect programs.
evaluate or EVAL Means to flnd the value of a form. For example, if the variable X is bound to 5, we get 5 by evaluating X. Evaluation of a Interlisp function involves evaluating thø arguments and then applying thøfunction.

file packagø A set of functions and convøntions that facilitato thø bookkeøping involved with working in i largø systøm consisting of many sourcø codø files and thøir compiled countørparts.

Essentially, thø fllø packagø k:ps track of whørø things arø and

1,

AøR1EFGLos~y
l

A øRl—F GLoSSARY

whet things hevo chonged. N 4150 kaps trøck of which files hove
been modifiod end n#d to be updetod end recompiled.
form Another wey of seying ~xpre5sion. An Jntorlisp-D ex~on
tbetcen be evaluated.

function A Lisp function is e piece of l;sp code thet executes end returns e
veiue.

history The progremmerøs essistent is l,uilt eround e memory structure
celled the hirtory Iirt. The hirtory functions (e.g. FIX, UNDO,
REDO) ere part of this essirtant. These operations allow you to
conveniently r~work previously specifiecl operations.

History List As you type on the xreen, you will notice a number followecl by
a prompt attow. Each number, and the information on that line,
is seeluentially rtored as the History List Using the History List,
you can easily reexecute lines typed earlier in a worksession. See
Chapter6.

icon A pictorial representation, usually of shrunken window.

lnterlis~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
lnterlis~D.

NIL

8~#iPRO*PTPRIHT "HELLO" A

Fqurø tJ. m window

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

iterative statement (also called i.s.) A statement in Interlisp that repetitively executes
abody of code. (E.g.(forxfromltosdo(PRlNTx))isani.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 leecause its value is being changed by
each cycle through the loop. All iterative variables are local to
the iterative rtatement 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 rtructures. lnterlis~D is an
implementation of the LISP language together with an
environment (set of tools) for programming, an a set of packages
that ertend 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 lJ
1

A BRIEF GLOSSARY

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

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

Marterscope A program analysis tool. When told to analyze a program, Masterscope creates a data base of information about the program. In particular, Marterscope 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: p0~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.).

Fw- 1.& Mousø

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

F~m I.L Mousø c~~~
Mousø Curtor Icons

I.A A llEF GLos~Y
l

A IRlEF GLOSSARY

I Wait Tho processor is busy.

The processor is saving a anpashot or your cureent system session. This is usually donø when tbc procffssor hos ~n idle for a while.

The "Mouse Confirm Cursor". It appeatt 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 tight, 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 leff 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 lnterlis~D symbol for the empty list h can also be represented by a lefl paren followed by a right paren: (). It is the only expression in lnterIis~D that is both an atom and a list pixel Pixel rtands for PIcture Element. The xreen 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 lnterIis~D functions are printed with special indentation, to make them easier to read. Functions are pretty printed in the structure eclitor, 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 tunction-name). For an example of this, see Figure 1.6.

96.(PP HEAD)

[LANBDA (LST) <łø6rtøG; 'ø~JlinH13;3&ø)
(CAR LsTJ)
(HEAD)
97.'

Fbm 1.6. An oxam~ u~oftho pro 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, UlOO, and'or REDO your previous expressions typed to the lnterlis~D executive window. (See Chapter 6.)

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

Fqurø 1.7. Prompt window

property list A list of the form (<property-namel> <property-valuel> <property-name2> <property-value2> ....) associated with an atom. It accessed by the functions GETPROP and PUTPR0P.

record A record is a data-structure that consists of named "flelds". 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 appeaff 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
8ury

Redi~play
Hardcopy~
Move
Shapo
Shrink

f~a 1.1. ttø Right Sutton DqfaultWindow Menu

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

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

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

sysout A flløcontaining G wl'0lø Lisp environmønt: namely, lnterlis~O, evørything thø user' døfinecl or loaded into the environment, thø

1.6 A ~N:EF GL0SsARY
I

A BRIEF GLOSSARY

windows that øppeored on tho Kreen, tbø ømount of memory used, and 50 on. Evorything ;s rtorød in thø sysout filø exactly .5 it was whon tho function SYSWT was called).

