Free Lunch for All M Programmers

Winfried Gerum

by Winfried Gerum

Writing articles for *M Computing* is rewarding by itself. But feedback from a reader (hey, is there someone?) to the author is very rare indeed. With a delay of four years I now have a response to my previous article "Double your disk space for $49.95." Despite the fictitious price tag, my reader expected a kind of free lunch. He bitterly complained about the complete uselessness of the published algorithm because it took impossibly long to execute on his 386 machine and because of compression ratios on his files not living up to expectations. However, he came up with ways to improve the compression ratios on his files, albeit at some cost to the system resources.

Everyone knows that there is no such thing as a free lunch.

But now the rest of the story: sometimes there is a free dessert.

My reader did not want to compress any files under the control of an M routine, but especially "Global Save" files. This is hardly surprising. M routines are small anyway, especially when written in traditional style. But global save files can become quite large and they have a structure that compresses well.

Usually a global save file is the output produced by code similar to that in Figure 1.

If you look at a Global Save file you see that every other line features a full global reference. Adjacent references usually share some of the subscripts. Remember, M has an intrinsic mechanism to make use of this situation. It is called "naked reference." There has been much discussion about the merits and dangers of naked reference. The point is that M is good at processing a naked reference. Programmers are sometimes not as good at using the naked reference.

```
. . . OPEN file for writing, USE it . . .
. . .WRITE header . . .
SET X=$NAME(^GLOBAL)
FOR   SET X=$QUERY(@X) QUIT:X=""   WRITE X,!,@X,!
. . . WRITE trailing lines . . .
```

To restore the global, you simply:

```
. . .OPEN file for reading. . .
. . .skip header. . .
FOR  READ X QUIT:X'?1"^".E  READ Y SET @X=Y
. . .skip trailing lines. . .
```

Figure 1

Most global references in a global list file can be expressed in the form of a naked reference relative to the previous global reference. If you do so, the resulting global save file is usually significantly smaller than one in the traditional format. The lunch is not for free: the time to produce the global save file increases. But there is a free dessert: the global restore routine does not need any modifications and it runs in equal or better time!

Figure 2 shows the core part of a global save routine that produces the desired naked references.

As this takes somewhat longer to execute, you might want to do the global save quickly and convert the traditional format to the thrifty format later. This can be done by:

```
. . .OPEN IN for reading . . .
. . .OPEN OUT for reading. . .
. . .copy header. . .
SET Z=""
FOR READ X QUIT:X'?1"^".E
DO
.READ Y WRITE
$$NREF(Z,X),!,Y,! SET Z=X
. . .copy trailing lines. . .
```

Using this function $$NREF (see Figure 3) there is a straightforward way to modify the traditional global list procedure:

```
Naked global list procedure variant #1

          ;procedure GL(gvn)
          ;produces a listing of an M global variable
          ;e.g. G=”^PATDAT” or G=”PATDAT(4711)”
          ;the listing prints the contents of the global variable
          ;in the form reference, NEWLINE,data,NEWLINE
          ;in order to save space in the output file. “reference”
          ;may be an M naked reference that gets its meaning
          ;from the previous reference. A corresponding %GRESTORE
          ;program looks somewhat like
          ;         FOR  READ X,Y QUIT:$$ENDOfFILE  SET @X=Y
 GL(G) NEW D,N,X ; D,X=vars of GLI, NEWed here for efficiency
          SET (N,X)=”” ; naked indicator,X=last subscript
          DO GLI(G)
          QUIT
          ;GLI:  only internal use by procedure GL
          ;G=subscript to be processed
          ;if @G has a value       —>print reference and data
          ;if @G has descendants  —>recursive proc. of descendants
GLI(G)   SET D=SDATA(@G)
          WRITE:D#2 $SELECT(N-””:G,1:N_$$IND(X)_”)”),!,@G,!
          SET N=$SELECT(X-””:””,D=1:”^(“,D=11:”^(“_$$IND(X)_”,”,
——>N=””:””,1:N_$$IND(X)_”,”)
          QUIT:D=1
          NEW X SET X=””
          FOR  SET X=$ORDER(@G@(X)) QUIT:X=””  DO GLI($NAME
——>(@G@(X)))
          SET N=””
          QUIT
          ;produces an M-expression that evaluates to X
          ;if X is canonic numeric, it is returned unchanged
          ;otherwise quotes are doubled and a quote is put at
          ;the beginning and the end of the string
IND(X) IF X’[“E” QUIT:+X+X X ;!!Beware X=”1E999999”
          IF X[“”””” NEW I FOR I=$LENGTH(X):-1:1 SET:$EXTRACT(X,I)
——>””””” $EXTRACT(X,I)-”””””””””
          QUIT “”””_X_””””
```

