22. RESOURCE MANAGEMENT

Naming Variables and Records

You will find times when one environment simultaneously hosts a number of different programs. Running a demo of several programs, or reloading the entire Medley environment from floppies when it contains several different programs, are two examples that could, if you aren't careful, provide a few problems. Here are a few tips on how to prevent problems:

- If you change the value of a system variable, MENUHELDWAIT for example, or connect to a directory other than {DSK}<LISPFILES>, write a function to reset the variable or directory to its original value. Run this function when you are finished working. This is especially important if you change any of the system menus.

- Do not redefine Medley functions or CLISP words. Remember, if you reset an atom's value or function definition at the top level (in the Executive Window), the message (*Some.Crucial.Function.Or.Variable* redefined), appears. If this is not what you wanted, type **UNDO** immediately!

  If, however, you reset the value or function definition of an atom inside your program, a warning message will not be printed.

- Make the atom names in your programs as unique as possible. To do this without filling your program with unreadable names that noone, including you, can remember, prefix your variable names with the initials of your program. Even then, check to see that they are not already being used with the function BOUNDP. For example, type:

```
(BOUNDP 'BackgroundMenu)
```

  This atom is bound to the menu that appears when you press the leff mouse button when the mouse cursor is not in any window. BOUNDP returns T. BOUNDP returns NIL if its argument does not currently have a value.

- Make your function names as unique as possible. Once again, prefixing function names with the initials of your program can be helpful in making them unique, but even so, check to see that they are not already being used. GETD is the Interlisp-D function that returns the function definition of an atom, if it has one. If an atom has no function definition, GETD returns NIL. For example, type:

```
(GETD 'CAR)
```

  A non-NIL value is returned. The atom CAR already has a function definition.

- Use complete record field names in record FETCHes and REPLACEs when your code is not compiled. A complete record field name is a list consisting of the record declaration name and the field name. Consider the following example:

```
(RECORD NAME (FIRST LAST))
(SETQ MyName (create Name FIRST←'John LAST←'Smith))
(FETCH (NAME FIRST) OF MyName)
```

- Avoid reusing names that are field names of Lisp system records. A few examples of system records follow. Do not reuse these names.

```
(RECORD REGION (LEFT BOTTOM WIDTH HEIGHT))
```

```
(RECORD POSITION (XCOORD YCOORD))
(RECORD IMAGEOBJ (- BITMAP -)))
```

- When you select a record name and field names for a new record, check to see whether those names have already been used.

  Call the function RECLOOK, with your record name as an argument, in the Executive Window (see Figure 22-1). If your record name is already a record, the record definition will be returned; otherwise the function will return NIL.

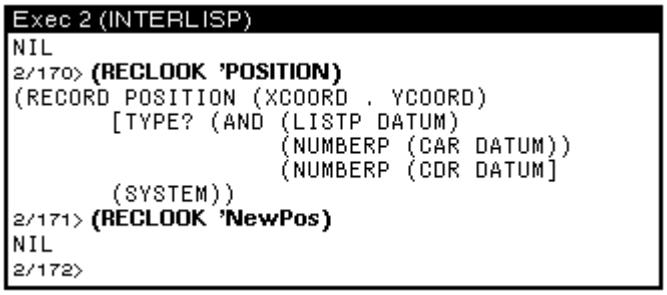

Figure 22-1. Response to RECLOCK

Call the function FIELDLOOK with your new field name in the Executive Window (see Figure 22-2). If your field name is already a field name in another record, the record definition will be returned; otherwise the function will return NIL.