TFACE A function that crøates a trace of the execution of another function. Eich 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 lnterlis~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; offen it matters whether text is typed in capital- or lower-case letters. The shifflock 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 lnterlis~D interpreter wtll read from the leff 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 lnterlis~D executive window. This entire process is called the read-eval-print loop, and is how most Ll5P interpreters, including the one for lnterlis~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 lnterlis~D interpreter. Type the function (PLUS 3 4), and notice the number the History List assigns to the function (the number immediately to the leff of the arrow). lnterlis~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 assistantøø. It is the programmer's assistant that prints the number as part of the prompt in the lnterlis~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 lnterlis~D top level, that is, when you are typing into the

lnterlis~D executive window. They will not work in user-defined functions.

As an example use of the programmer's assistant, use REDO to redo your function call (PLUS 3 4). Type REDO (Note: programmer's assistant commands can be typed in either upper

TYPING 5H0RTCUTS 6 1

TYPING SHORTCUTS

or lower case) at the prompt, then specify the previous expression in one of the following ways:

(1) When you originally typed in tne function you now want to refer to, there was a History List number to the left of the arrow in the prompt. Type this number affer the programmer's assistant command. This is the method illustrated in the following figure:

iPLUe~ 3 4)
Cø'5~REOO '24
7

26-','

, , , . . , . , ... , ... , , ,
, .
Figure 6.1. Using the programrner's assistant to REDO a function, when you know the its number on the history list

(2) A negative number will specily the function call typed in that number of prompts ago. in this example, you would type in-I, the position immediately before the current position. This is shown in the following figure:

'2ł~.(PLUS :3 j)
;;;;

'-,7'

ii, ~' F.EDS -i
7

:ø'0~.

,
, , , , ,: , , :, , ,:, , , , , , ;, : , , , ,; , ,

Figurø 6.2. Typing a negative number affer the programmer's assistant command will cause it use the function found on the History List that many positions before the current one.

(3) You can also specify the function for the programmer's assistant with one of the items that was in that function call. The programmer's assistant will se'rch backwards in the History List, and use the first function it finds that includes that item. For example, type REDO PLUS to have the function (PLUS 3 4) reevaluated.

(4) If you type a Programmer's Assistant command without specifying a function (i.e., simply typing the command, then a cr) the Programmer's Assistant executes the command using the function entered at the previous prompt.

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

G.a TYPING SHORrCUff
1

TYPING SH0RTCUTS
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
6IkB
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 leff 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 rype '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 ten. All of the blackened tert 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~prnnge

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~
pcpflnt
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 menus items have submenus associated witl, them. This means that, for these items, you can make even more precise choices if you would like to.

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

øl
Atter
8e?are
DoloCe
Replace
Yvitch

'ut
l)nda
Find
cap
Repnnt
Edit

EditL'om
Break

Eva OK

TOP

FigurQ 7.5. The submenu associated with the menu item Exit It appeared when the mouse curtor 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'ø 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.

Dnclelete .~ .
Copy

Rename
Harjcopv

.=.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 mou>e was moved (0 follow the gray arrow.

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

ø Press the leff 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 lnterlis~D interpreter. These expressions can include function or macro definitions, variables and their values, properties of variables, and soon. How to save Interlisp-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 lnterlis~D expressions stored on them. For example, TEdit files (see Chapter 23) store tert; 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:

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

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

a Sketch, then its extension should be ".SKETCHøø. For example, a file called ø'MOUNTAlNS.SKETCH" should contain a Sketch;

text, it should have the extension ".TEDlTø'. For example, a file called 'øREPORT.TEDlT'ø should contain text that can be edited with the editor TEDlT.

