NSMaintain allows you to view and modify objects in the Clearinghouse data base from inside Lisp. Similar operations are available when chatting to a Clearinghouse service.

Requirements

Xerox NS network environment with Clearinghouse server(s).

DES.LCOM.

Installation

Load NSMAINTAIN.LCOM from the library. This file automatically loads DES.LCOM. DES is currently only used by the Change Password command, so its loading can be omitted if you do not need that command.

Clearinghouse Concepts

The Clearinghouse maintains a distributed data base of *objects*, each of which has a set of *properties*. The objects are such things as users, groups, and network servers; the properties are such attributes as a server address or a user's mailbox location.

Clearinghouse objects are partitioned into a three-level hierarchy: each object is contained in a *domain*, which in turn is part of an *organization*. A fully qualified object name is a three-part name in the form *object:domain:organization*. Similarly, a domain name is a two-part name of the form *domain:organization*. Lisp maintains a notion of the *default domain*, which is typically the domain in which you and the servers in your immediate area are registered. When typing object names, you may omit the *organization* field or both *domain* and *organization* fields if they are the same as your default domain. Similarly, when typing a domain name, you may omit the *organization* field if it is the same as the default.

When printing the names of objects, the system usually elides the domain and/or organization, following the same rules. For example, for the object named "John Jones:Sales:ACME", the system would print "John Jones:" if the default domain were "Sales:ACME", or "John Jones:Sales" if the default domain were "Admin:ACME". NSMaintain, however, prints fully qualified names in certain places that do not need the compactness of the elided names, so as to reduce potential confusion. For the same reason, whenever NSMaintain prompts for an object name and you omit one or two of the fields, NSMaintain automatically

echoes the defaults for you. You can change the defaults with the "Change Default Domain" command.

Any object in the Clearinghouse can have one or more *aliases*, which are Clearinghouse names that point directly to the object. An alias can be thought of as a "nickname", and can be used interchangeably with the "real name" for virtually all operations. For example, it is common practice to register users with their full names and provide at least one alias consisting of their last names.

Some objects in the Clearinghouse are groups, rather than individuals. Groups are described further in the section on group commands.

For more information on the Clearinghouse service, consult the *Interlisp-D Reference Manual* or the Clearinghouse documentation, which is part of the Network Systems documentation kit.

User Interface

NSMaintain runs in an Exec window. To start it up, evaluate:

(NSMAINTAIN) [Function]

Starts an NSMaintain session. It prompts with "CH:" in the current window and awaits commands from you. Command names complete automatically following one- or two-letter inputs. Type Q, for the Quit command, when you wish to finish.

Most of the commands take as input from you one or more Clearinghouse object names. For many commands, NSMaintain offers you the same name as you last used in a similar context. For example, if you use the Describe command to learn about an object that is a group and then use the List Members command, NSMaintain offers you the group name just described. To accept the name, just press the carriage return; otherwise, start typing the desired name; your type-in replaces the offered name. The alphabetic case of names is not significant; you may type in either upper or lowercase. However, the commands that create objects will preserve the exact case of the name as you first type it.

Typing a null name to most commands aborts the command. If a sample name is offered, you have to backspace over it, or use Control-Q to erase the whole name. You can also usually use Control-E to abort a command.

The description of the commands below is partitioned into two parts: general *user* commands and *administrator* commands. The user commands can be used by anyone, and mostly are concerned with viewing the data base. The administrator commands allow system administrators to modify the data base; these commands cannot be used by ordinary users. However, there are two administrator commands, Add Self and Remove

Self, that can be used by anybody to join or leave groups with open access.

User Commands

Anyone can use these commands to obtain information, change passwords, and change NSMaintain's defaults.

Obtaining Information

These commands let you examine the data base.

Most of the List commands enumerate items in the data base matching a particular *pattern*. A pattern is a Clearinghouse name optionally containing asterisks as wild cards, which match zero or more characters. Wild cards are permitted only in the first component of the name; the remaining parts must be a valid domain and organization. In a two-part name, wild cards are permitted in the domain name but not the organization. Thus, for example, "*John*:Sales:ACME" is a valid pattern matching objects whose name component contains the substring "John"; "Joe:*:ACME" is not a valid pattern.

Following a List command (except List Domains), you can use the Show Details command to get more information about any of the names listed.

Describe Gives a description of any object registered in the Clearinghouse, what its registered name is (in case you typed an alias), and all interesting properties of the object. If the object is a group, its Owner and Friends are also listed; to see its members, use the List Members command.

List Domains Lists all domains matching a specified domain pattern; for example, "*:Xerox" to list all domains in the Xerox organization.

List Clearinghouses Lists all Clearinghouse servers that serve a specified domain.

List Administrators Lists the administrators for a domain.

List Aliases Lists all aliases matching a given pattern. Note that none of the other List commands (except for List Objects with property any) will match your pattern against an alias, so you may want to use the List Aliases command if you don't find the object you were looking for otherwise.

List Groups Lists all groups matching a given pattern.

List True Groups Same as List Groups, but filters out all names that also have a "user" property. These "groups" are typically used for mail forwarding. This command requires considerably more computation than List Groups.

List Servers Lists objects matching a given pattern and registered as a server. You are prompted for the type of server; for example, Mail, File, or Print. Type ? to see the choices.

List Users Lists the names of all users matching a given pattern.

List Objects Lists all registered objects of of an arbitrary Clearinghouse type that match a given pattern. You are prompted for the type(a Clearinghouse property name) and pattern. To list all objects,

that is, those with *any* property, press the carriage return to the property prompt, or supply the property "*".

List Members — Lists the members of a specified group.

Show Details — Prompts you for a name, performing automatic spelling completion from the names printed in the most recent List command such as List Users, and then performs the Describe command on the name. Press the carriage return in response to the name prompt to return to the main CH: prompt.

If there was only one name in the list, this command does a Describe on it without further prompting.

Type Entry — Synonym for Describe.

Type Members — Synonym for List Members.

Miscellaneous

Change Default Domain — Changes the defaults used on type-in inside NSMaintain for domain and organization. NSMaintain asks if you also want to change the defaults globally; if you say yes, the variables CH.DEFAULT.DOMAIN and CH.DEFAULT.ORGANIZATION are changed, so that the new defaults have effect outside of NSMaintain as well.

The defaults are never used when typing aliases in the Add Alias and Add User commands; these commands default the domain to be the same as the domain of the main object.

Note: Unless you change the defaults globally, this command does not affect type-out and the Change Login command, which are still performed with respect to the global defaults. This may change in a future implementation.

Change Login — Prompts you for a new name and password, which becomes the default NS login on your machine and for NSMaintain. You can also use this to fix your password if you were incorrectly logged in before you started NSMaintain.

Change Password — Allows you to change your password. Prompts for a user name, offering your logged-in name as default. A domain administrator can also change other users' passwords. After you type the new password, you are asked to retype the password, to ensure that you typed what you thought you had. Neither password is echoed.

Quit — Exit NSMaintain.

Administrator Commands

These commands modify the Clearinghouse data base. In general, they require that you have the appropriate administrator access.

Creating and Deleting Objects

To create or delete objects in a domain, you must be an administrator for the domain.

Add Alias	Assigns an alias for a specific object registered in the Clearinghouse data base.
Add User	Creates a new user. You are prompted for the user's name (preferably a full name), a brief description (for example, the user's affiliation, office number, etc), an initial password, and one or more aliases.
Change Remark	Allows you to change the remark; that is, text description, of any object in the data base. NSMaintain prompts you for the object name, then for a new remark, offering the old remark as default.
Remove Alias	Removes an Alias from the data base. You are prompted for the alias. This has no affect on the primary object for which the removed name is an alias.
Remove User	Undoes the effect of Add User; that is, removes a user name and all its properties from the data base.
Remove Registered Object	Removes a specified object and all its properties from the data base. The primary name and description of the object are printed first and you are asked to confirm the deletion. You can use this to remove groups and other kinds of objects.