Figure 2

```
;G1,G2 are full global refs.
;G1 is meant to define the naked indicator
;if G2 can be expressed with a naked reference
;relative to G1, the proper naked reference is returned
;otherwise the unchanged value of G2 is returned.

NREF(C1,C2) QUIT:G1’?1”^”1.E1”(“1.E1”)” G2 ;naked undefined
          QUIT:G2’?1”^”1.E1”(“1.E1”)” G2 ;G2 not a subscripted gl.
          NEW B,C,I,J,P,Q
          ;get common header chars
          FOR I=1:1 QUIT:$E(G1,I’=$E(G2,I)
          SET J=$F(G1,”(“) QUIT:J>I G2
          ;go back to beginning of subscript
          SET P=J-1,Q=0,S=0
          FOR J=J:1:I-1 SET c=$E(G1,J) SET:C=”” Q=’Q DO:’Q

          .SET:C=”(“ B=B+1
          .SET:C=”)” B=B-1
          .I ‘B,C=”,” S P=J
          ;is this the last subscript in G1?
          SET Q=0,B=0
          F J=P+1:1:$L(G1) S C=$E(G1,J) S:C=”””” Q=’Q D:’Q Q:’P
          .SET:C=”(“ B=B+1
          .SET:C=”)” B=B-1
          .IF ‘B,C=”,” SET P=0
          QUIT:’P G2
          QUIT “^(“ $E(G2,p+1,$L(G2))
```

Figure 3

```
Naked global list procedure, variant 2

. . .OPEN file for writing, USE it. . .
. . .WRITE header. . .
SET X=$NAME(^GLOBAL), Z=""
FOR  SET X=$q(@X) quit:X=""  WRITE $$NREF(Z,X),!,@X,! s z=x
. . .WRITE trailing lines. . .
```

Figure 4

On my system this variant 2 executed significantly faster than variant 1. The overhead of the recursive function call seems to be relatively expensive.

What can you expect for this naked global save? My reader told me that he would be happy to see a global save file of 3.5MB be compressed so that it would fit on a single (1.4MB) floppy disk.

I took two production globals, truncated a traditional global save file at the 3.5MB line with the following results (see Figure 5).

As you see, this simple modification brings some gain (file fits on two floppies instead of three). A global save with naked references takes more than twice the time to run. So, the first part clearly has a price. But it is an investment with some dividends. The global restore is slightly faster. And the smaller file fits on two floppies instead of three.

That is not good enough for our example problem. To make do with one floppy additional compression is needed. Usually applying multiple compression runs on the same data is useless. A well-compressed file cannot be compressed further. We tried several compression programs on our example files (see Figure 6).

	Global 1			Global 2		
	size		time	size		time
database occupancy	1374208			1944576		
global save full ref.	3561218	100%	412	3583007	100%	332
global save, naked var 1	1996443	56%	1479	2201678	61%	1150
global save, naked var 2	1996443	56%	947	2201678	61%	793
convert full to naked			1205			939
global restore, full ref.			342			292
global restore, naked ref.			332			284

Figure 5

•*compress* is the compression utility supplied with UNIX .

• *gzip* is the GNU compression utility, available for virtually every system. This is my favorite.

•*arco* arithmetic compression utility. My compression of last resort, takes eons to execute. Compression rates are frequently (but not always) much better than that of all other programs.

The external compression routines easily squeeze a traditional global save to much less than the target size of 1.4MB. However, when they act on the naked global save, they work consistently faster both in compression and decompression phase and they produce proportionally smaller files. Using naked global save plus an external compression routine, your floppy is worth the equivalent of about 10MB of a traditional global save file.

	Global 1			Global 2		
	size		time	size		time
global save full ref	3561218	100%		3583007	100%	
gzip on save w/full ref.	586226	16%	110	809747	23%	98
...reverse gzip			6			9
gzip on save w/naked ref	290212	8%	72	551279	15%	74
...reverse gzip			2			5
compress full ref	778921	22%	14	892980	25%	14
...reverse compress			10			11
compress naked ref	488559	14%	10	639869	18*	9
...reverse compress			5			7
arco full ref	424916	12%	695	534760	15%	825
...reverse arco			691			820
arco naked ref	331460	9%	525	435136	12%	669
...reverse arco			519			661

Figure 6

When I get a reminder to submit a manuscript before an impending deadline, it does not give me any idea what to write about. On the other hand, a good question or sincere criticism is half a new article. So next time you find something completely useless or if you miss something, give us some feedback. Maybe I will invite you to another free lunch, pardon, free dessert I mean.

Winfried Gerum is with Winner Software GmbH in Rottenbach, Germany. You may contact him by phone at 011-49-9195-940022 or by fax at 011-49-9195-940030