JUST ASK!

Question: We recently found the following code in a client's system:

```
FOR X=X:1:X+10 S $P(^DATA(X),";"
,3)=1
```

We know this is wrong—because X keeps getting reset, this just keeps going forever. The problem is, the code appears to work. What's going on?

Editors: This is a misunderstanding of how the FOR command works. In the example you gave (we changed the code to make the example clearer and to protect you and your client), we assume the variable X was given a value prior to entering this line.

The argument of the FOR has two components: the index variable (the variable to the left of the equal sign) and the parameters that define the loop (the three expressions to the right of the equal sign). When M encounters a FOR, it evaluates all the expressions in the parameters first and saves these, internally, as three values: starting value, incrementation value, and ending value.

Once the parameters are evaluated, M sets the index variable to the starting value and processes the rest of the line. When the line is complete, M returns to the FOR command to decide what to do next. If the current value of the index variable plus the incrementation value exceeds the ending value, M is done with this FOR. (If the incrementation value is negative, M looks to see if the sum is less than the ending value.)

Otherwise, M increments the index variable by the incrementation amount and processes the line again.

Notice that the *expressions* in the parameter list were evaluated only once. Another way of illustrating this is the following:

```
+1 SET START=1,INCREMNT=2,END=10
+2 FOR INDEX=START:INCREMNT:END K
 START INCREMNT,END W !,INDEX
```

This code will produce the following:

```
1
3
5
7
9
```

In your example code, the variable X is used as the starting value expression, as part of the ending value expression, and also as the index variable. Since the expressions are evaluated only once, the FOR loop will run eleven times.

Question: I just noticed that the contains operator on my system returns true when I ask if my string contains a null. I've checked this on two other implementations and they all appear to do the same thing. Now that I know this I can code for it, but why do they do that?

Editors: The answer lies in basic set theory. The following is taken from *Basic Abstract Algebra*, by P.B. Bhattacharya, S. K. Jain, and S.R. Nagpaul, 1986, Cambridge University Press: "Let A and B be sets. A is called a *subset* of B if every element of A is an element of B . . . Further, if A is a subset of B, we also say that B contains A. It follows immediately from the definition that A and B are equal if and only if A is a subset of B and B is a subset of A. Thus, every set is a subset of itself. Moreover, the empty set is a subset of every set because the condition X belongs to the empty set implies that X belongs to A is satisfied vacuously."

"Satisfied vacuously" means that "there is no element X in either [the empty set] or A to which the condition may be applied."

And all this is only page 4!

M follows the same logic, and any string contains the empty string. Thus: `A[""` is always true.

Your question also raises two other points. First, the term "null" can be confusing. There is a character called NUL (`$C(0)`), which is rarely used in M programs. Note the following code:

```
SET X=$C(0) W $L(X),""[X!
SET Y="" W $L(Y),""[Y
```

will display

```
10
01
```

X contains a NUL, while Y contains an empty string.

Finally, do not depend on "what the system does" to decide what is standard and what is not. It is possible (and has happened) that several implementations behave the same, and are all wrong.

Question: We use a file in which we keep device control codes. Recently

we installed an upgrade and noticed some changes. In the old version, the code for clearing the screen on a VT220 was:

```
#,*27,*91,*50,*74,*27,*91,*72
```

In the new system, the code is:

```
#,$C(27,91,50,74,27,91,72)
```

In both cases the screen cleared and the cursor was in the upper left corner. The first code left $X=0, however, while the second code left $X=7. Why is there a difference? I thought the W *X and W $C(X) were interchangeable.

Editors: The standard does not treat $C(X) and *X the same. $CHAR has a very specific, unambiguous meaning as long as you stay in the ASCII character set. $CHAR and $ASCII are exact inverses of each other with the exception of the handling of ASCII 0. The effect of W $CHAR (something) on $X and $Y is predictable.

The WRITE *X (as well as the READ *X) are in the original standard as "escape hatches" for the vendors to do terminal handling not provided by the standard. As such, the side effects are completely implementation-dependent. You will find that, generally, systems that updated $X the same for W *X as for W $C(X) also provided the (nonstandard) capability of SETing $X to allow for correction. (This was true several years ago, but is less true now as more vendors provide SETable $X and $Y in anticipation of the next standard.) Thus, if the vendor is supplying the code to clear the screen, and is using WRITE * syntax, it will generally look something like:

```
W *27,*91,*50,*74,*27,*91,*72
S $X=0,$Y=0
```

This problem with handling devices is probably why you have a file of device functions to begin with. The functionality you want to perform is, by definition, nonstandard and nonportable.

All this will change with the next standard. The new X3.64 binding has several features not currently available. First, there is the potential for greatly increased functionality beyond the W #!?X capability we now have. Second, we can find out what controlmnemonics are supported programmatically. So if W /CUP does not move the cursor to the upper left corner, you'll be able to tell by checking the new $DEVICE special variable. ❖

Just Ask! is the forum for sharing the wealth of information our community has acquired. Send your questions to the Just Ask! editors at *M Computing*. We regret we cannot respond to requests except in this space.

New Products for M Users

Arnet Corp. (Nashville) has a report for value-added resellers (VARs), system integrators, MIS professionals, and users who need more information on serial communications. "How to Get Optimum I/O Performance" covers connectivity solutions, multiuser boards' functions and features, and technical information on multiuser boards. It should be useful to those environments with a variety of operating systems and computer platforms. Contact Linda Cullum, Marketing Relations, Arnet Corp., #6, 618 Grassmere Park Drive, Nashville, TN 37211. Phone: 615-834-8000; Fax: 615-834-5399.

Antrim Corp. (Plano, TX) now can port its laboratory systems, financial systems, and blood bank systems to **Digital Equipment Corp.**'s (Marlboro, MA) 64-bit Alpha AXP RISC platforms. Digital's computing architecture, intended to last for the next twenty-five years, supports fast, uniprocessor and multiprocessor implementations from palmtops to supercomputers. The two companies are offering Alpha AXP technology to all medical laboratories in the United States and Canada.

InterSystems Corp. (Cambridge, MA) just announced a new version of its Open M/SQL relational database-management system for use with M systems from **Micronetics Development Corp.** Open M/SQL runs on top of a number of host ANSI M systems, and it is apparently the first product to enable SQL queries to be embedded in M programs that comply with the draft ANSI/ISO SQL2 standard. This brings to three the number of Open M/SQL products available for use with MSM: Developer, M/PACT, and RDBMS Engine.