SPY

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 oly 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 turns on sampling, and the window looks like this:

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

Note: You cannot 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)
```

enables Control-C (or any other character you specify) as an interrupt which turns spying on and off, the same as clicking on the Spy button. (If the Spy button is visible, it responds to the interrupt.) Then, a Control-C turns on spying, and another one turns 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.

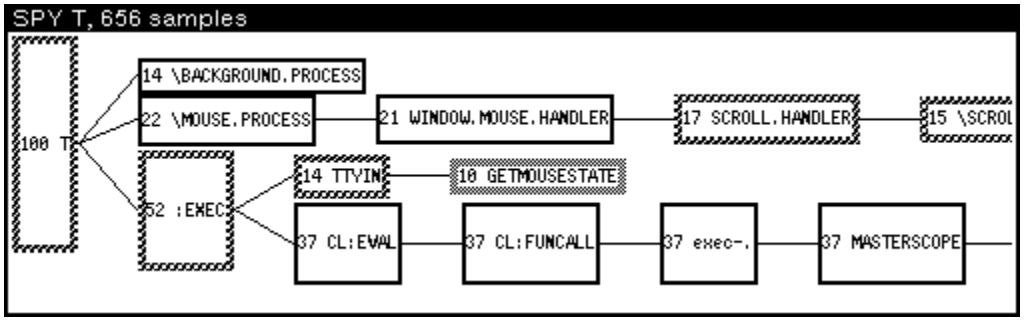
THRESHOLD is a percentage (defaults to 0). If a function's contribution to the total elapsed time is lower than the threshold percentage, it is not displayed.

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.

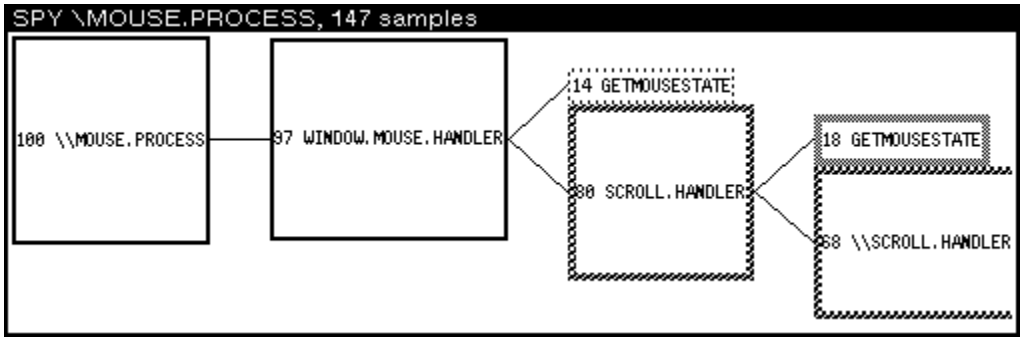
Right Button Operation

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:

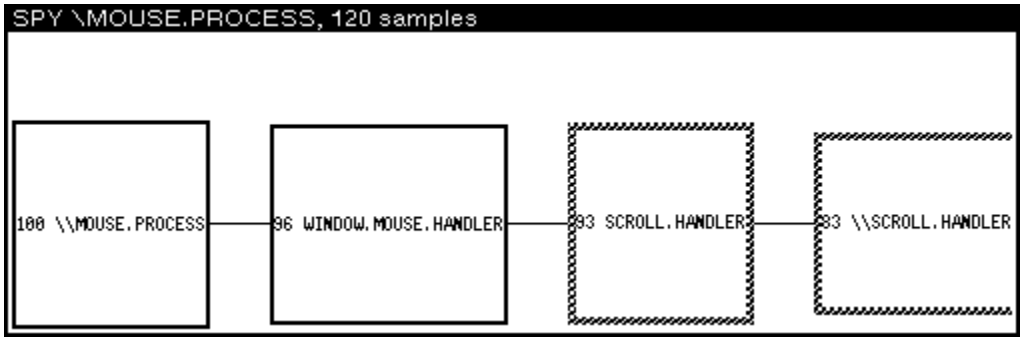
NewSubTree Creates another Spy window that includes data only from this node and its descendents (with the tree rooted at the selected node). Suppose that you were only interested in the actions that you had invoked with the mouse. You can left-click `\MOUSE.PROCESS` and select the NewSubTree option.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

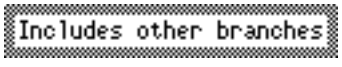
Legend	Displays SPY border interpretation.
Inspect	Allows for the inspection of the current parameter settings for the Spy window.
SetThreshold	Sets the threshold for displaying a node. Any node whose percentage is below the threshold won't show up (unless it is needed to connect the graph together). 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

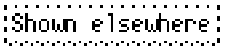
If you press the left, or middle button while the control key is down, you will get the same menu, but the action will be deferred until you next use the left/middle button. For example, you can delete several nodes and then do one update.

Merged Nodes

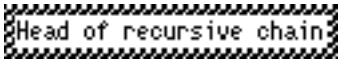
If two nodes are merged, then the merged node will include the time (and descendents) of the other. The display of a merged node is different; it is shown with a thick gray border; e.g. ,



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.

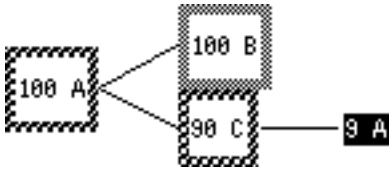


The case where a function is merged with a recursive call to itself is handled specially: the head of the recursion is marked with a wider checkered border:



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.

[This page intentionally left blank]