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 Y;ard disk, type
(DIR (DSK))

HOW TO USE FILES B1

DlR—G0R1E5

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 "ø'ø can be used to match any number of unknown characters. For example, the command
(DIR (DSK)Tø)

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'.\PFIL—.'.'PRIMER~T';l
'LPQh/".LI."'l FILE.C\,'PRIMER\
Tsi'REF.>.TEP[1)2
T6LlClNT.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~PRIflER?

TA'1"REF~TEPlT;2 26-lun-R5 19:A,O:R2
TBLnrnNT.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 H0W TO USE FILES

SUøFlLE DlREO0RlES

8.4 Subfile Directories

Subfile directories are very helpful for orgonizing 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 subfiledirectory.

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

[DSK)<D1røctory>SubOlmctory>SrnbSubDirøctor-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 simplicityø
To have the contents of a file displayed in a window:
(SEE '[DSK)f11ønrn)

To copy a file: (coPYFILE '[~)o1dfi1øn~ '[DSF)ne,r,,ilonrn)
An example of this is shown in Figure 8.3
(sOPvFILE 'T~0r,R—Fc.TEDIT 'PF;IMEFRøEFOø.T—DITJ
t'Dcxl,(.LIsPFILEs.PRIM—P.;ø.PRIMEP.fiEFs.TEDIT;1

Figure 8.3. An example of the use of the function COPYFILE
To delete a file: (DEl.FIL— '(~)fi1on~)
An example of this is shown in Figure 8.4.
ø ,, OELFIL— 'L'AMPLE.TEPITJ
ø . \l.. I'PfILE;'."PRIMER?>AnPLE.T—PIT;1

FigureS.O. The function DELFlLE

To rename a file: (RENlEFILE ø(osK)oldftlønrn '(rSF)ner,r11øn~)

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
Offen, 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, COilKin the following figure:
- 1 l 11

29#(L'OtJN ''CDv~K1,.LIv"PFILES~\PP,IMER7IM\,!,I
t'OS'Y96)cLIy'PFIL—Cv;~PRIh1—R.~lM>
30#

Fqrnre 8.5. COflffeaing to the subdiredory "PRIMERs srnbsu~i'edory ,.lM" connects you to the subsubdirectory iM, in the subdirectory PRIMER, in the directory LlSPFlLES, 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

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

fffILE.TEOIY;1

The number ii the vertion 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 vettion number one

8.1 HOW T0 us: FILES

FILE VERSION NUMBERS

greater than the lost. This now fle will hove 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 vertion is not specified, and there is more that one vertion 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 vertion 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---->

tO.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 lnterlis~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 lnterlis~D environment. One is the lnterlis~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 lnterlis~D. It is shown in Figure 10.1.

Figure 10.1. Interlsp-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.

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 leff 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 leff mouse button. More specific information about the creation of windows, such as giving them titles and specifying their size and position on the xreen 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,ø/
Pant

'[oar
Bury

RoJisplay
Hardsopy~
Movc
'5hape
shrink

Figurø 1O.& The Right Button Default Window Menu

As an example, select "Move" from this menu. The mouse cursor will become a ghost window Oust 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.
~l

~l

F~ure 10.1 Thø mousø cunor !or moving & window

Move the mousø around. The ghost window will follow. Click the left mousø button to placø tho window in a new location.

10.1 TH0Sff w0NKQFUL WlH00~l

f

THE RIGHT 8Uff0N DEFAULT MN00~ MFNU

Choose "Shape", afid 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 cleart the window by erasing everything within the window boundaries;

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

Hardcopy sends the contents of the window to a printer or to a flle;
Move allows the wi ndow to be moved toanew spot on the screen;
Shape repositions and/or reshapes the window;
Shrink reduces the window to a small black rectangle callecl an icon. (See Figure 1O6.)

Figum lO.L An example icon

Expand changes an icon back to iB 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 lnterlis~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 curtor to the leff or bottom border of a window. If the window is scrollable, a "scroll bar" will appear.