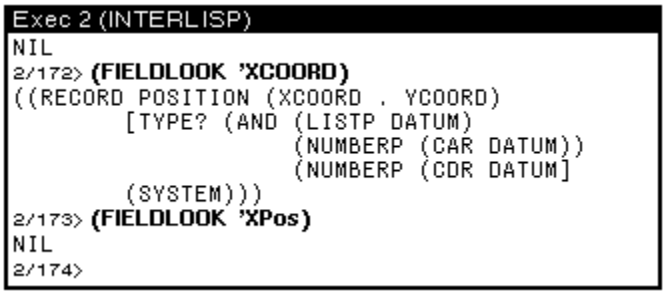

Figure 22-2. Response to FIELDLOOK

Some Space and Time Considerations

In order for your program to run at maximum speed, you must efficiently use the space available on the system. The following section points out areas that you may not know are wasting valuable space, and tips on how to prevent this waste.

Often programs are written so that new data structures are created each time the program is run. This is wasteful. Write your programs so that they only create new variables and other data structures conditionally. If a structure has already been created, use it instead of creating a new one.

Some time and space can be saved by changing your RECORD and TYPERECORD declarations to DATATYPE. DATATYPE is used the same way as the functions RECORD and TYPERECORD. In addition, the same FETCH and REPLACE commands can be used with the data structure DATATYPE creates. The difference is that the data structure DATATYPE creates cannot be treated as a list the way RECORDs and TYPERECORDs can.

Global Variables

Once defined, global variables remain until Lisp is reloaded. Avoid using global variables if at all possible! One specific problem arises when programs use the function GENSYM. In program development, many atoms are created that may no longer be useful. Hints:

- Use

  (DELDEF *atomname* 'PROP)

  to delete property lists, and

  (DELDEF *atomname* 'VARS)

  to have the atom act like it is not defined.

  These not only remove the definition from memory, but also change the appropriate fileCOMS that the deleted object was associated with so that the file package will not attempt to save the object (function, variable, record definition, and so forth) the next time the file is made. Just doing something like

  (SETQ (*arg atomname*) 'NOBIND)

  looks like it will have the same effect as the second DELDEF above, but the SETQ does not update the file package.

- If you are generating atom names with GENSYM, try to keep a list of the atom names that are no longer needed. Reuse these atom names, before generating new ones. There is a (fairly large) maximum to the number of atoms you can have, but things slow down considerably when you create lots of atoms.

- When possible, use a data structure such as a list or an array, instead of many individual atoms. Such a structure has only one pointer to it. Once this pointer is removed, the whole structure will be garbage-collected and space will be reclaimed.

Circular Lists

If your program is creating circular lists, a lot of space may be wasted. (Many crosslinked data structures end up having circularities.) Hints when using circular lists:

- Write a function to remove pointers that make lists circular when you are through with the circular list.

- If you are working with circular lists of windows, bind your main window to a unique global variable. Write window creation conditionally so that if the binding of that variable is already a window, use it, and only create a new window if that variable is unbound or NIL.

Here is an example that illustrates the problem. When several auxilIary windows are built, pointers to these windows are usually kept on the main window's property list. Each auxilIary window also typically keeps a pointer to the main window on its property list If the top level function creates windows rather than reusing existing ones, there will be many lists of useless windows cluttering the work space. Or, if such a main window is closed and will not be used again, you will have to break the links by deleting the relevant properties from both the main window and all of the auxiliary windows first. This is usually done by putting a special CLOSEFN on the main window and all of its auxiliary windows.

When You Run Out of Space

Typically, if you generato a lot of structure! that won't get garbage collected, you will eventually run out of space. The important part ii being aNe to track down those structures and the code that generates them to become more space efficient.

Use the Lisp Library Package GCHAX.DCOM to track down pointers to data structures. The basic idea is that GCHAX will return the number of references to a particular data structure.

A special function exists that allows you to get a little extra space so that you can try to save your work when you get toward the edge (usually noted by a message indicating that you should save your work and load a new Medley environment). The GAINSPACE function allows you to delete non-essential data structures. To use it, type:

```
(GAINSPACE)
```

into the Executive Window. Answer N to all questions except the followi ng.

- Delete edit history
- Delete history list.
- Delete values of old variables.
- Delete your MASTERSCOPE database
- Delete information for undoing your greeting.

Save your work and reload Lisp as soon as possible.