INFO: Interactive APL Documentation

George Mebus
Cognos/LEX2000 Inc.
2 Independence Way
Princeton, NJ 08540
George@LEX2000.com

ABSTRACT

A large body of APL code may be hard to understand and analyze, particularly if you are not its author. A code system that spans multiple workspaces (WSs) compounds that problem.

INFO is a Multi-WS system written in APL+Win that provides convenient interactive documentation of APL+Win and APL+DOS multi-WS systems. The User has a variety of displays that give insight into the system structures and relationships within single- or multi-WS systems. An administrator can easily set up and maintain the INFO static analysis data bases for any number of WS groups.

This paper demonstrates INFO by showing how INFO documents itself. A distribution package contains the complete code and instructions for setting up this self-documentation.

1. INTRODUCTION

For large systems of APL code, such as our LEX2000 Financial Reporting Software products, it is necessary to break the code into several WSs. Some reasons for using multiple WSs are:

- to reduce the amount of code that is resident in computer memory at one time;
- to separate code by major responsibilities, then use only the WS that you need for a particular job;
- to maintain single copies of objects (functions and variables) common to several WSs and then copying them as needed, rather than duplicating them in each WS;
- to keep collections of code that are seldom used from being always resident in memory.

One's ability to remember and understand the interrelationships among various functions decreases as the body of code increases, as less of it is one's own and as the original code gets older and gets changed. Some help is then needed to illustrate those relationships or to track how a variable changes among the function calling chains; one needs help in searching out information in the extended WS (the chosen WS with all of the objects from other WSs that are used in it).

INFO gives such help. INFO provides a User with scrollable alphabetically-ordered lists of the functions and/or variables or all names used in the extended WS. It displays function and variable definitions, name uses in functions (where and how used), function calling trees and "name usage trees" that show where a name is assigned, referenced and changed within function calling tree structures. INFO also supports flexible regular expression text searches. Each display is in a separate window; a name (or group of names) in the text of any window can be used to launch the display of other information about it (them).

An Administrator can easily define WS file groups that tell which WSs share sets of objects among themselves and how they are shared. Static analysis files are then generated for all with one button click. Thereafter, the INFO form shows when a WS has changed and allows quick incremental updates to the analysis data base, again, with the click of a button.

2. EVOLUTION

When I joined what has become LEX2000 Inc. ten years ago, I inherited a fairly large body of APL code (about 900 functions) in six WSs with little documentation and few comments. This code was for an application with which I was not familiar. Its developer had recently died. To be able to both develop and maintain this code required extra assistance. A debugger from Uniware was helpful in showing some detail workings, but gave no special insight into the design. The need to develop some "insight tools" was the genesis of INFO.

APL99, 07/99, Scranton, USA
©1999 ACM 1-58113-126-7/99/0 008 5.00

To determine the calling structure of APL code, each name used in each function must be found and the chain of function dependencies forged from them. Doing that analysis on demand in a WS wastes much time and space. Quick, comprehensive displays of such interrelationships can only be accomplished by a static preprocessing of the information and storing the results in a data base. This is INFO's approach.

Three static analysis files characterize the information about a WS:

1. definitions of functions and variables in the WS and related information about them;
2. the names used in each function, where and how;
3. the functions that use each name, where and how.

These files give quick access to the information required for INFO's variety of displays. They also require a lot of processing to generate.

2.1 Incremental Updates

As the LEX2000 code grew to double the original size and moved to new platforms (32-bit machines and then Windows), its complexity (and obscurity) also grew, and INFO had to grow with it to meet new challenges. For example, development code is always changing; so then must the contents of the INFO static analysis data base. This created the need to frequently re-analyze everything to keep up to date. That took too much time, so the data base was seldom current.

The data base and processing were redesigned to allow incremental updating so that only objects that had changed were re-analyzed and their information updated. This saved a lot of time in many cases, but not always.

The original multiple WS structure involved five separate WSs and one other (QPKFNS) for holding "packages" of shared and specialized objects that were copied by the others when needed. INFO would document a WS by copying all of these foreign objects from QPKFNS and then determining which in this extended WS had changed for the incremental updates. Unfortunately, the incremental updating still had some high overhead problems:

- copying objects from the large QPKFNS WS (and later from every other WS in LEX2000 for DOS);
- comparison of the `⎕vr` of every function to determine which had changed;
- re-creation of the function calling tree structure that is saved for each extended WS;
- if a QPKFNS common object (one used by all the other WSs) had been even trivially changed, then every WS would be re-evaluated and the changed object would be re-analyzed for every one -- a great waste of processing.

One special feature of the DOS version of INFO was the incremental WSDOCs. WS documentation was important in the early 1990's because so little could be seen in a computer screen's worth of text. From its data base knowledge of which functions had changed, INFO could generate incremental WSDOC change pages on demand, reducing the amount of paper used for WSDOCs while allowing frequent, timely updates.

2.2 Windows

When APL+Win became an effective development system, the advantages of Windows for *INFO* control and presentation became more obvious. The generation of separate resizable display windows offered more flexible control and comparison of various views. Better interactive displays were possible using color to identify text search results. A Windows control form could be much more convenient and capable than the set of command words used in the DOS version. So a conversion to the current Windows-based INFO began in 1997, still using command words at first but later being driven by the more convenient menus.

A new approach was also taken to incremental updating of the data bases. Update speed was increased and automated by:

- evaluating each WS separately (including the QPKFNS), then linking the results of dependent WSs at the time that a main WS is "opened" by the User;
- saving and comparing `⎕at` (timestamp) results for functions, then comparing the `⎕vr` of only those functions that have changed timestamps;
- using a modular function calling tree structure that can be incrementally updated;

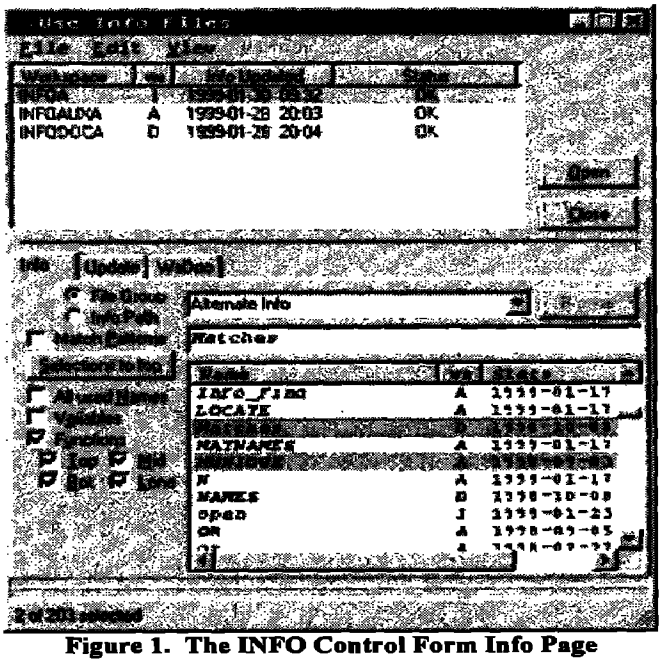

Figure 1. The INFO Control Form Info Page

- notification by display color that some WSs in a WS file group have been changed and that the data base files need updating.

These and other changes have made the maintenance of INFO data bases nearly automatic -- it is easy to keep them up to date. Because INFO makes on-demand documentation so easily available, the need for complete WSDOCs seems to have vanished. WSDOC generation is not currently implemented in INFO, but the original incremental code has been retained for possible inclusion in the future.