THOSE WONDERFlJL WINDOWS' 103

SCROLLABLE WINDOWS

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

. 1 , 1

Figuro 10.7. The scrolJ bar of a scrollable window. The mouse cursor changes o a double headed arrow.

The xroll 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 relationshi'p of the section currently diplayed to the 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 ø .

The amount of :>hadin~ in
A:::;:. the scroll bar represents
the amount of the rile

;>hown in the window. Most
of the file is visible.
Because the shading is at
the tap of the scroll bar,
you know you are looking
at the top of the file.

Figurø 10.1 Tho 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 sat 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 displayecl:

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

Fun 10.1. upward icrollinq cuttor.

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

10.4 THo¤a ~HOERFUL~~w51

SCROLLASLE MN00~S

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

Flgrnro 10.10. Oownwørd scrollinq 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 crnrtor.

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 PROMPTPRlNT

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 Il

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

Flurf 10.12. PROMPTPRlNTing

THOSE WONDERFUL WINDOWS' 10.5

OTHER WINDOW FUNCTIONS

10.6.2 WHlCNW

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

(VHICHW) 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. —DlTUNG 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-offrnnction>j>

New functions can be created with DEFINEQ by ryping directly into the lnterlis~D executive window. Once defined, a function is a part of the lnterlis~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-

F1ure 11.1. Defining the function EXAMPLEøADDER

Now that the function is defined, it can be called from the lnterlis~D executive window:
ø . -

NIL

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

c~g

Fq'rnre IIJ. After EXAMPL—øADDER 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 furttiorvnamej

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,flF PJ—~-HOT'E'['øTi

ø Ho FH~., dean ;or oil NflT-Eøøø 1:-7, on øoU Ui ';øh ro ?dlr a 'lu

60A "Fløø;:øl

Figurn 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~(oEFIxEQ (Y~R-FIRST-fuKTIrn (A B)
(if (GREATERP A B

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

To run the function, type (YOUR-FIR¤T-FUflcTIoa 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. (SeeFigure 11.4.)

llJ EO1Y1~ AHO SAVING

r,

SIMPLE EDITING IN TNE INTERLISPøD EXECUTIVE WIND0W

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 Jeff of the keyboard.
Now changeGREATERtoBIGGER:

(1) Position the mouse cursor on the G of GREATER, and click the leff mouse button. The text curtor is now where the mouse cursor 15.

(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 crø In both cases, the function has been edited!

Trythe new version of the function bytyping:
58~(Y~-FZRST-F~Tzrn 8 9)
(TN— 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 STRUaURE 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, offen called DEdit. This editor is evoked with a call to OF:

81~(DF YWR-FIRST-f~TIa)

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 leff 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' øø:ø1øO:cw '~;'~''ø .~.tr~r
(IF 113'REATEPP A B'i EqV;r~
THEN iOøUUTE "THE ::.pø'.T f ~Ir,GER),l cl,,t'
ELSE 1~UUTE THE .=øE.u'N& j:. eluh'ER;J)) 4ep~:c
/''tC.h
.
Un~io
Find

Rcorint
cit.

EOlfl/C T7~
Sr:ok
Eø.. y

E..t.

Figurø Il.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 crø (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)) cOIOtO
(15 (QUOTE VTHE SEL'ONO IS BIW~Ry,\.i Ropl&ce
witch
( )

y)out
Unoo
Find
wap

FQum Il.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 affer 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); cOlGte

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

tJut
Undo
Find
,=-.i,1r
P.O~rir,t
Eda

fiUre 11.7. The command Before is chosen; the word LARGER appean
Iefore 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 occurance 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 occurance, 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~ asøfa1 If yri ant to &pøcø ~r coøants)
There are other editor commands which can be very UsefUl. To learn about them, read to the lntertis~D Reffrence Manual, Volume 2, Section 16, on DEDIT.

