GRAPHCALLS

By: Christopher Lane (Lane@Sumex-Aim.Stanford.Edu)

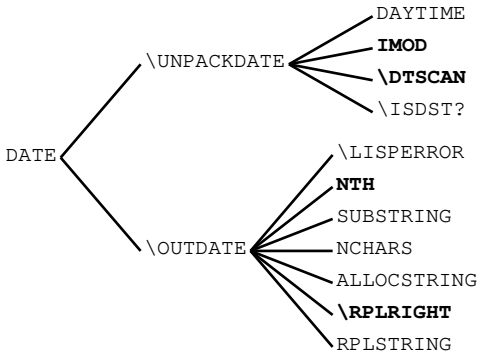
Uses: GRAPHER, MSANALYZE (WHERE-IS & HELPSYS optional)

GRAPHCALLS is an extended graphical interface to the Envos Lisp CALLS function. It is to CALLS what BROWSER is to SHOW PATHS in MASTERSCOPE. It allows fast graphing of the calling hierarchy of both interpreted and compiled code, whether or not the source is available (see the CALLS function in the MASTERSCOPE section of the *Lisp Library Modules* manual), allowing examination of both user and system functions. The sources of the functions do not have to be analyzed by MASTERSCOPE first.

Additionally, buttoning a function on the graph brings up a menu of operations that can be done with the function, such as editing, inspecting, further graphing etc.

(GRAPHCALLS *FUNCTION* &REST *OPTIONS*) [Function]

Graphs the calling hierarchy of *FUNCTION*. Terminal nodes on the graph (those which call no other functions or are undefined) are printed in a bold version of the graph's font indicating that they cannot be graphed further:



The remainder of the arguments, in keyword format, make up *OPTIONS* eg.

```
(GRAPHCALLS 'DATE :FONT '(GACHA 10) :DEPTH 4 :FILTER 'FGETD)
```

Options include:

:STREAM An image stream to display the graph on. The options list is saved on the stream.

:FILTER A predicate to apply to the functions when building the graph to test their eligibility to appear on the graph. The filter can be any defined function; the default is not to filter. Interesting filters include:

WHEREIS Limits the tree to only functions the user as has loaded and prunes out system functions and SYSLOADed files. Quite useful.

FGETD Limits the tree to only functions that are actually defined. Thus if you are perusing the tree for BITBLT and do not have and are not interested in the color code, FGETD will remove all of the undefined color bitmap functions.

EXPRP Limits the tree to interpreted functions. Useful for graphing functions in the development stage.

CCODEP Limits the tree to compiled functions.

NO Keeps low level functions starting with \ (i.e. \OUTDATE) off of the graph. Useful for getting an overview of system functions and when advising system functions (as \'ed functions should probably not be advised).

:DEPTH The calling hierarchy is graphed to *depth* levels (defaults to 2).

:FORMAT Passed to LAYOUTGRAPH and can be any format specification (LATTICE, VERTICAL, REVERSE etc.); defaults to (HORIZONTAL COMPACT REVERSE/DAUGHTERS). In the forest format multiple instances of a function appear on the graph after every calling function and a boxed node indicates the function appears elsewhere on the graph, possibly graphed further. In the lattice format each function gets placed on the graph only once (particularly useful for dynamic graphing, described below), and boxed nodes indicate recursive functions calls.

:SEARCHFN A function to use to generate the children of a given node. It should return a list whose first item is a list of the children, the other items in the list are ignore. Using this feature, it is possible to graph things other than functions. To graph what files load other files, supply a search function of `(LAMBDA (FILE) (LIST (FILECOMSLST FILE 'FILES)))` and a file name for the function argument.

:ADVISE Advises the functions after they are graphed (see *Dynamic Graphing* below); recognized values are one or both of the following:

INVERT Visually tracks a running program .

COUNT Counts function calls in a running program.

:DELAY The delay to use in advised graphs; defaults to 500 milliseconds.