3. A USER'S TOUR

The best way to see what INFO does is to take a walk around the main form. When INFO is opened and a WS file selected, the User is presented with something like what you see in Figure 1 above.

3.1 Selections

The first step to using this form is to select the File Group (if one has been defined) or the Info Path to a folder containing INFO static analysis data base files. A Browse button can be used to find the desired folder. In Figure 1, a File Group comprised of an alternative version of the three WSs that make up the INFO documentation system have already been selected.

Once the file group has been chosen one must select the primary WS to be documented. INFOA in the WS list has been opened by double-clicking the primary ("left") mouse button on that line or selecting the INFOA row and either pressing the Enter key or clicking the Open button. Related information from the other WSs was then automatically linked to the one chosen and names of all the functions in the extended WS have been displayed alphabetically in the Names listview.

With this extended WS opened, we may notice several things:

- a "*ws*" letter and timestamp are associated with each name;
- the selected name appears in an edit field immediately above the Names list;
- a status line at the bottom shows how many names have been selected and how many there are in total;
- a "Functions" check box is checked, as are four others below it.

The "*ws*" letters show in which WS each of the functions resides. They match the "*ws*" identifier letters in the WSs listview at the top of the form.

The name in the edit field can be retyped by the User; as each letter is typed, the matching word is selected if there is an exact match, otherwise the alphabetically next word is selected. If a pattern match is required instead, the Match Pattern box can be checked and one or more patterns entered in the form used for the APL+ `]fns` user command. This typically results in several names being selected. The status bar message tells how many and the Selections to top button will move all selections to the top of the list for better observation.

The four check boxes below the Functions box classify the functions by whether they call others and are called by others:

- Top call others but are not called;
- Mid call others and are called;
- Bot (bottom) are called but call no others; and
- Lone are neither called nor call others.

Unchecking all but Lone will show all the orphans, then checking the Top will together show all that are not called by other functions. This kind of display may be helpful in showing functions that are not used and may be unnecessary.

If the Variables box is also checked, then the names of global variables defined in the WSs will be added in their proper alphabetic locations. Instead of timestamps, their *Stats* column values are indications of type, depth and shape. If only the Variables box is checked, only variable names are shown.

If the Names box is checked, then all the Functions and Variables boxes are disabled and all names used in any function in the extended WS will be displayed and can be selected as described above

Name lists can be reordered by clicking on the column headers of the list box. The first click will reorder the rows to alphabetize the column clicked. An immediate second click reverses the ordering. Thus if a group of names has been moved to the top of the list by Selections to top, the original order can be regained by clicking the "*Name*" column heading.

3.2 Real Information

Why do we need all these ways of selecting names? Because we can display a great variety of information about the selected names when the secondary (right) mouse button is clicked on the Names listview or when the F10 key is pressed. Either action presents a menu as shown in Figure 2. Most of the menu's selections open an information display window.

∇	function name
∇	locked function
!	recursive function
α	left argument
ω	right argument
ρ	result
;	local name
∘	referenced
←	assigned
]	indexed
⊢	index assigned or selective assigned
∈	strand assigned
:	label

Table 1. Name Identification Symbols

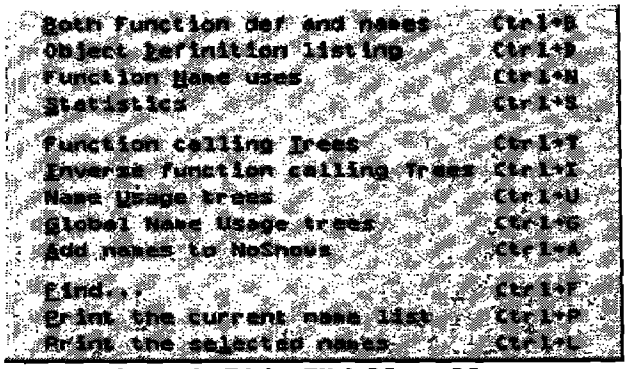

Figure 2. Right-Click Mouse Menu

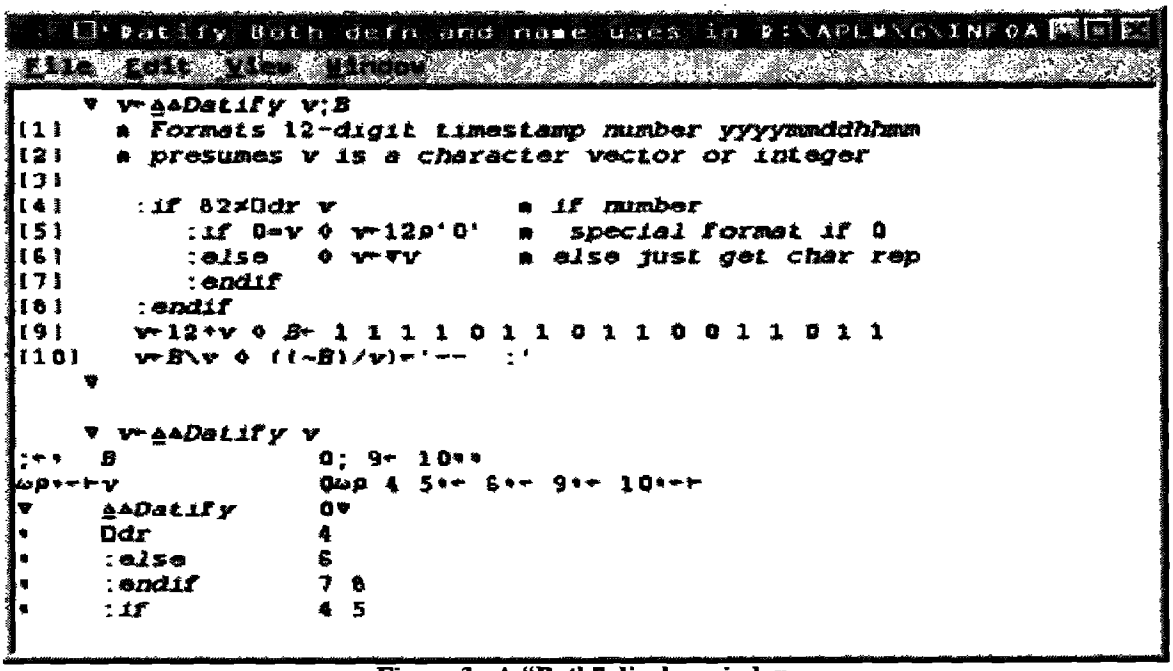

Figure 3. A "Both" display window

Now, let us look at the items in each section of the menu.

3.3 Definitions

The first section of this menu is concerned with "Object" (function or variable) definitions. The Both Function def and names combines the next two entries for functions in a single display. The Object Definition listing is a normal listing for a function or a value display for a variable. If the variable is nested, the standard nested "Display" form is used. The Function Name uses shows where and how names are used in a function.

Figure 3 shows a "Both" display window for a small INFO function named ⍙⍙*Datify*. The top part is a normal function listing and after that is the "function name uses" -- a compact display of each name used in the function, alphabetically ordered.

To the left of the names is a digest of how the name is used in the function and to the right are the particulars in a compact form. Each use is identified by a single symbol; multiple symbols after a line number show multiple uses for the name on that line. Note that the symbols associated with a line are in "chronological" order. For example, the name *v* is first referenced, then assigned in the same statement on lines 5, 6, 9 and 10. Table 1 shows the complete list of name identification symbols

A "Statistics" display window shows information about the selected name or names, telling in which WS they reside, their sizes and other specific information such as number of lines and timestamp for a function, type/shape/depth for a variable. Figure 4 shows statistics for a function and a variable.