it .4 _ File _ Functions and _ Variables - _ How to _ See Them _ and _ Save Them
With lnterlis~D, all work is done inside the "Lisp Environment". There is no "Operating System" or "Command Level" other than the lnterlis~D Executive Window. All functions and data strUctures are defined and edited using normal Interlisp-D commands. This sertion 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 lnteris~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*CTIil)
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.flFBo~P~ø~T\P _ B! (,'-.J'l-.J. _ .z' _ P..,n _ 1-
THEN 'c1.lcTE _ 'THE _ FIPT _ ~I.'i'.EP']
ELSE 1øtlJlJTE _ HE _ '/E/l)MlD

TrtI,,,v Sr.i
El

ET.

Figwø 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 Iinebreaks within a comment. A dash should be placed in the comment whcrevør A carriagø return is needed. Thii feoturø allows several commønt1 to bø placed

insidø one S.t of pirøntheses.

(ø This cołoo.t ø111 h t~~ at. tøo 1iøos. -

11.6 FIrING AND LAYING

SAVING INTERLlSP-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 part of any file.

Type:

(FILES?)

the system will respond with something like:
tbø variables: ~.VARIlLE cURREKT.tuRTLE.. to be du;ed.
thø functions: RI6HT LEFT FOIAff liCK*Aa cLEAR-uREEil.. to be d~~.

srit to s&y øbere thø abovø 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 crø 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 afler the item.

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

HIL

u31~(FILES,)

Che variables: MY-'y'AR. To be di.imped.
the functions: MY-SEcuNO-FUtllJTIJN,
YJUP-FIPøoT'FUNi)TIJN
to be dumped.

want to say where the .ibove 30 ø? 'ye'
(variables)
NY-VAR Nowhere
(runctions)

NY-SELrnNO-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 upclates the global variables discussed in SectionIt.S.

(NAKEFILE 'Tl lenaøe) actually writes the file to secondary storage. Files should only be writen 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, typø
(riTE)

If the date is correct, yoU can safely use IRE FILE. If it is riot correct, set the time with the function SETTIKE. To use it, type (SETTIKE date), where datø isa string such as the one shown inFiguretl.10.

it.a Eomlflll ANC SAVING
I

SAVING INTERUSP~ ON FILES

NIL

97;k(SETTIME "10-Jul-86 15:08 2<8)
"i6-Jul-86 15;08:22 EDT"
98+

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

and the system will create the file. The function returns the full name of the file created. (i.e. (DSK)MY.FlLE.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---->

t3. FLEXIBILITY AND FORGIVENESS:
CLISP AND DWIM

CLlSP, (Conversational Lisp), and DWlM, (Do What Mean), are two Interlisp utilities that make life easier.

13.1 CLlSP

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

NIL

b'4-iJ + 5;
9

85'

F9ure 13.1. cLlsP allows the use of infix notation

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

CLlSP also allows you to use more readable forms inrtead 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 CLlSP code into Interlisp-D code. Setting flags will allow you to either save the CLlSP code, or save the translation. One such flag is CLISPIFTRANFLG; if it is set to ffIL, all the IF statements will be replaced with the equivilent CORD statements. This means that when you DEdit the function, the IF will be removed and replaced with the CORD. Typically, flags such as this one are set in your INlT file. These flags are dixussed in the Intertlsp-D Reference Manual in Volume 2, Section 21.

FLEXIBILITY AND FORGIVENESS. cLIsP AND DWIM 13 I
I

OWlM

13.2 DWlM

DWlM 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 CLlSP and DWlM features

Sometimes DWlM 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~lON" from "MY". DWlM 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~1lUM AND ~ROVENES¤: cub AND OWN
I

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

Figure 13.2. Examples of CLlSP and DWlM features

Sometimes DWlM 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~lON" from "MY". DWlM 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~1lUM AND ~ROVENES¤: cub AND OWN
I

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