Manipulating Groups

A group is a Clearinghouse object with members. Groups are most commonly used for mailing lists and access control. The members can be either individuals or other groups. In the case of groups used for access control, a member can also be a pattern in which "*" usually replaces one or more entire fields of a three-part name.

A group has associated with it two access control lists: *owners* and *friends*. Owners can make any change to a group; they are like domain administrators for the narrow scope of the group itself. Friends are allowed to add or remove themselves from the group. For example, common interest groups typically have "open" membership, consisting of a friends list of "*:domain", or even "*:*:*" for a completely open group.

If the Owners or Friends list is empty, it defaults to the administrators of the domain. However, if the Owners list is non-empty, it *overrides* the administrators list. For example, if you remove yourself from the owners of a group, you can no longer modify the list, even though you are a domain administrator. The defaulting can lead to confusion, especially since the Describe command does not (and unfortunately cannot) indicate whether the owners and friends it displays are explicit or defaulted. For example, if a group previously had no explicit owners, then the Remove Owner command cannot be used, and any use of the Add Owner command implicitly removes all the domain administrators.

Add Group	Creates a new user group. You are prompted for the group's name, a short description of the group, its initial members one at a time, its owners and its friends. NSMaintain checks all of the names except those that are patterns to ensure that you gave valid Clearinghouse names, and to resolve aliases. The Clearinghouse does not actually require that members of a

group be registered Clearinghouse names, as it does not attach explicit meaning to the contents of a group until told to do so. Thus, if you type an invalid name, NSMaintain asks whether you really meant it, and keeps the name if you answer yes. Note, however, that any group used for access control must contain only registered Clearinghouse names or patterns.

If you specify any owners, you are always made an owner yourself as well, whether you explicitly said so or not, so as to avoid the anomaly of your not being able to further modify the group. You can, of course, remove yourself afterward if you really meant to.

Add Member
Add Friend
Add Owner — Adds a member, friend or owner to a group.

Add Self — Adds you, the currently logged in user, to a group. You must be a friend or owner of the group.

Remove Member
Remove Friend
Remove Owner — Removes a specified member, friend or owner from a group.

Remove Self — Removes you, the currently logged in user, from a group. You must be a friend or owner of the group.

Manipulating Domains

These commands change the list of administrators of a domain. You must be an administrator of the domain or the parent organization to do this.

Add Domain Administrator — Adds a user to the set of administrators for a domain.

Remove Domain Administrator — Removes a specified user from the administrators for a domain.

Errors

NSMaintain always ends each command with some sort of feedback about the completion of the operation. In information commands, the feedback is, of course, the requested information. In commands that change the data base, NSMaintain usually prints "done". If a command fails, NSMaintain prints a terse error message. Listed here are some of the more common ones:

NoSuchObject — You asked about a name that does not exist in the Clearinghouse data base. Check that the spelling and the domain are correct.

IllegalOrganization
IllegalDomain
IllegalObject — The name you gave is not legal as a Clearinghouse name. Since NSMaintain already checks for incorrect use of asterisks, this usually means the name is too long. (The name component must be no more than 40 characters long; domains and organizations are limited to 20 characters each.)

Missing — The name you specified for a group, such as in the AddMember command, is not a group.

CredentialsInvalid
VeriferInvalid — You are logged in incorrectly; that is, either your name or your password is incorrect. You can use the Change Login command to log in correctly.

AccessRightsInsufficient — You do not have the authority to make the change you requested. You can find out who does have the authority by using the command Describe for changing a group, or List Domain Administrators for all other changes.

NoChange — The change you requested would have no effect; for example, you added to a group a name that was already a member, or requested to remove a name that was not there.

TooBusy — The Clearinghouse contacted by NSMaintain was too busy to field the request. Lisp's present Clearinghouse implementation, unfortunately, does not handle this error, so passes it along to you. If you repeat the operation it may succeed. If this error persists for a long time, you may want to evaluate (START.CLEARINGHOUSE T) to completely clear the Clearinghouse cache; the system may then succeed in locating a more responsive server.

Examples

In the example session that follows, all user input is in boldface; everything else is typed by the system. To avoid clutter, carriage returns typed by the user are not shown. In many cases, a simple carriage return accepts the default input typed by the system, or completes a partially typed name. For clarity, most of the user input is in uppercase, although lowercase is equally acceptable. Commentary is in italics.

```
64> (NSMAINTAIN)
[Default login:  Arthur Dent:Research:ACME;
 Default domain: Research:ACME]
```

NSMaintain shows me the defaults.
(My password has not, however, been verified.)

```
CH: Describe name: EDISON:Research:ACME ...

Thomas A. Edison:Research:ACME is a User (Electronics Div., Rm 2732)
Aliases: Edison:, Wizard:
```

Domain and organization are elided here

```
Mailboxes: [Time:  8-Aug-86 17:30:54; Mail.Service: (Snail:)]
Userdata: [Last.Name.Index: 10; File.Service: Phylum:]
```

The Userdata property is used by Viewpoint

```
CH: List Groups by pattern: *:Research:ACME ... AllResearch, Consultants,
ED, LispImplementors, LispInterest, NetAdministration, Skiiers, Staff,
WireBusters

CH: Show Details of previously listed names
```

```
  name: Consultants                    "C<cr>" is all that I typed

Consultants:Research:ACME is a User Group (Part-time personnel)
Owners: Staff:
Friends: NetAdministration:

  name: SKiiers

Skiiers:Research:ACME is a User Group (Snow sport enthusiasts)
Owners: UserAdministration:All Areas, Perry White:
Friends: *:*                           Anyone in organization ACME can join

  name:

CH: List Members of group: Skiiers:Research:ACME ...
Alexander G. Bell:Telcom, Christopher Craft:, Staff:

CH: Add Self to group: Skiiers:Research:ACME ...  done
                                       Skiiers was offered as default, being the last
                                       group I mentioned—I had only to type a cr.

CH: Add Self to group: Skiiers:Research:ACME ...  failed: NoChange
                                       I.e., I'm already a member

CH: List Servers of type File
 by pattern: *:Development:ACME ... Arrow, Quiver

CH: List Users by pattern: Ed*:Research:ACME ... (none)
                                       There are no users whose full name starts "Ed"

CH: List ALiases by pattern: Ed*:Research:ACME ... Edison, Educators
                                       But there are some aliases (not necessarily all users)

CH: List objects having property wORKSTATION
 by pattern: *M*:Research:ACME ... Archimedes, Camero, Cardamom,
Homestead, MayDay, Mendel, Ramanujan, SatanicMechanic, TheTajMahal

CH: Show Details of previously listed names
  name: Archimedes

Archimedes:Research:ACME is a Workstation (1186 in Rm. 2732)
Address.List: (6285#0.125101.20200#0)
Authentication.Level: [Simple: true; Strong: false]

CH: Change Default Domain (for name entry) to be: Development:ACME
Set this default globally as well (i.e. for use outside Maintain)? N

CH: Add Alias for object: Newton:Development:ACME
 Alias: Isaac:Development:ACME ...  done

CH: Describe name: Newton:Development:ACME ...

S. Isaac Newton:Development:ACME is a User (Apple Tester, Rm. 34)
Aliases: Isaac:Development, SIN:Development
Userdata: [Last.Name.Index: 0; File.Service: Arrow:Development]
```

```
CH: Remove Alias alias: SIN:Development:ACME ... done, alias was removed
from S. Isaac Newton:Development

CH: Add User
New user's name: Charles S. Brown:Development:ACME ...
Remark (terminate with CR): Test Team captain
Alias: Chuck:Development:ACME
Alias: Brown:Development:ACME
Alias: CSB:Development:ACME
Alias:  xxx                                  Bare <cr> was typed to end the list
Initial password: ******* (retype password) *******...  done
                                             Chuck can later use Change Password to
                                             set a password of his own choosing.

CH: Add Group
New group name: Entomologists:Development:ACME ...
Remark (terminate with CR): Seekers of bugs

Enter names of members, owners and friends, one per line, terminated with
a blank line.

Member: brown:Development:ACME = Charles S. Brown:Development:ACME
                                             NSMaintain resolves alias, so that member
                                             names are in canonical form
Member: F. Kafka:Development:ACME
Member:  xxx

(If you enter no owners, the group will be owned by the administrators of
Development:ACME.)

Owner: brown:Development:ACME = Charles S. Brown:Development:ACME
Owner:  xxx

Friend: *:Development:ACME
Friend: *:Research:ACME
Friend:  xxx

Adding members... done
Adding owners... (including Arthur Dent:Research:ACME) done
                                             I'm an owner, too (else I couldn't modify the group)
Adding friends... done

CH: Remove User: BILBO:Development:ACME ...
Bilbo Baggins:Development:ACME (Furry ring finder)
 Confirm deletion (y or n): Y
 done

CH: Quit [confirm]                           Back to the exec now.
```

[This page intentionally left blank]

Xerox Lisp includes utilities for generating hardcopy in "Press" format. This is a file format for communicating documents to Xerox laser xerographic printers of the Press, Spruce or Fullpress class, which are known by the names Dover, Penguin, Raven and Spruce.

