Just Ask!

Question: Does the LOCK command change the naked indicator?

Editors: The short answer is: sometimes. The long answer is a bit more helpful.

First, it may be helpful to understand what the LOCK command is doing. The LOCK command is really an inter-process communication mechanism. The purpose of the LOCK command is to inform other processes that your process intends to do something. Exactly *what* you are going to do must be decided upon ahead of time for the command to be useful.

The simplest example is a system wherein anytime a process is going to change a global node it must first use the LOCK command with an argument that matches the name of the node. For example, if you are going to change ^TEST(9), you would first issue the command:

```
LOCK +^TEST(9)
```

You would then make the change. After the change was completed you would unlock with the command:

```
LOCK -^TEST(9)
```

Most programmers commonly describe the process by saying that they are going to "lock the node, change it, then unlock it." While this appears to be what has happened, it is misleading.

The standard defines the argument of a LOCK command as an nref (name reference). An nref looks like a variable name. That is, it has a name with an optional prefix of a circumflex (^) and an optional list of expressions separated by commas and contained within parentheses. The list of expressions ends up looking just like a list of subscripts. A valid nref looks no different than a valid glvn (global or local variable name) except that an nref must always have a name (a naked reference is not allowed). M knows the difference between an nref and a glvn by the context (LOCK command arguments are always nrefs).

An easy way to think of the LOCK command is as a bulletin board where we place Post-it Notes. (In this example we will assume that the LOCK command is used to regulate access to globals and that the argument(s) of the LOCK is the same as the global nodes to be accessed.) Before going to the file room (database) and taking a file (global node) out, each clerk (process) puts a note up on the board to let the other clerks know that he wants to work on those files. Before posting the note, the clerk looks to see if any other clerk already has the file checked out. If not, the clerk posts its note then goes to the file room to get the file.

There is no connection between the bulletin board and the file room. The rules for usage are those agreed upon by the clerks. Perhaps the clerks agree to use yellow notes to indicate they are making changes to the file and green notes when they only want to read them. Perhaps the files are organized by name, but the clerks agree to place the file identification number on the notes.

Whatever system they use, the system works because everyone agrees to the system *and abides by it*. There is nothing that prevents a lazy clerk from ignoring the bulletin board (saving a few steps for himself, making a potential mess for everyone else) and going directly to the file room and taking out a file.

Back to the naked indicator. The only thing that can change the naked indicator is a gvn. This is what we expect: Any time we reference a global in a SET or KILL command the naked indicator is affected. Since the argument of the LOCK command is not a gvn (global variable name), the naked indicator is not affected.

The consequence of this is that the argument of the LOCK command need not have anything to do with the global(s) to be modified. However, we usually do match them to make the code more readable and the process more understandable.

All that is to explain why the LOCK command does not directly change the naked indicator. However, it is only the nref itself that does not affect the naked indicator. How we determine what the nref actually is *may* change the naked indicator.

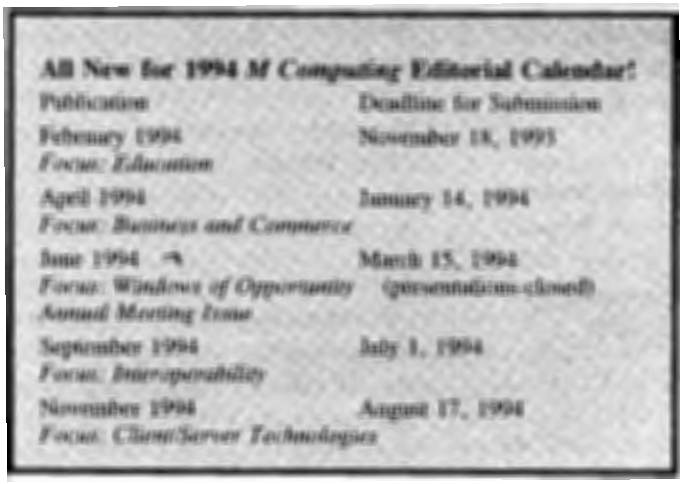
Example:

```
KILL ^TEST
SET ^TEST(1)="^TABLE(5)"
SET ^TEST(2)=9
SET ^TEST(9)="Middle"
SET ^TEST(1,9)="Bottom"
LOCK ^TABLE(5)
WRITE ^(9)
```

will display "Bottom".

An Invitation from *M Computing* Editors

We welcome articles and news items submitted by our readers. If you have an idea about M application areas, new systems and installations, interfacing, programming techniques, challenges of technology, or something related to our M community, let us hear from you. Book reviews, meeting notes, product announcements, and industry news items are also of interest. All submissions are reviewed and are subject to editing. Final determination about copy submitted for publication in the magazine rests with the editors. Send a brief description of your proposed article to Marsha Ogden, Managing Editor, MTA, Suite 205, 1738 Elton Road, Silver Spring, Maryland 20903-1725. Phone 301-431-4070, fax 301-431-0017, or use FORUM.



All New for 1994 *M Computing* Editorial Calendar!

Publication	Deadline for Submission
February 1994 *Focus: Education*	November 15, 1993
April 1994 *Focus: Business and Commerce*	January 14, 1994
June 1994 *Focus: Windows of Opportunity* *Annual Meeting Issue*	March 15, 1994 (presentations closed)
September 1994 *Focus: Interoperability*	July 1, 1994
November 1994 *Focus: Client/Server Technologies*	August 17, 1994

However:

```
KILL ^TEST
SET ^TEST(1)="^TABLE(5)"
SET ^TEST(2)=9
SET ^TEST(9)="Middle"
SET^TEST(1,9)="Bottom"
LOCK ^TABLE(^TEST(2))
WRITE ^(9)
```

will display "Middle".

Likewise:

```
KILL ^TEST
SET ^TEST(1)="^TABLE(5)"
SET ^TEST(2)=9
SET ^TEST(9)="Middle"
SET^TEST(1,9)="Bottom"
LOCK @^TEST(1)
WRITE ^(9)
```

will display "Middle".

Because a gvn reference was used to determine the nref, the naked indicator was affected.

Finally, there is one twist on the LOCK command. Notice that the circumflex is optional and that the nref may look like either a global or a local variable name. Usually we think of locking globals, since we generally use that LOCK command as if it were really locking something rather than just posting a notice. However, locking a *local* variable name is also possible.

When the standard was first developed in the early seventies it assumed that a routine or global name was unique within the system. That is, multiple environments (*namespace*, *UCI*, *UIC*, etc.) did not exist, at least in the standard. With the advent of different environments, we needed to handle locking of a global name that might not be unique.

In general, the implementors handled this well, assuming that the global to be locked was the one in your current default environment, unless you explicitly referenced a different environment. But what happens when you lock a local?

With the standard mute on the subject, the implementors were on their own. This led to different behaviors on different systems, none of which are *wrong*, because the standard never addressed it.

Thus, LOCK DUZ on one system may lock DUZ only for processes running in the same environment. Yet, on a different system, the same LOCK may take effect across all environments on the system. While standard, it is not portable. M

Send *Just Ask!* questions or requests to the managing editor at *M Computing*.