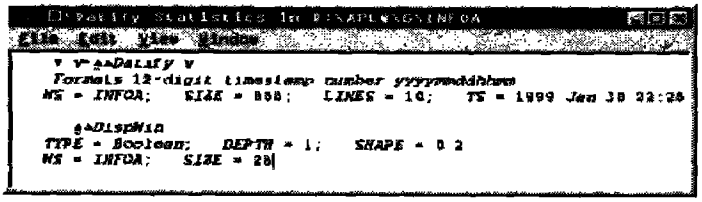

Figure 4. A Statistics Display Window

3.4 Trees

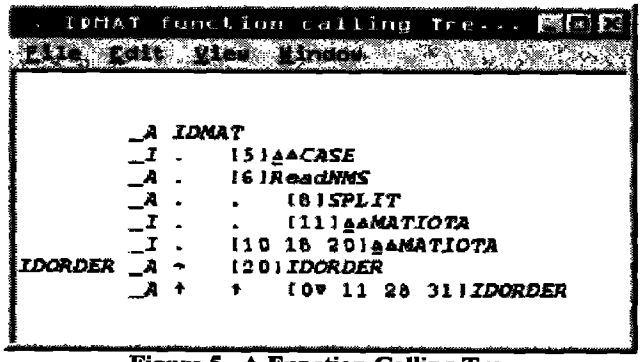

Figure 5. A Function Calling Tree

The second section of the menu is concerned with the display of function calling trees. These are currently implemented as text but may use Widows tree structures in the future. Figure 5 on the next page shows the Function calling Tree display for a function named *IDMAT*: It calls four functions. The line numbers of *IDMAT* on which they are called precede the function names. Those line numbers are colored blue.

The letters preceded by underbars are WS identifiers. The indented text shows the "who calls whom" relationships. Line numbers in brackets before functions names mean that they belong to the calling function, that they are the lines of the calling function on which the named function is called. Specifically Figure 5 shows that *IDMAT* calls ∆∆*CASE* on line 5, *ReadNMS* on line 6, ∆∆*MATIOTA* on lines 10 18 20, and *IDORDER* on line 20. Also, *ReadNMS* calls *SPLIT* on its line 8 and ∆∆*MATIOTA* on 11.

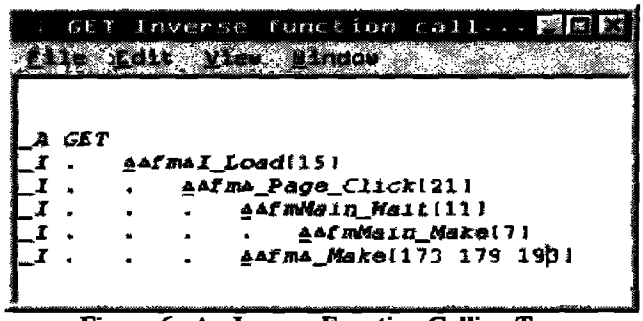

Figure 6. An Inverse Function Calling Tree

Note that *IDORDER* is recursive, calling itself on its lines 11 28 31. Where a function whose subtree expansion has already been shown appears again lower in the tree, its expansion is not shown again. Instead, up arrows are displayed in the column markers to show that the expansion appears above. The name of first-time expansions are also displayed on the far left to make the expansion easier to find.

Figure 6 shows an Inverse function calling Tree that illustrates the "who is called by whom" relationships. Although this looks very much like the forward calling tree, notice that the functions named in the tree have bracketed line numbers after the function names instead of before. It shows that the function *GET* is called by ∆∆*fmAI_Load* on line 15 of that function. The trail of calls continues to the top functions which are both Make functions, defining the Windows forms that start the whole process.

The name usage trees expand on the idea of function calling trees to show how a selected name (typically a non-global variable) is used within a calling tree structure. Figure 7 demonstrates the local Name Usage trees for a variable named *LocalNameFns*.

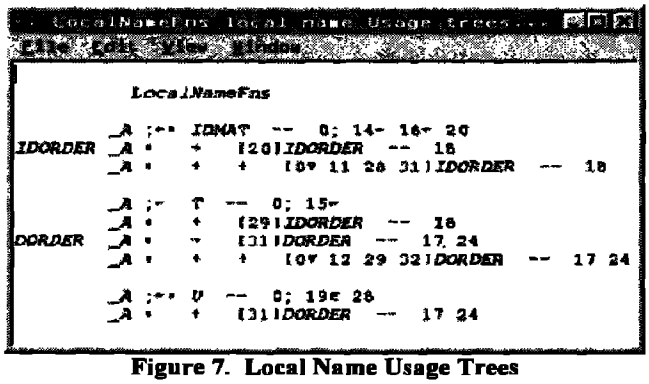

Figure 7. Local Name Usage Trees

Some additions to the normal function calling tree are:

- There are multiple trees;
- The selected name is listed above the trees;
- Name usage summaries appear between the WS identifiers and the function names. These summaries are especially useful in quickly spotting where a value is assigned and reassigned;

- Line numbers and specific uses follow the function names showing exactly where changes and references are made.

Global Name Usage trees differ from their local versions only in that they do not start where a name is first localized, but always begin at a "top" function. Figure 8 illustrates a global name usage tree for the same name as used before. They are generally much larger, as this is.

Finally, Add names to NoShows allows you to add the selected function names to a(n as yet hidden) list of names not to include in a Tree display. This feature can be used to avoid the clutter of low-level utility functions that don't really aid your understanding of the function being expanded. We will say more about that when we examine the Manage Tree Displays... dialog in Figure 12a..

3.5 Finding text strings

The third section of the right-click mouse menu has entries called Find..., Print the current names list and Print the selected names. The printing is just what you would expect it to be and won't be examined here. The Find capabilities, however deserve some special attention.

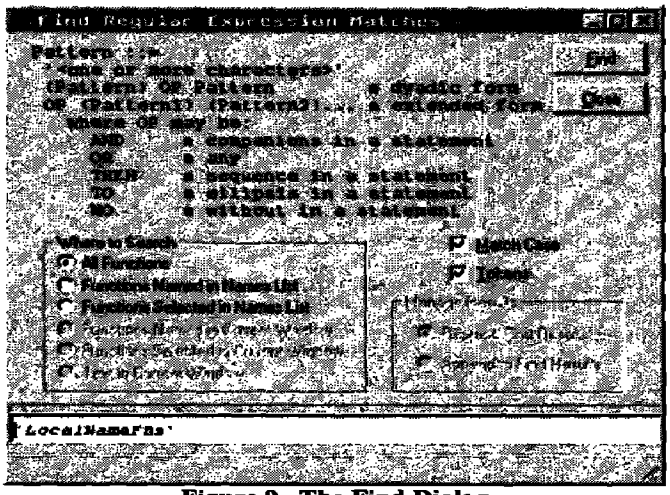

Figure 9. The Find Dialog

When you open the Find dialog, you may notice first that the name selected in the INFO Control form names list appears in the edit control near the bottom of the Find dialog and that it is enclosed in quotes. Check boxes for Match Case and Tokens tell how to search for the text string. The frame

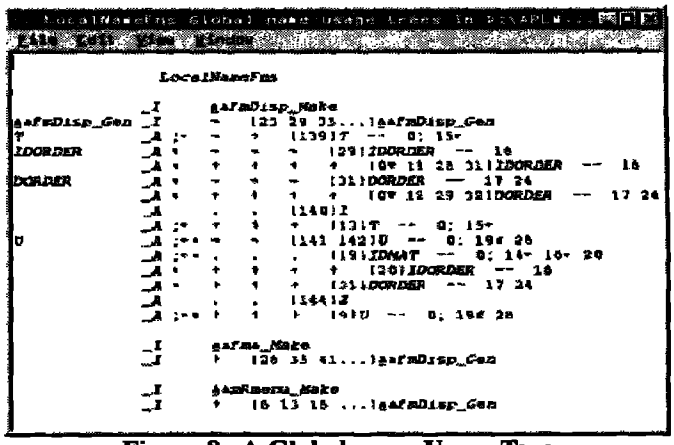

Figure 8. A Global name Usage Tree

Where to Search contains six options, enabled or disabled according to the current situation. They are:

- All Functions
- Functions Named in Names List
- Functions Selected in Names List
- Functions Named in Current Window
- Functions Selected in Current Window
- Text in Current Window

If we simply click the Find button or press Enter, the search begins. Because the All Functions option is chosen, all functions in the extended WS will be searched for this string and the results displayed as shown in Figure 10.