Note: Press, along with other classes of printers, is described fully in two sections in the *IRM*, "Hardcopy Facilities" and "Fonts."

The reason for putting this document into the *Lisp Library Modules Manual* is that support of these printers is no longer an integral part of the system.

Requirements

FONTS.WIDTHS
PUPPRINT.LCOM
A Press print server

Installation

Load PRESS.LCOM from the library.

Set the variable PRESSFONTWIDTHFILES (see below).

User Interface

You can use the usual means:
HARDCOPY in background (right-button) menu,
HARDCOPY in FileBrowser menu, or
functions typed into the exec window (see below).

Functions

DEFAULTPRINTINGHOST	[Variable]
PRESSFONTWIDTHFILES	[Variable]
SEND.FILE.TO.PRINTER	[Function]
LISTFILES	[Function]

(All: see the *IRM* for full details.)

Limitations

PRESS does not support NS characters or NS fonts.

ReadNumber contains functions for implementing a calculator-type menu for entering numbers with the mouse.

Installation

Load READNUMBER.LCOM from the library.

Functions

ReadNumber functions are called either from the Executive window or programmatically from another process.

The numbers captured by ReadNumber are passed to whatever process currently has the TTY.

Create a Key Pad

(RNUMBER *MSG POSITION MSGFONT DIGITFONT INCLUDEABORTFLG FLOATINGPTFLG POSITIVEONLYFLG ACCEPTTYPEINFLG*)

[Function]

Brings up a menu that looks like a ten-key calculator pad. Your selections, made by pressing the left mouse button when the cursor is on a digit, are accumulated in a displayed total. The key pad includes a backspace key (BS), a clear key (CLR), and a +/- key (-). When OK is selected, the total is returned.

If *MSG* is given, it is displayed at the top of the menu.

If *POSITION* is given, the menu will be put there; otherwise it will be put at the cursor.

If *MSGFONT* is given, *MSG* will be printed in it. If *MSGFONT* is NIL, DEFAULTFONT is used.

If *DIGITFONT* is given, the labels on the keys will be printed in that font. If *DIGITFONT* is NIL, BOLDFONT is used.

If *INCLUDEABORTFLG* is non-NIL, the menu will also include an abort key (abt). If the abort key is pressed, RNUMBER returns NIL.

If *FLOATINGPTFLG* is non-NIL, the menu will include a decimal point, and the value returned may be a floating point number.

If *POSITIVEONLYFLG* is non-NIL, the menu will not include a +/- key (-) and you will only be able to input positive numbers (but see *ACCEPTTYPEINFLG*).

If *ACCEPTTYPEINFLG* is non-NIL, the menu will also respond to user-typed input (i.e., numbers typed in on the keyboard, rather than selected with the mouse). In this mode, carriage return corresponds to OK.

Note: The decimal point (.) and the minus sign (-) are also accepted, even though they are not options in the key pad menu.

If you close the key pad window, the action taken by RNUMBER depends upon the value of *INCLUDEABORTFLG*. If *INCLUDEABORTFLG* is NIL, RNUMBER generates an error (i.e., calls (ERROR!)). If *INCLUDEABORTFLG* is non-NIL, RNUMBER returns NIL (the same thing it does if the abort key is pressed).

Create a Key Pad for Repeated Use

For some applications, it may be beneficial to avoid the creation of the key pad menu window each time a number is asked for. The following functions allow you to create a key pad menu window and use it repeatedly to get values from you.

Note: When used in this manner, a key pad menu window can only be used by one process at a time.

(CREATE.NUMBERPAD.READER *MSG WPOSITION MSGFONT DIGITFONT INCLUDEABORTFLG FLOATINGPTFLG POSITIVEONLYFLG*) [Function]

Creates a window suitable for use by NUMBERPAD.READ (see below). Its arguments are the same as for the function RNUMBER.

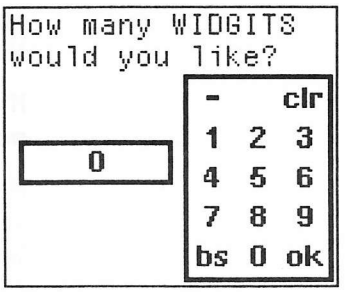
(NUMBERPAD.READ *NUMBERPAD/READER ACCEPTTYPEINFLG*) [Function]

NUMBERPAD/READER should be a window returned by the function CREATE.NUMBERPAD.READER (see above). NUMBERPAD.READ uses the window in the same manner as the function RNUMBER.

Examples

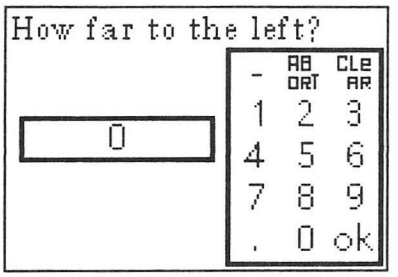
```
(RNUMBER "How many WIDGITS would you like?")
```

will result in the following menu being popped up:



```
(RNUMBER "How far to the left?") NIL '(CLASSIC
12) '(MODERN 14) T T)
```

will result in the following menu being popped up:



Limitations

If you choose both FLOATNGPOINTFLG and INCLUDEABORTFLG, then there is no room for the backspace key, so the input is correctable only by selecting CLEAR and starting over. However, if ACCEPTTYPEINFLG is T, the keyboard's backspace key can be used.

[This page intentionally left blank]

The 1108 and 1186 each support two RS232 ports. One of these ports is configured as Data Terminal Equipment, and is intended to be connected to modems or terminal ports on other computers. (On the 1108, this port is available only with the addition of the E30 option.) The other port is configured as Data Communication Equipment, and is meant to drive printers or terminals. In this document, the DTE port is called the RS232 port, and the DCE port is called the TTY port.

Lisp provides a stream-oriented interface to the RS232 hardware. Users' programs can open streams to the hosts {RS232} or {TTY}, and perform input or output using standard Lisp I/O functions, such as READ-BYTE, READ-CHAR, etc.

Programs may use RS232 streams or TTY streams with the same programmatic interface; however, the RS232 port is preferred over the TTY if the application expects to handle large amounts of input data. In the 1108 and 1186, data entering the RS232 port is buffered independently of Lisp by the I/O processor (IOP). In addition, the RS232 software provides an additional layer of character buffering, freeing user programs from having to monitor the RS232 hardware frequently.

No independent buffering is provided for data entering the TTY port. As a result, Lisp cannot guarantee to catch all characters received on this port. For this reason, the TTY port should be used primarily to drive output devices such as printers.

Requirements

Hardware

To connect to a modem or another device, you need an RS232 or TTY cable (see the wiring diagrams in the Introduction of this manual).

For the 1108 only, you need the E-30 upgrade kit.

For the RS232 port to operate correctly, the remote device must assert the standard RS232 signals DSR and CTS. This requirement means that the appropriate pins on the 1186 be driven high, either properly by connecting the 1186 side to the corresponding device side, or permanently by jumpering the pins.

