# Indirection

*by Frederick Hiltz, Stage Manager*

Indirection, boon or bane? At the 1995 MTA conference I heard two comments:

*No good or useful MUMPS application can be built without using indirection.*

*Indirection should be banned. It causes more bugs than any other part of M.*

In the light of such divergent opinions, what is the place of indirection in good M code? Like any tool, it can be used for good or for harm. Anyone who has watched an expert sushi chef or butcher wield a razor-sharp knife can appreciate the speed, efficiency, and—yes—beauty of a tool skillfully applied. But would you hand that knife to a beginner?

The more we know about the tool, the better we can use it. Good M code communicates a method from one programmer to another; indirection can help or hinder the process.

***Comprehension*** - Indirection is often incomprehensible because of two flaws common to any bad code: meaningless variable names and lack of useful comments. Anonymous references to @GLO and @REF litter the world of M. However, indirection can aid understanding:

```
N ORDERN ; # of a medications order.
N MEDN   ; A medication number.
N XMEDLIST ; -> list of patient's
                meds.
S XMEDLIST=$NA(^|"OE"|OE("RI","
     ID.ST.TY.IMNU",ID,STATUS,TYPE))
  ...
S MEDN=""
F  S MEDN=$O(@XMEDLIST@(MEDN))
Q:MEDN=""  D
. S ORDERN=$O(@XMEDLIST@(MEDN,""))
. ; processes the order.
.    ...
Q
```

This example from my shop hides some variables that are not essential at the moment (ID, STATUS, and TYPE) and focuses our attention on the list of a patient's medications.

I like to reserve X-variables for indirection as a reminder. The use of more than one indirection in an expression requires serious commenting!

Indirection baffles some code maintenance tools, reducing comprehension. The better tools, because they analyze more than one line of code at a time, do handle simple cases like this one.

***Generality*** - Whenever more than a line or two of code is duplicated, or almost duplicated, the expert programmer combines and generalizes their functions. This costs time at first, but builds a kit of tested general functions to be reused for increased productivity in the long run. The following review of indirection in M should suggest ways to generalize your code.

***Indirection in M*** - The @ sign introduces indirection in the form *@term*. The value of *term* replaces *@term* when the M code is executed. One might wish the value of *term* to be any string that is legal in its context, but M restricts it to four types. The ANSI standard contains more detail than space allows here:

***Name indirection*** - *@term* appears where the name of a variable would be used. The value of *term* is a literal string or number, the name of a variable (perhaps with subscripts), an intrinsic or extrinsic function, or an expression surrounded by parentheses.

```
S XPRICE=$NA(^INVENTO
  ("BOSTON",PARTNO,"PRICE"))
S @XPRICE=55.89
S DELIVER=$H+5,XWHAT=
      "DELIVER"
S TICKLER(@XWHAT)=WHAT
; now TICKLER(56605)=
       "DELIVER"
```

***Array indirection*** - *@term*@(sub,sub,...) Zero or more leading subscripts of an array reference come from the value of *term*. The final subscripts appear explicitly. The previous example of a patient's medications illustrates array indirection.

***Argument indirection*** - *@term* is a complete argument for most M commands; FOR is the notable exception.

```
S XARG="DELIVER=$$YESERD"
        S @XARG
S XARG="RIDE^CHARIOT:
       MALE" D @XARG
```

***Pattern indirection*** - *@term* appears where a pattern code would be used, and the value of *term* is a pattern code.

```
S XPATTERN=^DICT("SURNAME"
       ,"PATCODE")
I LASTNAME?@XPATTERN D
       REGISTER
```

***Performance*** - Our nearly 30-year-old language is up-to date; today's buzzword for delaying the evaluation of syntax until execution time is "late binding." Does it hurt performance? The compiler cannot evaluate indirection in advance, but some can save the result during execution to be reused. Today's processors are so much faster than disks that the saving from smaller, more general routines usually pays the cost of indirection and leaves some profit. **M**

---

Frederick L. Hiltz, Ph.D., develops medical information system software at Brigham and Women's Hospital, Boston,Massachusetts. fhiltz@bics.bwh.harvard.edu

---

Do you have a question that deserves discussion? Have you found a good answer to someone else's question that you would like to share? How about a controversial question and a discussion of pros and cons? If you prefer that your name not be published, please say so in your contribution, which should be sent to the Managing Editor at *M Computing*.

**1996 Editorial Schedule**
***M Computing***

July 1996 Issue
Deadline for submission:
April 15, 1996
Development Environments

September 1996 Issue
Deadline for submission:
June 15, 1996
M and the Internet

December 1996 Issue
Deadline for submission:
September 15, 1996
Business & Finance

Please submit articles to:
Managing Editor
M Technology Association
1738 Elton Rd., Suite 205
Silver Spring, MD 20903