Although not shown in this publication except by lighter print, every occurrence of the text found is colored magenta. The first occurrence is selected and pressing the F3 key will move the selection to the next occurrence; Shift+F3 will move to the previous. It may be interesting to compare the results in this display window with those obtained from a Usage Tree as shown in Figure 7. One difference is that Find displays each entire line containing the searched text, but the functional context is not made obvious as it is in the Usage Tree. Another difference is that Find also displays matches in comments.

Besides displaying a name selected in the names list, one can type a string in the edit control. The up and down arrows will cycle through the set of strings already used, and there is always one empty string in the list so you can easily start over. These strings may be arbitrarily complex patterns through the use of five "operators" (or OPs): *AND*, *OR*, *THEN*, *TO* and *WO*. They may be used in upper case or lower case. Each may be used in a dyadic form (e.g., `'global' and 'FNS'` will match the two quoted strings only where they both appear in the same statement or comment). The arguments can be arbitrarily complex patterns. The other form each OP can take is a prefix form with any number of strings or other patterns as right arguments -- the processing is like an OP reduction of these arguments. If several names are selected in the names list when Find is selected or Ctrl+F is pressed, the pattern `OR 'name1' 'name2' 'name3'...` will be placed in the Find dialog edit control.

The meanings of the OPs are:

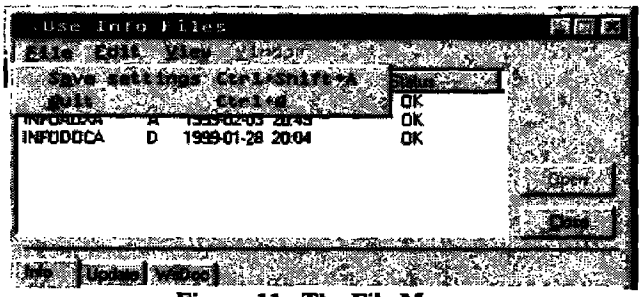

Figure 11. The File Menu

- *AND* -- all arguments are matched only where all the argument patterns occur in the same statement;
- *OR* -- any argument pattern is matched wherever it is found;
- *THEN* -- all arguments are matched only where the first pattern precedes the second (and the second precedes the third ...) in a statement;
- *TO* -- all text is matched from the first pattern to the second (and the second to the third ...) in a statement;
- *WO* -- (meaning "without") the first pattern is matched only if the second is not also found in a statement. If there are more than two argument patterns, the reduction model applies as if it was written: `'pattern1' wo 'pattern2' wo 'pattern3'...` and then evaluated from right to left.

Thus, patterns of some complexity may be searched and one match may vary quite a lot from the next. That is why color is used to mark the matches and why the Find Next and Find Previous (shown in Figure 15) highlight matches in sequence -- even if the matches overlap.

3.6 Dropdown Menus

Before leaving the INFO Control form, let us peruse the Dropdown menus at the top of the form. They offer some additional capabilities and repeat some others already seen. The four menus are : File, Edit, View and Window. As you have seen with other menus, these all have shortcut keys assigned so that nearly any operation can be done with or without the mouse. Even if some menus are clumsy ways to

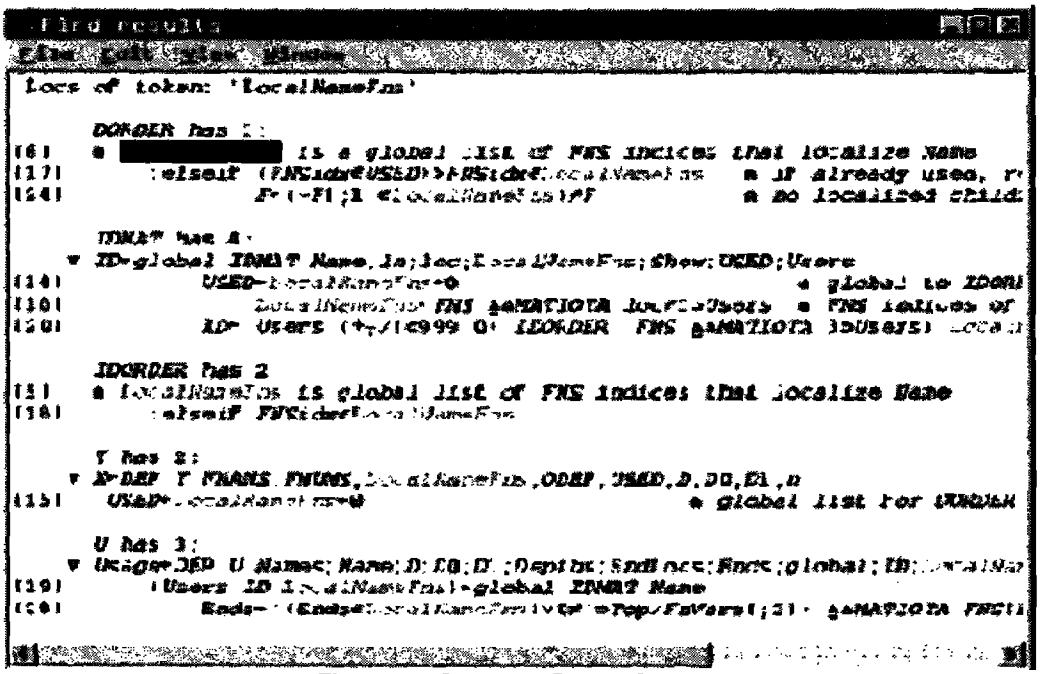

Figure 10. The Find Results Display

perform an action, they are good reminders of shortcuts that may be more efficient.

The File Menu in Figure 11 offers two capabilities. The second is Quit -- exiting from INFO, which can also be done by clicking the Close button or the Windows standard Alt+F4.

The first, Save settings, saves the positions, option values, check values and other settings of the INFO Control form and any other form or dialog that INFO displays. These settings are saved to a file named INFO.INI in the Windows folder.

The INFO Control form is fully resizable and there is a horizontal "splitter" separating the Files listview from the tabbed pages below it. The splitter can adjust the relative areas assigned to those two regions. As one uses more features of INFO (like the form size and splitter position) the Save settings will be used to record values and positions of these features.