In order for the TTY port to operate correctly, the remote device must assert DTR and RTS. This requirement means that the appropriate pins on the 1186 be driven high, either properly by connecting the 1186 side to the corresponding device side, or permanently by jumpering the pins.

Software

You need the following .LCOM files in order to run this module successfully:

DLRS232C or DLTTY (one of these is required)

RS232MENU

RS232CHAT or TTYCHAT, and KERMIT (these are optional).

(See also the file dependencies enumerated in the Introduction of this manual.)

Installation

Load the required .LCOM modules from the library.

Run RS232C.INIT or TTY.INIT to set parameters.

Using the RS232 Port

Support for the RS232 port is contained in the file DLRS232C.LCOM. Before using the RS232 port, it is necessary to initialize the RS232 hardware. The function RS232C.INIT is provided for this purpose:

(RS232C.INIT *BAUDRATE BITSPERSERIALCHAR PARITY NOOFSTOPBITS FLOWCONTROl*)

[Function]

The arguments correspond to the port parameters (see below).

Alternatively, the *BAUDRATE* argument can be an instance of the RS232C.INIT record. If *BAUDRATE* is NIL, the value of the global variable RS232C.DEFAULT.INIT.INFO is used in its place. This provides a means of automatically initializing the RS232 hardware without user intervention.

RS232C.DEFAULT.INIT.INFO [Variable]

This variable controls default initialization of the RS232 port. Its value may be set in the site INIT.LISP file, or in your INIT.LISP file. If RS232C.DEFAULT.INIT.INFO is not set when the RS232 module is loaded, its fields will be set to the following default values:

BaudRate: 1200
BitsPerSerialChar: 8
Parity: NONE
NoOfStopBits: 1
FlowControl: XOnXOff

Programs may use the Lisp function OPENSTREAM as an alternative to calling RS232C.INIT directly, with the parameters bundled up into the PARAMETERS argument.

For example, (RS232C.INIT 9600 8) can also be achieved by:

```
(OPENSTREAM '{RS232} 'INPUT NIL '((BaudRate
9600) (BitsPerSerialChar 8))
```

(RS232C.SET.PARAMETERS *PARAMETERLIST*) [Function]

This function allows applications to change the settings of the RS232 hardware while the RS232 port is in use.

PARAMETERLIST is an association list of parameter names and values. The following example sets the baud rate to 9,600 baud, and the character length to eight bits:

```
(RS232C.SET.PARAMETERS '((BaudRate . 9600)
(BitsPerSerialChar . 8)))
```

The following is a list of legal parameter names and values:

BaudRate 1186: 50, 75, 110, 150, 300, 600, 1200, 2400, 3600, 4800, 7200, 9600, 19200.

1108: all of the above, and 28800, 38400, 48000, 56000, 57600.

BitsPerSerialChar 5, 6, 7, 8 bits of data. If 5 or 6 bits of data are sent, they should be DATA, not CHARACTER.

Parity NONE, ODD, EVEN (1 parity bit).

NoOfStopBits 1, 1.5, 2 (i.e., the stop bit may have a period equal to 1, 1.5 or 2 bit-widths).

FlowControl NIL, NONE, XOnXOff, list.

NIL and NONE: no flow control.

XOnXOff: flow control using Xon and Xoff characters. For applications requiring XOn and XOff characters other than ↑Q and ↑S respectively, this parameter may be supplied as a list in the form: (1 <XOn> <XOff>), where <XOn> and <XOff> represent the character codes (ASCII 1 - 127) of the characters which are to be treated as the XOn and XOff characters. The leading "1" signifies that flow control should be enabled; a leading "0" will program the RS232 port with the appropriate XOn and XOff characters, but leave flow control disabled.

ModemControl This parameter should be a list of modem control signals to be enabled, such as DSR, CTS, DTR, DTR, RI and HS.

The functions RS232MODEMCONTROL, RS232MODEMSTATUSP, and RS232MODIFYMODEMCONTROL provide finer control over the settings of modem signals (see below).

DTR This parameter enables or disables the data terminal ready signal; it may be specified as T or NIL.

RTS This parameter enables or disables the request to send signal; it may be specified as T or NIL.

(RS232C.GET.PARAMETERS *PARAMETERLIST*) [Function]

The current settings for the RS232 port may be obtained at any time by calling this function.

PARAMETERLIST should be a list of parameter names. RS232C.GET.PARAMETERS returns an association list of parameter names and values, in a format acceptable to RS232C.SET.PARAMETERS.

(RS232C.SHUTDOWN) [Function]

The RS232 port is turned off by calling this function. It disables the RS232 port and closes any open streams on the devices.

Using RS232 Streams

Programs may open streams to the RS232 port by calling OPENSTREAM with the file name {RS232}. The *ACCESS* argument to OPENSTREAM controls whether an INPUT or OUTPUT stream is returned. RS232 streams are unidirectional; to obtain a second stream open for the opposite access, call the function RS232C.OTHER.STREAM.

Only one pair of RS232 streams may be open at a time; an error will result if you attempt to open more.

(RS232C.OTHER.STREAM *STREAM*) [Function]

STREAM should be an RS232 stream. If *STREAM* is open for INPUT, an RS232 stream open for OUTPUT is returned; conversely, if *STREAM* is open for OUTPUT, an RS232 stream open for INPUT is returned.

The following Lisp functions are defined to work on RS232 streams open for the appropriate access: BIN, BOUT, READP, OPENP, CLOSEF, and FORCEOUTPUT.

(RS232C.CLOSE-STREAM *DIRECTION*) [Function]

This function closes one or both RS232 streams, so you don't need to have access to the streams to close them.

DIRECTION can be one of INPUT, OUTPUT, BOTH, or NIL. The function closes the RS232 stream open in *DIRECTION* mode; if *DIRECTION* is BOTH or NIL the input and output streams will be closed, if they exist.

RS232 streams are buffered. Input and output are performed in units of packets of data. The I/O processor collects incoming data into a packet, and makes that packet available to Lisp when one of the following conditions is true:

The packet is filled.

The frame time-out has expired.

The frame time-out is the number of hundredths of a second that are allowed to occur between the reception of single characters. This parameter is automatically set by the RS232 module. Its value depends on the baud rate of the RS232 port. If the value is set too large, interactive applications such as Chat will suffer from uneven typeout; if the value is too small, a

larger number of shorter packets may be exchanged between Lisp and the I/O processor, resulting in increased processing overhead.

Lisp buffers data for output in packets of up to 578 characters. Output packets are sent to the RS232 port when one of the following conditions is true:

The current output packet is full.

The user program calls the FORCEOUTPUT function to force the current output packet to be sent.

The output stream is closed.

Applications that generate a large amount of output slowly may wish to reduce the size of outgoing packets. Although this will require additional processing overhead, it will cause output to occur more frequently without the program explicitly calling FORCEOUTPUT.

(RS232C.OUTPUT.PACKET.LENGTH *NEWVALUE*) [Function]

This function returns the current setting of the variable that controls the maximum size of output packets. If *NEWVALUE* is supplied, the setting is changed to *NEWVALUE*. *NEWVALUE* may be a number between 1 and 578.

Specifying a value is a matter of how long you are willing to wait at input time before you are assured of seeing incoming data. You can do a straightforward division to set the length (e.g., at 9600 baud, you get about 1 char/millisecond, so the max delay is 578ms; if you can tolerate only 1/4 second, then set it to 250 or so).

(RS232C.READP.EVENT *STREAM*) [Function]

Many RS232 applications are time-dependent. File transfer protocols such as Kermit and Modem depend on one or both sides of a file transfer detecting connection problems by means of time-outs. This function allows user programs to detect time-out conditions efficiently.

STREAM should be an RS232 stream open for input. This function returns an event that a program may wait upon for input data to become available on the stream. The following example illustrates how a program could wait up to 10 seconds for a character to become available:

```
[LAMBDA (STREAM)
      (LET ((EVENT (RS232C.READP.EVENT STREAM))
      (TIMER (SETUPTIMER 10000)))
      (until (OR (READP STREAM) (TIMEREXPIRED? TIMER))
      do (AWAIT.EVENT EVENT TIMER T)
      finally (RETURN (COND ((READP STREAM) (BIN STREAM]
```

Using Modems

The following functions are useful for controlling modems:

(RS232SENDBREAK *EXTRALONG?*) [Function]

This function sends the out-of-band BREAK signal, for a period of 0.25 seconds; if *EXTRALONG?* is non-NIL, then the period is extended to 3.5 seconds.

