About Thieves, Valuables and Globals

by Winfried Gerum

Programmers do to their globals what police recommend to do with thieves and valuables: Lock them up! At least that is the perception of programmers, of what they do when they use the M LOCK command.

Following the discussions on the GUM project, I again have seen a lively discussion between old hands and people who actually want to lock globals the same way the police lock up thieves.

LOCK does not work this way. In fact, it has nothing at all to do with globals. With LOCK you can claim "rights" to a name, or, to put it more precisely, to a namespace. If your request is granted, nobody else claiming this namespace (or a superset of it) can succeed as long as you hold to your claim.

The syntax appears in Figure 1.

An argumentless LOCK frees all namespaces claimed by your job.

In the form LOCK glvn, you first surrender all previously claimed namespaces. Then your job is suspended until your claim to the namespace is granted. The optional timeout gives you a chance to react, if your request cannot be satisfied within a specified timelimit. If your request is successful within the timelimit, $TEST is set to 1. Otherwise, the LOCK command sets $TEST to 1 and terminates.

Why does LOCK free my previously claimed namespaces before it gets additional namespaces? The reason is that there is the danger of a deadlock. If you own "^A" and want to add to your list and your neighbour owns "^B" and wants to claim "^A," then both of you will wait until the end of time. So the simple syntax of LOCKs makes it a bit more complicated if you want to shoot yourself in the foot: You can add ^B to your locks with LOCK +^B and you can selectively free your lock on ^B without giving up other locks by LOCKs by LOCK -^B. If you own ^A and you want to add ^B to your list of namespaces then there are only two ways to do so without the danger of a deadlock:

```
L (^A,^B)
```

or

```
LOCK +^B:timeout Else  Goto Sorry
```

In the former case you give up the rights on ^A and then you

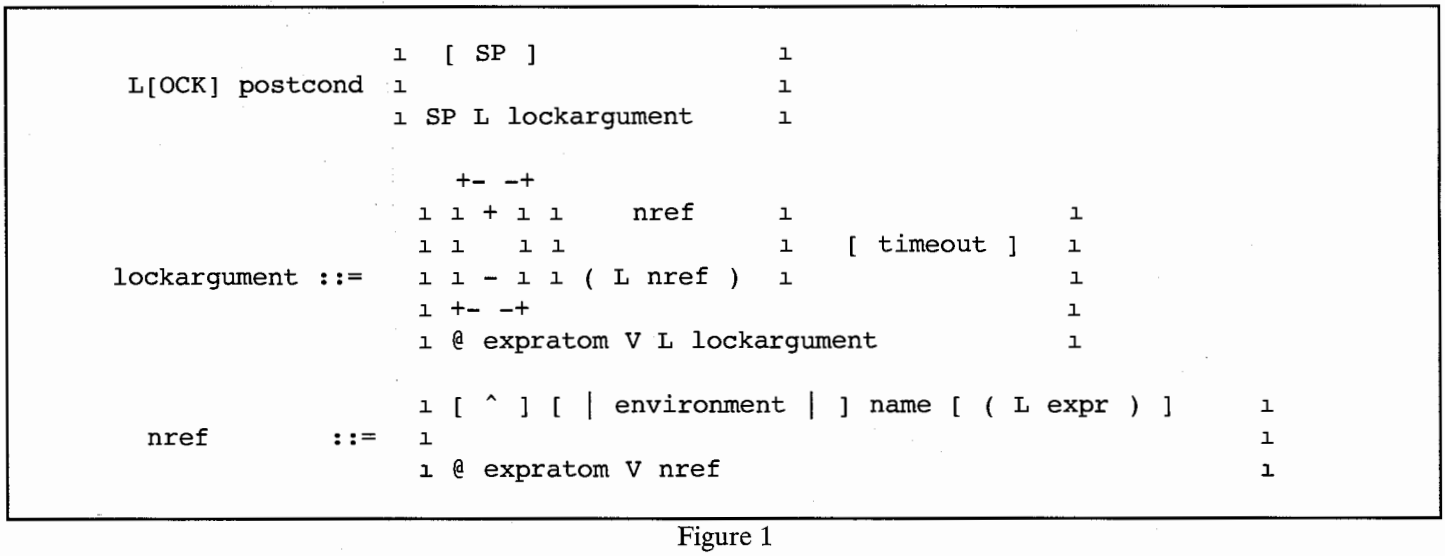

Figure 1

request to get ^A and ^B simultaneously. In the latter you try to get ^B only for a given time.

What is a namespace? It is a subtree of a global or a local variable. For example, if you own a LOCK on:

```
^ACCOUNT("FirstBoston","Jones")
```

another job may own

```
^ACCOUNT("FirstBoston","Miller")
```

but your lock prevents another job from locking either a subtree of your reference e.g.,

```
^ACCOUNT("FirstBoston","Jones","David")
```

or something that your own namespace is part of e.g.,

```
^ACCOUNT("FirstBoston").
```

The similarity with globals is intended. But there is no connection between a global and locked names from the viewpoint of your M interpreter. There is no need for a global to exist to make it lockable. In fact, even if you own a lock, nobody is barred from messing with the actual global.

Some are surprised that "you can lock local variables," and they ask about the effects of this. Again there, nobody really puts your locals into a locker. And you can imagine a name as not being the name of a local variable but the name of . . . a routine, a device, a friend, just about anything. If you have a routine %SYSTEM, and you want just one job executing this routine at a particular time, then guard its entry with LOCK %SYSTEM.

While novices rapidly learn how to do useful things with M, any expert occasionally finds surprising new usage of M syntax. When we discussed the LOCK command, there was the opinion that sometimes LOCK is too restrictive: If I lock an object (i.e., global, program) before inspecting it, I do so to prevent another job from changing the values of this object simultaneously. But I do want to exclude others from inspecting this object in parallel. My opinion was to amend the M syntax to provide for this less restrictive LOCK.

To my surprise, MDC member Rod Dorman told me that he uses the LOCK command as it is to provide for this kind of functionality.

If you want to do an "inspect"-lock on DEVICE("LPT1") then issue a LOCK DEVICE("LPT1",$J). Other jobs may successfully issue a LOCK DEVICE("LPT1",$J) with their job number. As long as someone owns a lock of this kind, nobody can successfully LOCK DEVICE("LPT1"). And the other way around: As long as one owns DEVICE("LPT1") nobody can LOCK DEVICE("LPT1",$J).

My deepest respect to the people who dreamed up this language. Things come so easily and naturally . . . if you know about it. So share your experience with your peers! M

Winfried Gerum is president of Winner Software in Röttenbach, Germany. His column appears regularly in M Computing. *He can be reached at: wg@winner.de.*