The Edit Menu has three capabilities. The first is Find which we have already explored. The second Edit Menu

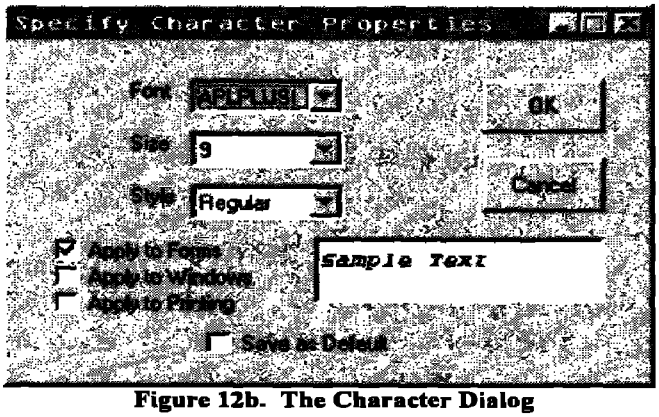

Figure 12b. The Character Dialog

entry is the Manage Tree Displays dialog shown in Figure 12a:

The Tree Properties frame of this dialog provides spinners to establish the number of character columns each tree level should be indented and how many levels should be displayed, and a check box for whether to use the NoShows list to hide a specific set of function names when generating tree displays. This shows the otherwise hidden NoShows list and gives tools to manage it. One may select function names from the current list of functions in the extended WS and can

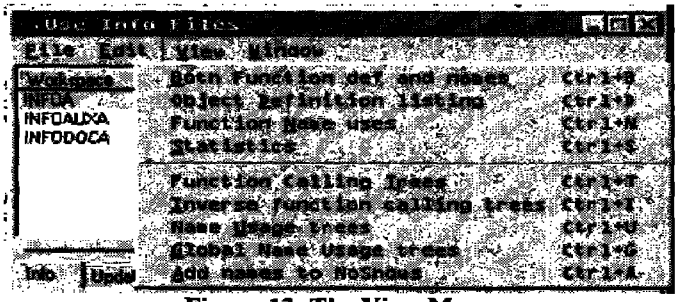

Figure 13. The View Menu

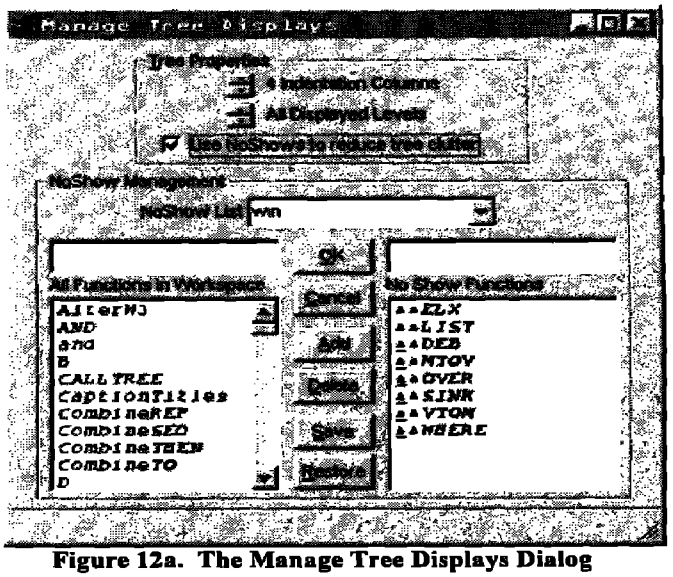

Figure 12a. The Manage Tree Displays Dialog

type other names in the edit control over the No Show Functions list. Names can be added to or deleted from the list; the changes made can be removed or can be permanently saved to the "Save settings" file, INFO.INI.

Note also that in the NoShow Management frame, a combo named NoShow list allows specification of and selection from a set of named NoShow lists. Each File Group that an Administrator may define identifies the NoShow list that it uses when the File Group is opened

When a User selects Add names to NoShows as described earlier, the names are added to the current list of NoShow names.

The third is a Character... dialog which allows adjustments to the character sets used for INFO forms, for the Display

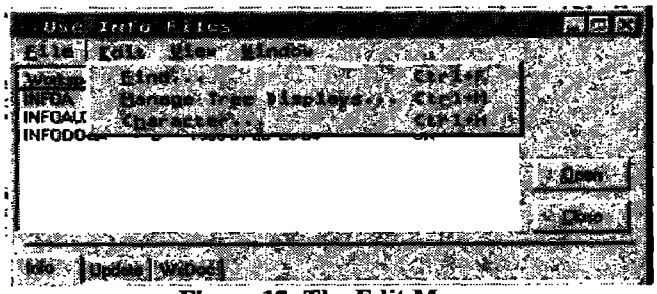

Figure 12. The Edit Menu

windows and for printing. The font, size and style can be specified for each as shown in Figure 12b: I like APLPLUSI 9 pt Regular and will continue to use it for the remainder of the paper.

The View Menu is a strict subset of the right-click mouse menu discussed thoroughly above.

The Window Menu controls the display windows generated by commands in the View Menu or the right-click mouse menu. The illustration in Figure 14 was captured when four display windows had been opened, and they are listed on the bottom section of the menu. F_Found is the result of a Find operation. The other three are various views of information

Figure 16. The Window File Menu

for the function *IDMAT* . B_IDMAT is a "Both" display, I_IDMAT is an inverse calling tree and S_IDMAT is a statistics display. The windows are listed in alphabetical order by object name then by single letter prefix. The top window is checked. You can get to any of these windows by clicking its menu entry; the top window can also be accessed by Ctrl+1.

The first section of the menu allows the User to make the set of display windows more orderly by Cascading them. They can all be closed, iconized or normalized (i.e., un-iconized).

3.7 Display Window Features

The Display windows are not only for display of information about APL objects and their relationships, useful as that is. Like the Names list on the INFO Control form, their displays include the names of functions, global variables and local names along with other information. Although the display windows are read-only, their text can be selected, and any name or set of names in a selection can be used to launch another display window. This flexibility provides much of INFO's power.

A right-click mouse menu much like that for the INFO Control Names list is available. If the name of a function is included in a selected area of a display window the menu looks like Figure 15.

The first two sections of the menu are identical to that for the Names list and work identically for names in the selected area. The third section has more features. Copy is available for copying the selected text to the clipboard. Two more Find-related items, Find next match (F3) and Find previous match (Shift+F3) also appear. These allow stepping from one matched pattern to the next (or previous). Clearly, the use of the F3 keys is more convenient than using these menu entries.

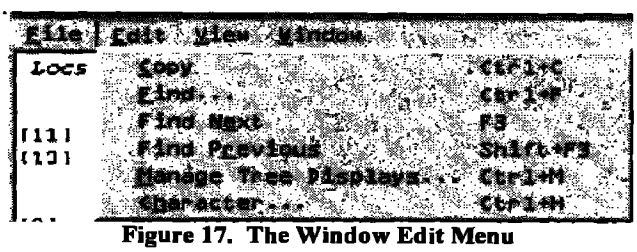

Figure 17. The Window Edit Menu

One additional note about the Find operation: it can find text patterns in the text of any display window -- even a Find Results window. These matches are displayed in red (in addition to the original magenta matches of the Find Results).