(RS232MODEMCONTROL *SIGNALSONLST*) [Function]

This function is a Lambda-NoSpread function that sets the modem control lines to be "on" for the signals named in the list *SIGNALSONLST*; it returns the former setting of the lines. If *SIGNALSONLST* is not supplied (which is not the same as supplying NIL), then the control lines remain unchanged. The entries in *SIGNALSONLST* are symbol names for standard modem control lines; currently usable signal names are DTR and RTS.

(RS232MODIFYMODEMCONTROL *SIGNALSONLST SIGNALSOFFLST*) [Function]

Changes only those modem control lines specified in the union of the two arguments; those in *SIGNALSONLST* are set to be on, and those in *SIGNALSOFFLST* are set off. Returns the former state just as (RS232MODEMCONTROL) does.

(RS232MODEMSTATUSP *SPEC*) [Function]

Returns non-null if the reading of the modem status lines is consistent with the boolean form specified by *SPEC*; modem status signals currently supported are DSR, RI, and RLSD. *SPEC* may be any AND/OR/NOT combination over these signal names.

Example:

```
(RS232MODEMSTATUSP '(AND CTS (NOT RLSD))).
```

(RS232MODEMHANGUP) [Function]

This function takes whatever steps are appropriate to cause the modem to hang up. Generally, this means turning the DTR signal down for about three seconds, or until the DSR signal has gone down.

Error Condition Reporting

The RS232 port detects parity errors, character framing errors, lost characters, and a number of other unusual conditions. As the I/O processor delivers each input packet to Lisp, it reports when the packet was received without error. If an error did occur while the packet was being received, Lisp will report this fact by writing a message to RS232C.ERROR.STREAM.

RS232C.ERROR.STREAM [Variable]

RS232 error conditions are reported on this stream. This stream is initially the PROMPTWINDOW.

(RS232C.REPORT.STATUS *NEWVALUE*) [Function]

There are circumstances in which the RS232 hardware believes it has encountered an error, when in fact it has not. A frequent

cause is an incorrect parity setting in the RS232 port. Continually reporting RS232 errors is likely to slow RS232 processing severely. In cases where error reporting is not important, it is possible to disable error reports with the RS232C.REPORT.STATUS function. This function returns the current setting of status reporting, which may be one of the following:

T Errors are reported on both input and output.

NIL Errors are never reported.

OUTPUT Errors are reported on output only.

INPUT Errors are reported on input only.

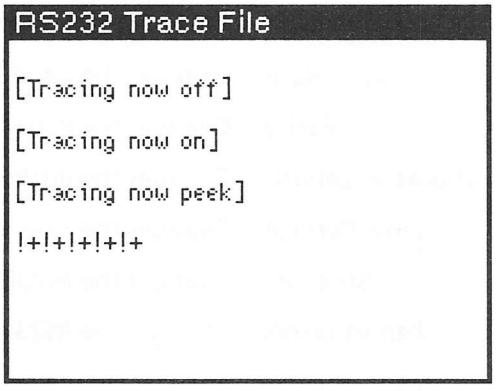
In addition, if *NEWVALUE* is supplied, the current setting of status reporting is changed to *NEWVALUE*.

RS232TRACE

To help in debugging RS232 applications, it is possible to trace the data that is being sent out via the port. RS232 packets are traced using:

(RS232C.TRACE *MODE*) [Function]

MODE is one of PEEK, T, or NIL. If MODE is either T or PEEK, RS232C.TRACE will open a trace window in the mode selected. T indicates a full trace, with every byte being shown. PEEK is a less verbose trace, with every incoming packet shown as a " + " and every outgoing packet shown as a "!". NIL will turn off the tracing. Clicking the left mouse button in the trace window will cycle between the modes.

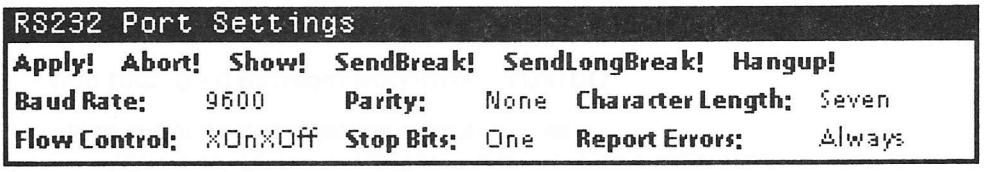


RS232CHAT

The RS232CHAT module is a facility that permits the Chat library module to communicate over the RS232 port. Once it is loaded, you may chat to the host named RS232 to open a connection to the RS232 port.

RS232CMENU

RS232CMENU is a utility that provides a menu interface to controlling the settings of the RS232 port. When loaded along with RS232CHAT, this utility may be invoked by choosing the "Set Line Parameters" entry in the options menu that appears if you select "Options" in the middle-button Chat menu.



The following commands are available in the RS232 menu:

Apply — This menu item changes the RS232 port settings according to the values of the fields in the rest of the menu. No changes are made to the RS232 hardware until this command is given.

Abort — Closes the RS232 menu and aborts any changes that could have been made.

SendBreak — Sends a normal (0.25 second) break signal. Requires confirmation, as this command could cause the modem connection to be broken.

SendLongBreak — Sends a long (3.5 second) break signal. As above, this requires confirmation.

Hangup — Tries to hang up the modem (by dropping DTR). Requires confirmation.

The following fields are multiple choice items; when their labels are selected with the mouse, a menu of possible values for the field appears. Choosing a value from the menu will change the field; clicking off the menu will leave the field unchanged.

Baud Rate — Changes the BaudRate of the RS232 port.

Parity — Changes the Parity setting of the RS232 port.

Character Length — Changes the BitsPerSerialChar setting of the RS232 port.

Flow Control — Changes the FlowControl setting of the RS232 port.

Stop Bits — Changes the NoOfStopBits setting of the RS232 port.

Report Errors — Changes the RS232C.REPORT.STATUS setting of the RS232 port.

File Transfer Using RS232

Files may be transferred using RS232CHAT and either the Kermit or Modem protocols.

Using the TTY Port

Support for the TTY port is contained on the file DLTTY.LCOM. The TTY port is designed to support low-speed communications with RS232 devices. The I/O processor offers no low-level support for input buffering. As a result, it is quite possible for newly received input characters to overwrite previously received but unread characters in the input hardware. The TTY port provides exactly one character's worth of buffering; each character must be read by Lisp before the next character is completely received.

Note: No hardware flow control is provided on the TTY port. The Lisp TTY port service routines will obey received flow control commands, but will not generate flow control commands in response to increased input data rate.

(TTY.INIT *BAUDRATE BITSPERSERIALCHAR PARITY NOOFSTOPBITS FLOWCONTROL*) [Function]

This function is very similar to the function RS232C.INIT.

Before using the TTY port, the TTY hardware must be programmed with the proper characteristics for your application.

Alternatively, the *BAUDRATE* argument can be an instance of the RS232C.INIT record. If *BAUDRATE* is NIL, the value of the global variable TTY.DEFAULT.INIT.INFO is used in its place. This provides a means of automatically initializing the TTY port hardware without user intervention.

TTY.DEFAULT.INIT.INFO [Variable]

This variable controls default initialization of the TTY port. Its value may be set in the site INIT.LISP file, or in your INIT.LISP file. If TTY.DEFAULT.INIT.INFO is not set when the TTY package is loaded, its fields will be set to the following default values:

BaudRate: 1200
BitsPerSerialChar: 8
Parity: NONE
NoOfStopBits: 1
FlowControl: XOnXOff

Programs may use OPENSTREAM as an alternative to calling TTY.INIT directly, with the parameters bundled up into the PARAMETERS argument as shown below.