:NAMEFN A function to use to generate the node labels on the graph.

:FONT The font to use to display the graph; defaults to (GACHA 8).

:SHAPE A boolean that indicates if the window should be shaped to fit the graph; defaults to NIL.

:PRIN2FLG A boolean that indicates to use PRIN2 when printing node labels, defaults to NIL.

:SUBFNDEFFLG A boolean that enables graphing of compiler generated functions; defaults to T.

:TOPJUSTIFYFLG Passed to SHOWGRAPH; defaults to NIL.

:ALLOWEDITFLG Passed to SHOWGRAPH; defaults to NIL.

GRAPH MENUS

The menu that pops up when you left button a function on the graph contains the following items:

?= Print the arguments to the function, if available.

HELP Calls HELPSYS on the function.

FNTYP Print the function's FNTYP.

WHERE Do a WHEREIS (with FILES = T) on the function.

EDIT Calls the editor on the function if available for editing.

TYPEIN BKSYSBUFs the name of the function into the typein buffer.

BREAK Applies BREAK to the function. Its subitems are:

- **BREAKIN** Breaks the function only in the context of a particular calling function. In lattice format, if the function has more than one function calling it on the graph, the user is prompted to indicate the caller in which to break the function.
- **UNBREAKIN** Undoes BREAKIN.
- **UNBREAK** Applies UNBREAK to the function.
- **TRACE** Applies TRACE to the function.
- **TRACEIN** Traces the function only when called from inside a particular function, like BREAKIN above. Use UNBREAKIN to remove the trace, or else UNBREAK on the window menu.

CCODE Calls INSPECTCODE on the function if it is compiled code.

GRAPH Calls GRAPHCALLS to make a new graph starting with function, inherits the original graph's options.

FRAME Inspect the local, free and global variables of the function. These are the last three lists of the CALLS function placed into INSPECT windows. Its subitems are:

- **>FRAME** Like FRAME but for all of the functions on the sub-tree starting at the selected node and only for FREEVARS and GLOBALVARS.
- **<FRAME** Like >FRAME but for all of the functions above the function in the graph, i.e. the FREEVARS and GLOBALVARS in the function's scope.

Buttoning the graph outside a node give you a menu with these options:

UNBREAK Does an (UNBREAK), unbreaking all broken functions.

RESET Resets the counters for the COUNT option and redisplays the graph.

DYNAMIC GRAPHING

When the ADVISE option is specified with the value(s) of INVERT and/or COUNT, GRAPHCALLS will advise all of the functions on the graph (in the context of their parent) to invert their corresponding node on the graph (as well as delay some to allow it to be seen) and/or follow each function name by a count of the number of times it has been executed. In invert mode, a node remains inverted as long as control is inside its corresponding function and it returns to normal when the function is exited. The

lattice format is best when using the invert feature. Closing the graph window UNADVISEs the functions on the graph.

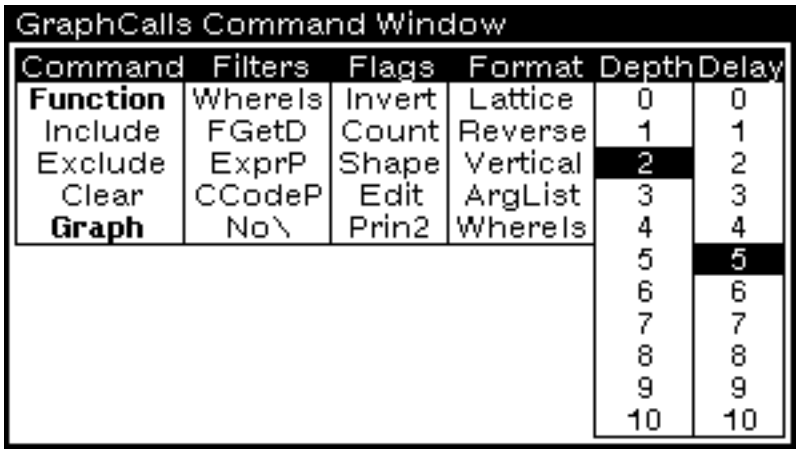
An example of this is `(GRAPHCALLS 'DATE :ADVISE 'INVERT)` and then evaluate `(DATE)`.