The two Print entries will print either the full text of the top window or its selected text. A feature of the printing is that it has some smarts. It tries to break a line first at a comment or statement separator, then at a delimiter. Also, if page breaks are needed, it inserts them at pairs of newlines unless

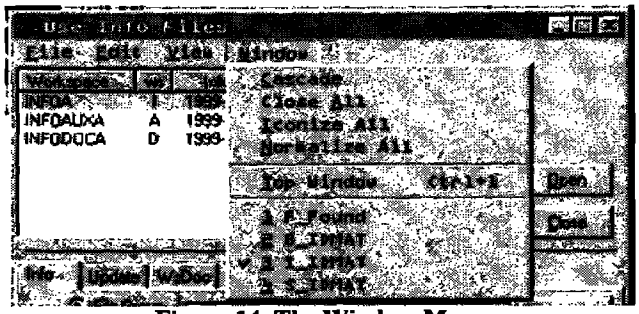

Figure 14. The Window Menu

this would increase the number of pages printed. A header with the window caption, a timestamp and "Page n of N" numbering is printed at the top of the page in Roman bold.

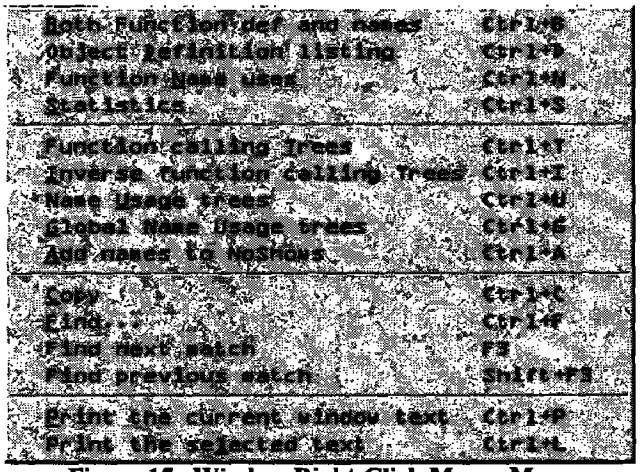

Figure 15. Window Right-Click Mouse Menu

3.8 Window Dropdown Menus

You may have noticed that the display windows have dropdown menus with the same names as those on the INFO Control form. There are some differences.

The File menu shown in Figure 16 duplicates the Print features in the right-click mouse menu, and allow quitting (closing) the window. Note that Esc will close the window. It is different from the Ctrl+Q needed for the INFO Control

form so that one may repeatedly press the Esc key to individually close windows without accidentally closing the INFO form by going too far.

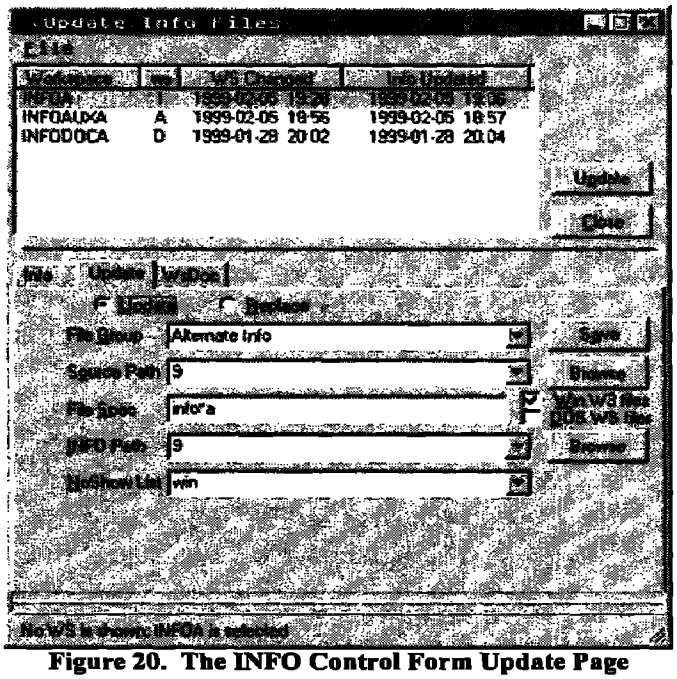

Figure 20. The INFO Control Form Update Page

The Edit menu shown in Figure 17 has the offerings of the INFO Control form with the addition of the extra items in the Window right-click mouse menu.

The View menu is identical to that on the INFO Control form, so it need not be re-displayed.

The Window menu in Figure 18 is completely different from its INFO Control form counterpart. All of the shortcut keys listed are more useful than the menu itself, but this is a good way to remind Users that they are there.

If several display windows are open, one can scroll through them in forward or reverse order. One can also scroll the text of a window up or down without the mouse (just as APL+Win allows). Finally, you can always get back to the INFO Control form by using Ctrl+1 -- the same shortcut used to get from the INFO form to the top display window; thus one may toggle back and forth between the two.

That concludes the User's Tour. I hope you got the impression that this is a useful, integrated set of capabilities that can quickly give new insight into the workings of a large system of APL code.

During this tour you surely noticed that the static analysis data base was already in place and current and you may have wondered just how that happened. Another tour that aims to answer this question is about to launch.

4. AN ADMINISTRATOR'S TOUR

If, when on the Info page (as we have been until now), we observe that one or more of the WS entries say "Needs Update" instead of the friendlier "OK" as shown in Figure 19, or notice that the background color of the WS name list has changed from white to pink, it is an indication that some coding changes have been made that are not yet reflected in the static data base.

Figure 18. The Window Window Menu

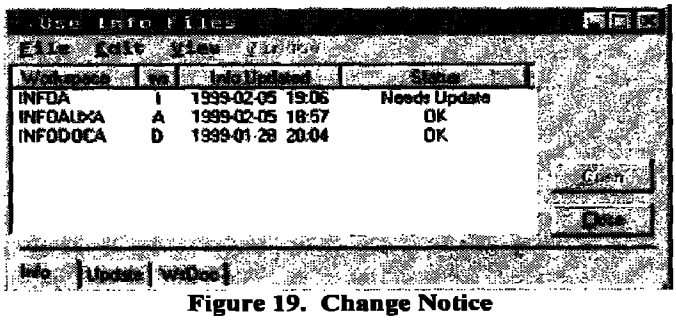

Figure 19. Change Notice

When the Update tab is clicked, the Update (or Administrator's) page is presented. From this page one maintains the data base of existing File Groups and adds new ones. Let us consider the maintenance first in Figure 20.
Note that the button that said "Open" when we were on the Info page now says "Update." Clicking that button causes an incremental update of the changed functions and variables to be made. Figure 20a shows such an update caught in the act. An Update Log window collects and displays the history of the update processing, capturing the status bar messages from the bottom of the INFO form and other information about the update. Once the update is completed, a return to the Info page allows selection on the Update Log of those functions and variables that were added and changed for viewing their definitions and uses, just as in any other display window.

That's all there is to updating the data base. You may notice in Figure 20 at the top of the page, just under the Update tab, are the options Update and Replace. As you have already guessed, if the Replace option was selected instead, the button would say Replace; clicking it would cause a complete replacement of the data base files for the selected

WS(s) -- a full analysis of all functions and variables in each WS, just as was done when the data base files were first created.

4.1 First Steps

So, how *were* they first created? Clearly, the Administrator had more to do than just click a button. Our tour will discover all the steps necessary for initially setting up the INFO static analysis data base for the INFO WS files themselves.

As has often been seen, the files to be analyzed are named INFOA, INFOAUXA and INFODOCA. They are slightly altered versions of the real group of INFO WS files named INFO, INFOAUX and INFODOC for reasons to be explained later. For now, let us look at the Update page controls that determine what gets processed. Although it is not essential to do so, we can consider the controls in geographical order.

First is the File Group. The text "Alternate Info" was typed in as the name that would identify the file group. As you see, the field is a combo control; it allows selection of existing file groups as well as specification of a new one.