For example:

```
(OPENSTREAM '{TTY} 'BOTH NIL '((BaudRate 9600)
(BitsPerSerialChar 8))
```

(TTY.SET.PARAMETERS *PARAMETERLIST*) [Function]

Applications may change the settings of the TTY hardware while the TTY port is in use.

PARAMETERLIST is an association list of parameter names and values. For example, let's set the baud rate to 9600 baud and the character length to eight bits:

```
(TTY.SET.PARAMETERS '((BaudRate . 9600)
(BitsPerSerialChar . 8)))
```

The following is a list of legal parameter names and values:

BaudRate — 50, 75, 110, 150, 300, 600, 1200, 2400, 3600, 4800, 7200, 9600, 19200.

BitsPerSerialChar — 5, 6, 7, 8. If 5 or 6 bits of data are sent, they should be DATA, not CHARACTERS.

Parity — NONE, ODD, EVEN (1 parity bit).

NoOfStopBits — 1, 2 (This parameter should be 1 except at 110 baud).

FlowControl — NIL, XOnXOff. For applications requiring XOn and XOff characters other than ↑Q and ↑S respectively, this parameter may be supplied as a list in the form:(1 <XOn> <XOff >), where <XOn> and <XOff> are replaced by the character values of the XOn and XOff characters. The leading 1 signifies that flow control should be enabled; a leading 0 will program the TTY port with the appropriate XOn and XOff characters, but leave flow control disabled.

Note: XOnXOff flow control is known to be reliable only up to 4800 baud.

DSR — This parameter enables or disables the data set ready signal; it may be specified as T or NIL.

CTS — This parameter enables or disables the clear to send signal; it may be specified as T or NIL.

(TTY.GET.PARAMETERS *PARAMETERLIST*) [Function]

The current settings for the TTY port may be obtained at any time by calling this function.

PARAMETERLIST should be a list of parameter names. TTY.GET.PARAMETERS returns an association list of parameter names and values, in a format acceptable to TTY.SET.PARAMETERS.

(TTY.SHUTDOWN) [Function]

This function turns off (disables) the TTY port and closes any open streams on the device.

Using TTY Streams

Programs may open streams to the TTY port by calling OPENSTREAM with the file name {TTY}. The ACCESS argument to OPENSTREAM may be INPUT, OUTPUT, or BOTH.

Unlike RS232 streams, TTY port streams are not buffered, and a single stream may be used for both input and output. The generic Lisp input/output functions BIN, BOUT, READP, OPENP, and CLOSEF may be used on TTY port streams.

TTYCHAT

TTYCHAT is a module that enables the Chat library module to communicate over the TTY port. No user-callable functions or user-settable variables are available in the TTYCHAT module. Once it is loaded, you may chat to the host named TTY to open a connection to the TTY port. TTYCHAT is contained on the file TTYCHAT.LCOM.

Because the TTY port does no low-level input buffering, it is quite likely that many input characters will be lost while chatting at 1200 baud or higher. This package should only be used for non-critical applications, such as testing connections between the TTY port and low-speed printers.

RS232CMENU

RS232CMENU is compatible with the TTY port as well. Certain RS232 port commands, such as SendBreak! are not available with the TTY port, and hence do not appear in the menu.

Examples

Testing the Connection Between Two Xerox Lisp Machines

To test for a working RS232 connection between Machine A (a Xerox 1108) and Machine B (a Xerox 1100, 1108, or 1186) by moving a file between them, proceed as follows:

Load RS232CHAT.LCOM on both machines.

Call RS232C.INIT to set up parameters.

Do RS232CHAT on both machines. You will be prompted for a window.

Whatever you type on machine A should be echoed on machine B and vice versa.

Testing the Connection Between Xerox Lisp Machines and a VAX Running VMS

VAX side: Set baud rate at which files will be transferred on the VAX side using the VMS command SET. For example, to set to 1200 baud, type:

```
SET TERM TTA1:/SPEED=1200/PERM
```

1108 side: Load RS232CHAT.LCOM

Initialize: `(RS232C.INIT 1200 8 NIL 1 NIL'RS232C)`

Then call `(RS232CHAT)`

You should be able to use the Chat window like a VAX teletype terminal.

By matching the baud rate on the VAX (through SET TERM) with that on the 1108 (through RS232C.INIT), you can use any speed up to 9600 baud.

SameDir modifies MAKEFILE to guard against inadvertently writing out a file onto a directory other than the one it came from.

SameDir adds the form (CHECKSAMEDIR) to MAKEFILEFORMS. It compares the (HOST&DIRECTORYFIELD *OLDFILE*) against (DIRECTORYNAME T T) to see whether the connected directory matches the old file's source.

Installation

Load SAMEDIR.LCOM from the library.

User Interface

If you do a MAKEFILE and you are connected to a directory that is not listed in the FILEDATES property of the file, and the file has a FILEDATES property at all (i.e., this isn't a brand new file), the system will prompt you with:

You haven't loaded or written TORTOISE in your connected directory {server}<user> should I write it out anyway?

Your options are reply with Y, N, C, or O:

Y Yes, do the MAKEFILE

N No, abort the MAKEFILE

C Connect to other directory: allows you to type in another path.

O Oops! Connect to the best guess; i.e., the directory where the file was last loaded or written. This option requires confirmation, in case you don't like the directory that the system prompts you with.

The default answer to the question is Y (do the MAKEFILE).

When comparing directory names, SameDir ignores case differences between the old and new directory names.

MIGRATIONS [Variable]

For those who regularly LOADFROM files on one directory and MAKEFILE elsewhere, the variable MIGRATIONS can be set to keep SameDir from asking too often. It is an association list containing pairs of (*OLDDIR* . *NEWDIR*), which specifies which migrations are allowable.

For example, if it is legitimate to LOADFROM a file on {MYHOST}<PUBLIC> and then do a MAKEFILE to {MYHOST}<TEST>, then adding ({MYHOST}<PUBLIC> . {MYHOST}<TEST>) to MIGRATIONS will prevent MAKEFILE from complaining about such movement.

Limitations

For Unix hosts using the PUP FTP protocol, there is sometimes an inconsistency between the directory name in the full file name and the directory name in DIRECTORYNAME. SameDir may have trouble in that case detecting that the directories are the same.

Spy is a tool to help you make programs run faster by giving you a picture of where the program is spending its time.

Description

Spy has two parts: a sampler and a displayer. The sampler runs while your program is running, and it monitors what your program is doing. The displayer displays the data gathered by the sampler.

The sampler periodically interrupts the running program to account the functions in the current call stack. This allows Spy to remember not only (proportionally) how long is spent in each individual function, but also how long each function is seen on the call stack. The sampler data structures minimize interference with the normal running of the program — there is little noticeable performance degradation. Spy doesn't log every call and return (it only samples), so you can run it even over long computations without fear of overflowing storage limits.

The displayer uses the Grapher module to display the data gathered by the sampler. In the graph, the height of each node is adjusted to be proportional to the amount of time. Just as MasterScope and Browser give an interactive picture of the static structure of the program, Spy gives an interactive picture of the dynamic structure. The displayer is interactive as well as graphic. That is, you can look at the data in a variety of ways, since it seems there is no one picture that says it all. Since the displayer knows the whole call graph, it can show the entire tree structure, with separate calls to a function accounted separately, or merge separate calls to the same functions. Since the sampler records the entire calling stack when it samples, it can account for both individual and cumulative time. When the sampler runs, if a function is on the top of the stack, it adds to its individual total; if the function is on the stack at all, the sampler adds to the cumulative total.

When there are several calls to the same function within the graph, the displayer can either merge the nodes (show the total time for the function in one node) or not. If a node is merged, then one of the boxes in the graph will have all of the time for that function accounted to it, and the rest will be left as ghost boxes. Spy has a variety of ways of controlling which nodes will be merged.

Requirements

GRAPHER
READNUMBER
IMAGEOBJ

Installation

Load SPY.LCOM and the required .LCOM modules from the library.