GRAPHCALLS will not graph or advise any function in the system list UNSAFE.TO.MODIFY.FNS when the advise option is used. Functions which are unsafe to advise should be added to this list.

CAVEAT PROGRAMMER! This feature must be used with caution. As a rule, one should not do this to system functions, but only one's own, use WHEREIS as a filter for this. Advising system code indiscriminately will probably crash the machine unrecoverablely.

You can, at some risk, interactively break and edit functions on the graph while the code is executing. Also, creating subgraphs of advised graphs will show the generated advice functions not the original functions called, as will creating new graphs of functions in advised graphs. You can create advised graphs of functions already graphed normally on the screen.

COMMAND WINDOW



(GRAPHCALLSW *[REBUILD?]*) [Function]

Puts up a command window with menus that will interactively set up calls to GRAPHCALLS. The menus let you set the Invert, Count and Edit flags, select from common filters and formats and set the depth of the graph. You can also change the amount of delay used in the advised functions when doing dynamic graphing. If you specify an advised graph (Invert or Count) and do not specify a WHEREIS filter, you will be asked to confirm with the mouse for your own protection.

More than one item on the filter and flags menus can be selected at a time. Buttoning a selected item on these menus unselects it. The command menu contains the following:

Function Prompts for the name of a function to graph when the **Graph** item is selected.

Include Adds files or functions to the list of items to allow on the graph, see the Include/Exclude algorithm below.

Exclude Adds files of functions to the list of items disallowed on the graph, see the Include/Exclude algorithm below.

Clear Clears all of the settings on the command window to their defaults. Also clears the Include/Exclude lists.

Graph Graphs the function by calling GRAPHCALLS with the selected options.

Include and Exclude allow fine tuning of the filter function. If the function passes the filter, then the following are tried until one determines whether or not the function will be on the graph:

If a set of functions has been explicitly excluded, and the function is a member of this set, it will NOT appear on the graph.

If a set of functions has been explicitly included, and the function is a member of this set, it WILL appear on the graph.

If a set of files has been explicitly excluded, and the function is in one of those files, it will NOT appear on the graph.

If a set of files has been explicitly included, and the function is not in one of those files, it will NOT appear on the graph.

The function WILL appear on the graph.

The format menu contains two items that are not passed on to GRAPHER but are used to select alternate NAMEFN options:

ArgList Supplies a NAMEFN that will print the function and its arguments (using SMARTARGLIST) as the node label.

WhereIs Supplies a NAMEFN that will print the function followed by the file(s) found by doing WHEREIS (with FILES = T) if any .

When the command window is open, middle buttoning a node on a GRAPHCALLS graph will bring up a menu of commands relating to command window and graphs. The menu contains:

EXCLUDE Adds the function to the exclude functions list of the command window. This is the only way to exclude system functions which get added to the SYSTEM file exclusion list.

The command window can also be obtained via the background menu. Subsequent calls to GRAPHCALLSW (either directly or via the background menu) will reuse the old command window if there is one. If this window is damaged, and redisplay does not help, then setting *REBUILD?* to T will build a new command window from scratch.

NOTES

- Function call graphs are constructed using breadth first search but GRAPHER lays out graphs depth first so functions may be expanded in different places on the graph than expected.
- GRAPHCALLS sysloads GRAPHER and MSANALYZE if they are not already loaded.
- In dynamic graphs, variables caused by advising show up in the frame inspections.
- The global variable GRAPHCALLS.DEFAULT.OPTIONS contains all of the defaults for GRAPHCALLS keywords, in property list format.