Next, the Source Path was specified. In this case a library number was used, but a file path name is just as acceptable and the Browse button can be used to find the path where the INFO WSs reside. A history of source paths is maintained

The File Spec limits the names of files in the Source Path to be considered as members of the file group. In this case info*a indicates that only files with names beginning with "info" and ending with "a" are displayed in the WS list. The check boxes to the right further specify whether Windows and/or DOS WSs are included in the group.

The INFO Path determines where the static analysis data base files will be installed.

The NoShow List is a name of a set of NoShow names -- functions that will not be included in function calling tree expansions. Each file group can have its own NoShow list or one can be shared among several file groups. When a file group is selected on the Info page, the associated NoShow list is automatically loaded with the rest of the file group information.

At this point, the data base files can be generated. Assuming they do not already exist, the Update or Replace option selection is immaterial -- a complete analysis will be performed and a complete data base will be created. When done, the contents of each WS could be examined individually, but the interrelationships among the WSs have not been established. This is the tricky part.

4.2 Connections and the Donor Model

INFO's handling of multiple-WS connections is based on what I call the "Donor" model which is:

- a "donor" WS provides objects to other WSs either as its entire contents or in smaller groups called "compo-

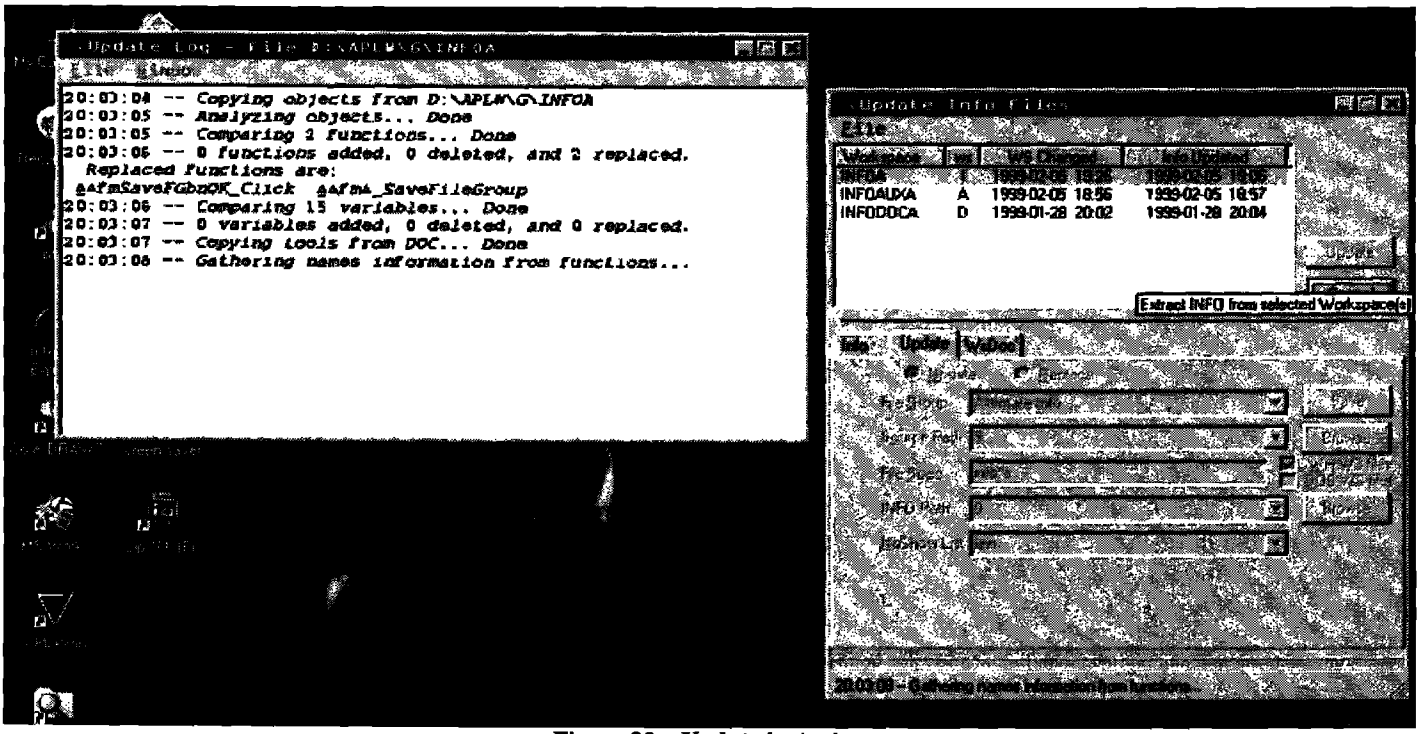

Figure 20a. Update in Action

nents;"

- if a donor WS provides components, it identifies them to INFO in a resident variable or niladic function named ⍙⍙*ComponentObjectNames* (its absence means the WS is only donated in its entirety);
- "Receiver" WSs in some manner receive components from the donor WSs, either by directly copying them or by reading a file component from a "package" file. Which components are received from donors are identified by component numbers.

Expanding, the value of ⍙⍙*ComponentObjectNames* can be almost any organization of a set of groups of names:

- a vector of nested vectors of names;
- a vector of matrices of names;
- a vector of vectors of space-separated names;
- a matrix with space-separated names on rows;
- a newline-separated character vector with space-separated names between the newlines.

Each group in the set is identified by the index number of its position in the list -- its row or element location. This position number is known as its "Component". With ⍙⍙*ComponentObjectNames* defined in each WS that acts as a donor (which could be any or all WSs in the file group) and the knowledge of which component numbers

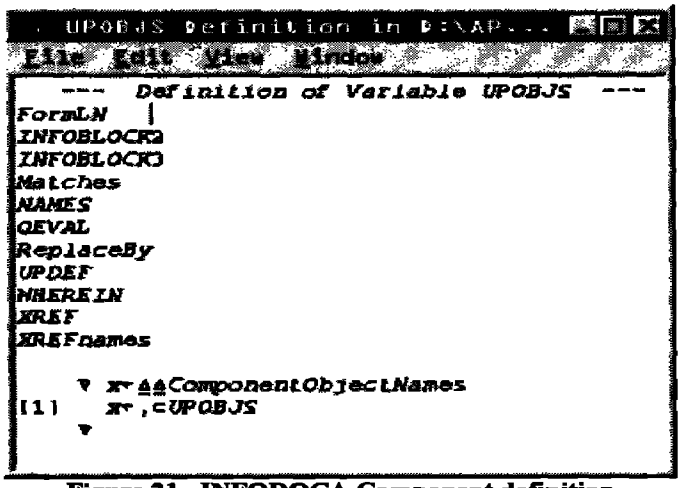

Figure 21. INFODOCA Component definition

from each donor are received by the receiver, the Administrator can define the WS connections.

4.3 The Organization of INFO

With that in mind, let us look at the three WSs that comprise this multi-WS system called INFO.

- INFO.W3 is the start-up WS that contains the always-resident functions. It copies in all of INFOAUX.W3 and a subset of the INFODOC.W3 objects when they are needed, then discards them when they are not. INFO creates and handles all the Windows GUI interactions -- forms, display windows and dialogs.
- INFOAUX specializes in generating the information shown in the Info page display windows. This includes Find processing, Trees and Statistics displays.
- INFODOC handles the analysis of functions and variables in the Update page processing. It also contains the old WSDOC generation code although that is not currently used in INFO.

Self-documentation of its own WSs presents INFO with a dilemma. INFO gets the function and variable definitions by copying the WS to be documented into the INFO WS. Since namespaces are not (yet) implemented in APL+, the resident INFO functions and variables have been given "funny" names beginning with the two characters ⍙⍙ to avoid name conflicts with the objects copied in. Definitions of only the non-funny objects are then extracted and placed in the first data base file. Then those copied objects can be erased and the INFODOC functions (which do not have funny names) can be brought in to process their definitions.