User Interface

(SPY.BUTTON *POS*) [Function]

This function puts up a little window, which you can use to turn Spy on and off. If *POS* is NIL, you can drag the window with the mouse. If *POS* is specified (in the format xxx . yyy) then the window is placed at those coordinates.

When Spy isn't watching, it looks like this:

Left-clicking (pressing the left mouse button) on it once will turn on sampling, and the window will look like this:

Clicking on it again will turn off sampling, and display the results (SPY.TREE 10). This is the simplest way of spying on operations. (See SPY.TREE below.)

Note: You can't turn off sampling if the mouse process is locked out.

(SPY.START) [Function]

Reinitializes the internal Spy data structures, and turns on sampling.

(SPY.END) [Function]

Turns off sampling, and cleans up the data structures in preparation for the display phase performed by SPY.TREE.

(SPY.TOGGLE) [Function]

If Spying is off, turn it on with (SPY.START). If it is on, turn it off with (SPY.END) and then show the results with (SPY.TREE 10).

It is reasonable to use this with an interrupt character; e.g.,

```
(INTERRUPTCHAR (CHARCODE ↑C) '(SPY.TOGGLE) T)
```

will enable control-C (or any other character you specify) as an interrupt which will turn spying on and off, the same as clicking on the Spy button. (If the Spy button is visible, it will respond to the interrupt.) Then, a control-C will turn on spying, and another one will turn it off.

(WITH.SPY *FORM*) [Macro]

Calls (SPY.START), evaluates *FORM*, calls (SPY.END), and then returns the value of *FORM*.

For example,

```
(WITH.SPY (LOAD 'FOO))
```

is basically

```
(PROGN (SPY.START) (PROG1 (LOAD 'FOO) (SPY.END].
```

(SPY.TREE *THRESHOLD INDIVIDUALP MERGETYPE DEPTHLIMIT*) [Function]

SPY.TREE displays the results of the last SPY sampling in a Grapher window. There are a number of parameters that control the display, which you can either set when you call SPY.TREE, or set interactively with the menu. You normally just use (SPY.TREE) and the menus.

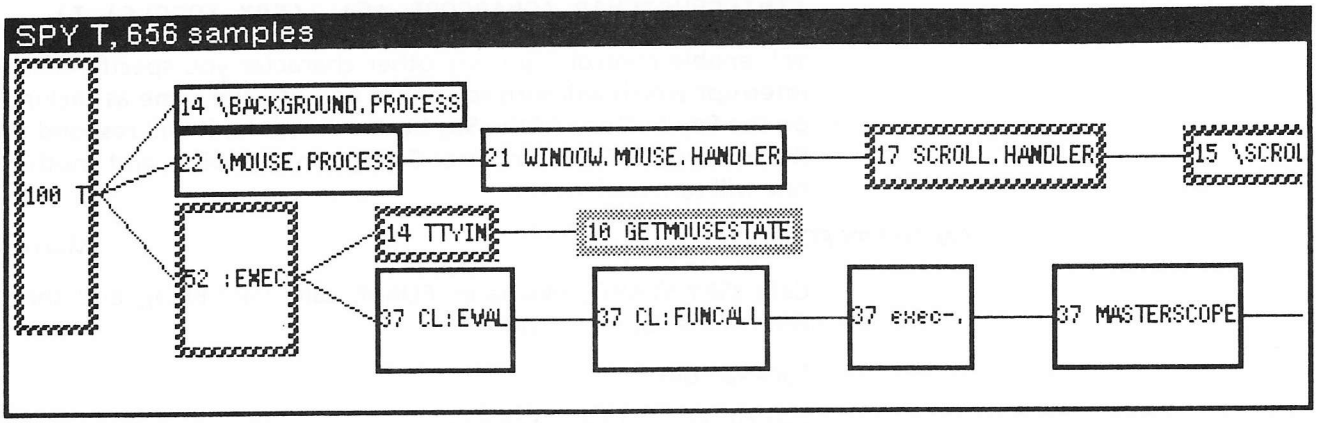
THRESHOLD is a percentage (defaults to 0). If a function's contribution to the total elapsed time is lower than the threshold percentage, it is not displayed.

INDIVIDUALP is either NIL or T. This controls whether cumulative or individual percentages are displayed. The default is cumulative, in which case a function is charged for the time spent in that function and all subfunctions; if individual, a function is only charged for the time spent in that function alone.

MERGETYPE is one of (NONE, ALL, DEFAULT). This controls accounting for functions that appear in several places in the calling tree. Mergetype ALL indicates the total time spent for all calls to the same function, regardless of where it appears. Mergetype NONE indicates the times separately for each instance of the function. Mergetype DEFAULT is the same as NONE except for recursive functions.

DEPTHLIMIT is a number (defaults to NIL = arbitrary depth; not completely debugged for other values).

You will get a prompt to open a window, and then a graph will appear in it, something like this:



In this example, 100% of the time was spent under the top frame, T. That time was then divided up among three processes: \BACKGROUND.PROCESS, \MOUSE.PROCESS, :EXEC. The numbers to the left of the label are the percentages. The height of the box is proportional to the percentage (except that it is always made big enough to hold the label). The width isn't significant; it is just wide enough to hold the name of the function

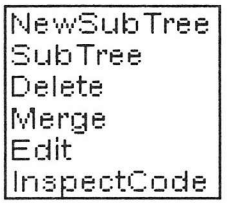
You can point at any of the nodes.

Right Button Operation

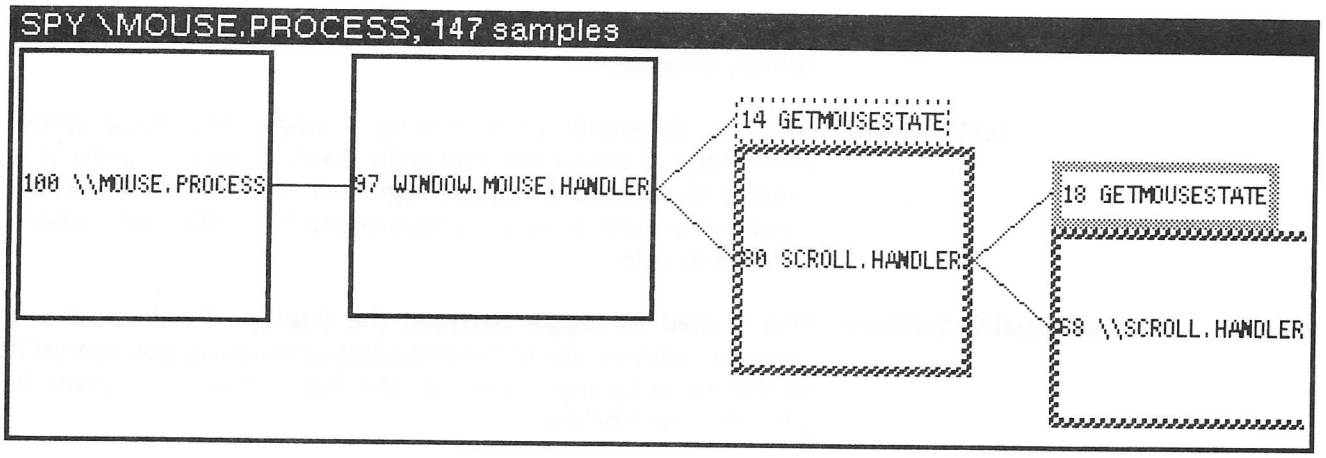
If you right-click on a node, the window title will change to show the function name, and the individual and cumulative percentages. The first number is the individual total and the second is the cumulative. If the right-click is not on a node, the title will change to show the name of the top frame and the total number of samples.