But when the WS to be documented is INFO itself, the funny names don't prevent name conflicts at all. There is a perfect match of every one, so no non-funny names are found and nothing gets documented.

The solution adopted here is to generate an Alternate INFO WS set (the INFOA set) that has all occurrences of ⍙⍙ names replaced by names beginning with ⍙⍙ and then document the resulting WSs. In this way the alternate set has no conflict with the original and it can be completely documented. A function named *AlterW3* is used to perform the transformation on each of the three WSs and generate or update the alternate WSs. In fact, *AlterW3* can be used to make a completely functional new INFO processor with any desired "funny" name prefix if one finds that desirable.

To recap, we document an alternate version of the INFO WSs: INFOA, INFOAUXA and INFODOCA. INFOA uses all of its own objects (the default connection of a WS to itself), connects to all of INFOAUXA and to one component of INFODOCA identified as component number 1.

The description of the single component definition in INFODOCA is shown in Figure 21.

First, the variable UPOBJS is displayed. It is an 11-row character matrix with one name per row. INFODA copies in this set of names, then copies those named objects in preparation for function and variable definition processing. The definition of ⍙⍙*ComponentObjectNames* is a function that just returns the ravel of an enclosed UPOBJS -- a one-element nested vector of character matrices. That satisfies the requirements of INFO WS connection evaluation.

4.4 Getting Connected

Thus, in our illustration, the three WSs with a file specification of **info*a** have been created, located, and had their data base files created. We only need to save the File Group information, including WS connections, to enjoy complete multi-WS documentation. When the **Save** button on the Update page is clicked, a dialog is presented for saving [changes to] the definitions of the File Group and editing the WS Connections.

If a new File Group is being defined, the Create File Group Definition dialog in Figure 22 appears, telling the new name, allowing the definition of WS Connections and offering to copy the contents of an existing NoShow names list into a new NoShow list. If OK is clicked here the new file group with all the definitions shown here and on the **Update** page will be saved to the INFO.INI file. If the **Edit WS Connections** box is checked, the Connections editor will appear. We will see this soon.

If an existing File Group definition is being revised, the Change File Group Definition dialog (Figure 22a) asks whether to replace the existing definition with any changes in the **Update** page specifications and whether to Edit the WS Connections. In our case we will uncheck the **Replace existing File Group** box and will check the Edit box. Clicking OK opens the Workspace Connections dialog (Figure 23).
This Connections editor is basically a listview control. Use of a third-party grid may be more elegant but INFO wants to use only the facilities provided by APL+Win.

Note that the three WS names in the system appear both down the left side and across the top of the listview. Initially, the word **ALL** appears down the diagonal of the array, all other entries are blank and the WS id column (next to the **Workspace** column) is blank. To specify the connections and the WS id letters the Administrator does the following:

- Clicks a Workspace row of the listview (I chose INFOA). That WS name is then displayed at the first line of the dialog -- in this case **Workspace INFOA;**
- Enters or changes the WS id letter in the edit control labeled **has WS letter**, and presses Enter;
- Clicks the column header for a donor WS in preparation for identifying which components of this WS the receiver WS will use (I chose INFODOCA). Note that the WS name appears in the line that says **and uses these components of Workspace INFODOCA;**
- Enters the connection component numbers in the long edit control just above the listview and presses Enter. In this case the **1** indicates that INFOA uses component 1 of INFODOCA;
- In the same manner declares that INFOA connects to ALL of INFOAUX (by typing an A and pressing Enter) and gives appropriate WS identifier letters to the other WSs; then
- Clicks the OK button to save all of this information to a file named INFOGRP.SF in the static data base.

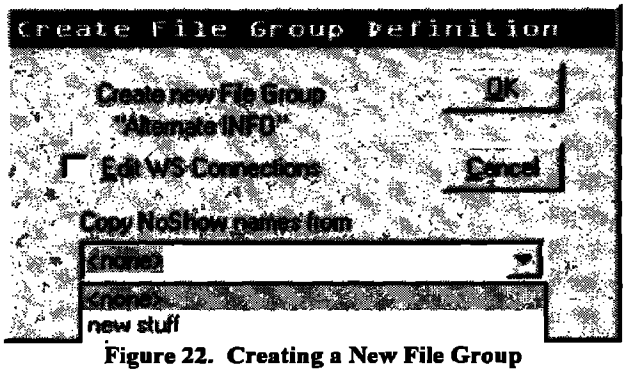

Figure 22. Creating a New File Group

With that, the setup of the File Group has been completed. Thereafter, any changes made to INFO code is reflected in the INFOA code by executing `AlterWS` on the modified WS(s). Upon opening INFO and selecting the Alternate Info file group, the fact that changes have been made are announced in black and pink. Moving to the Update page and

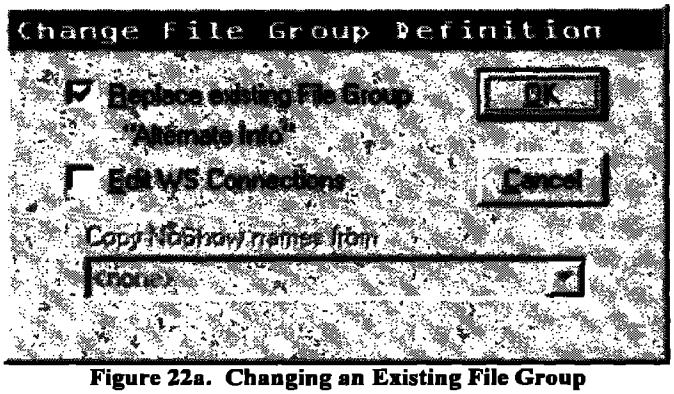

Figure 22a. Changing an Existing File Group

clicking the **Update** button makes the data base reflect the new changes.

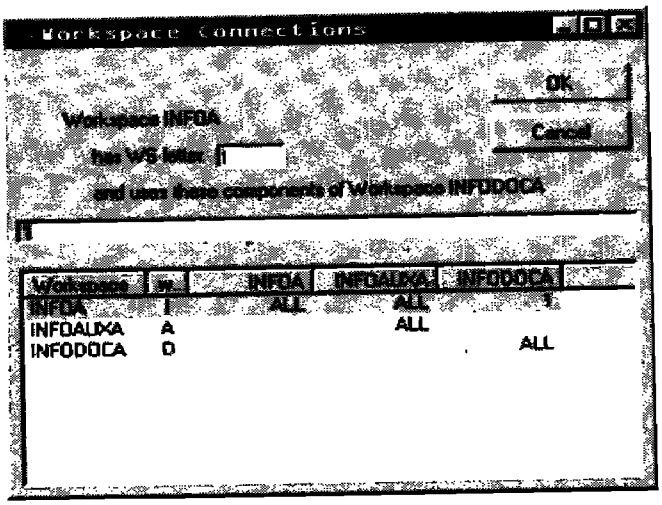

Figure 23. Workspace Connections Editing

5. PLANNED ENHANCEMENTS

There is always room for improvement. A number of changes are required, anticipated or hoped for the future. Among them are:

- true Windows tree controls instead of indented text
- improved handling of objects in different WSs with identical names.

ACKNOWLEDGEMENT

I wish to thank David E. Siegel for his consistent support and encouragement in making this paper possible and for insightful suggestions that have made INFO a much better tool.