Left/Middle Button Operations — on a Node

If you left-click, or middle-click on a node, you will get a menu:



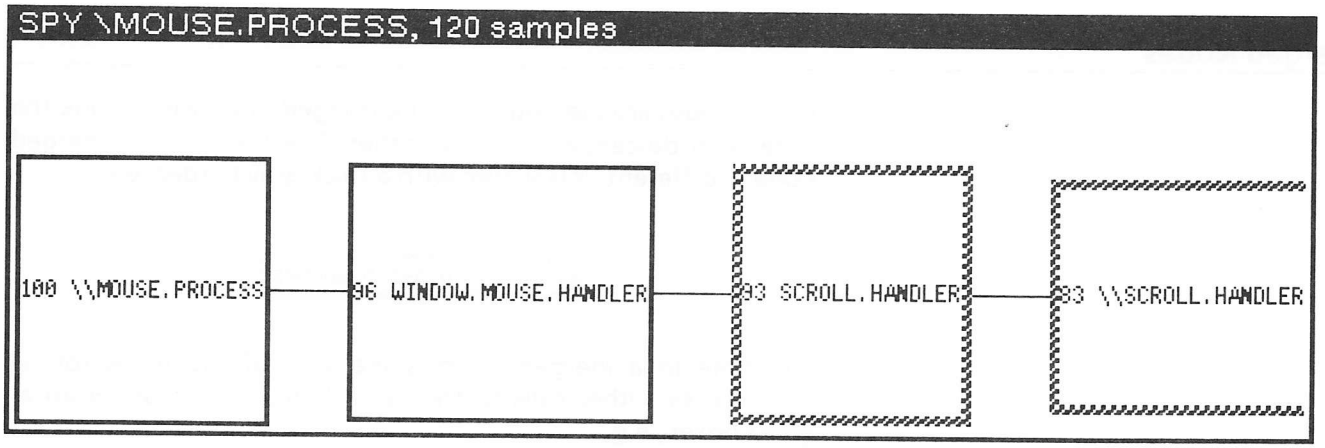
NewSubTree Creates another Spy window that includes data only from this node and its descendents (with the tree rooted at the selected node). Suppose that you were only interested in the actions that you had invoked with the mouse. You can left-click \MOUSE.PROCESS and select the NewSubTree option. You then get a picture like this:



SubTree Behaves just like NewSubTree except that Spy will reuse the same window.

Delete Removes the selected item (and its subbranches) from consideration in this window, and redisplays.

For example, if you don't want to consider GETMOUSESTATE in the graph at all, you can delete the GETMOUSESTATE box and get:



The percentage for the SCROLL.HANDLER changed from 80% to 93% when GETMOUSESTATE was deleted.

Merge Allows you to merge a node with its caller everywhere in the tree.

Edit Invokes SEdit for the function in the node.

InspectCode Shows the compiled code for the function in the node.

Left/Middle Button Operation — Not on a Node

If you press the left, or middle button while not on a node, you get a menu that will let you view or change the parameters for the Spy window:

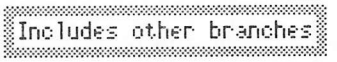
Legend	Displays SPY border interpretation.
Inspect	Allows for the inspection of the current parameter settings for the Spy window.
SetThreshold	Sets the threshold for displaying a node. Any node whose percentage is below the threshold won't show up (unless it is needed to connect the graph together). You can set the initial threshold via the threshold argument to SPY.TREE; otherwise it defaults to zero.
Individual/Cumulative	This is used to toggle between the display of individual and cumulative times. The initial default is cumulative; you can set it to Individual by supplying T as the INDIVIDUALP argument to SPY.TREE. (See below).
MergeNone/MergeAll/MergeDefault	This controls merging of nodes; see below.

Ctrl-Left/Ctrl-Middle Button Operations

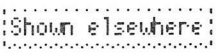
If you press the left, or middle button while the control key is down, you will get the same menu, but the action will be deferred until you next use the left/middle button. For example, you can delete several nodes and then do one update.

Merged Nodes

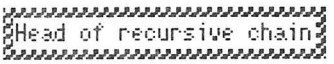
If two nodes are merged, then the merged node will include the time (and descendents) of the other. The display of a merged node is different; it is shown with a thick gray border; e.g. ,



The time in a merged node is the sum of the times for all occurrences. Other calls to the same function may show up as ghost boxes; e.g.

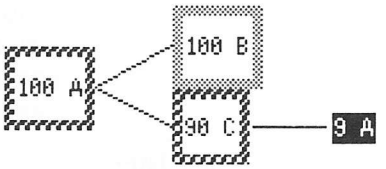


The case where a function is merged with a recursive call to itself is handled specially: the head of the recursion is marked with a wider checkered border:



and the tail of the recursion is shown in reverse video:

In the recursive case, you can find a situation like this:



In this case, A calls B and C, and C calls A. All of the time is really spent in B, although only 10% is due to the call from the top-level, and 90% is under a call to C, which called A, which called B. In this situation, C is also recursive, of course, and also has the recursive border. If you find this display confusing, try the MERGENONE option and see if you get a clearer picture.

MERGEDEFAULT means to merge any function that is not in SPY.NOMERGEFNS, initially set to (SI::*UNWIND-PROTECT* CL:EVAL \\EVAL-PROGN \\INTERPRET-ARGUMENTS \\INTERPRETER \\INTERPRETER1 ERRORSET \\EVAL \\EVALFORM APPLY \\PROGV EVAL).

MERGENONE means not to merge at all.

MERGEALL means to merge any two nodes for the same function.

The default for Individual mode is MERGEALL. The default for Cumulative mode is MERGEDEFAULT.

Individual and Cumulative Modes

Spy initially comes up with the height of the boxes showing the amount of time the function was on the current call stack. This is called cumulative mode, since each function gets the time that both it and the functions it calls account for. There is another kind of display, called Individual, in which the boxes are proportional to the amount of time the function was on the top of the stack.

One thing to watch for: when you switch between Individual and Cumulative modes, the threshold stays the same. Sometimes the threshold for Individual needs to be higher; otherwise, functions will tend to disappear in the Individual tree. Also, switching to Individual mode also changes to MERGEALL, while switching to Cumulative changes you to MERGEDEFAULT.

(SPY.LEGEND) [Function]

If you forget what the different shadings and borders mean, this function brings up a window that shows what they mean; i.e., it shows the interpretation of SPY.BORDERS or the other internal controls.

SPY.FREQUENCY [Variable]

How many times per second to sample? Initially set to 10. (Maximum 60).

SPY.NOMERGEFNS [Variable]

Functions on this list won't get merged under MergeDefault. Includes (SI::*UNWIND-PROTECT* CL:EVAL \\EVAL-PROGN \\INTERPRET-ARGUMENTS \\INTERPRETER \\INTERPRETER1 ERRORSET \\EVAL \\EVALFORM APPLY \\PROGV EVAL). You may need to add more.

SPY.TREE [Variable]

This variable (same name as the function) is used to hold the data from the last sampling. You can save it and restore it using UGLYVARS (see *IRM*).

SPY.BORDERS [Variable]

Used to control the border display on a tree. This is a list of (*NODETYPE DESCRIPTION BORDERWIDTH TEXTURE INTERIORTEXTURE*).

SPY.FONT [Variable]

Font used to display node labels. Initially GACHA 10.

SPY.MAXLINES [Variable]

Maximum height of a node in the graph, measured in multiples of the font height of SPY.FONT.

Limitations

Spy doesn't know anything about the interpreter or the internal workings of Lisp. Internal functions that are not REALFRAMEP and don't normally show up on BT backtraces (but do on BT!) will be shown in Spy. This includes things like \INTERPRETER1, which will appear underneath any interpreted function call. Thus Spy does not distinguish between frames that are interesting or not interesting to the user.

There are many system-internal data type declarations that don't usually appear in the sysout. For example, most people don't use the internal fields of strings or streams, so their definitions aren't included in the sysout. However, there is a collection of definitions that are used by people who are working on system-level code, definitions that are widely relied upon by more than one system source file.

SysEdit is provided for users who work with the system sources or who write system-level code — it brings in the frequently-used definitions.

Requirements

MASTERSCOPE
EXPORTS.ALL
CMLARRAY-SUPPORT

Installation

Lisp programming environment.

Definitions

The declarations and definitions are provided in the source listings of SysEdit and in the files it loads.

Limitations

SEDit internal definitions are not included, nor are declarations that are used only within a single system source file.