foxtalk™

A comprehensive monthly guide for users of FoxBASE+™

October 1989 Issue

$10.00 per issue

DIMENSION, SCATTER, AND GATHER COMMANDS

By J. J. Colbert

FoxBASE+ HAS THREE commands which are not found in dBASE III+ and which dBASE IV has only partially emulated. DIMENSION allows the programmer to create one- or two-dimensional arrays efficiently, while SCATTER and GATHER allow for efficient transfer of data between arrays and database records. SCATTER moves data from record(s) to an array, while GATHER moves data in the opposite direction. Together, these three commands allow one to write concise, generic (universally applicable) code for moving data in and out of records.

With the development of large databases and their use by many individuals, and data entry and edits being carried out by several people over a number of years, problems such as erroneous entries, multiple records, and related errors can easily arise and are difficult to locate. One such problem, that of multiple records describing identical data, can be corrected quickly and efficiently using the DIMENSION, SCATTER, and GATHER commands.

As an example, the following program was tested on a database that contained several thousand records, with data entered at various times over the past four years. These data were from an application in forestry, for which a number of locations were visited annually. At each location, a series of plots were established, trees within plots were tagged, and yearly measurements were taken on each tree.

Thus, each record in the database was to store the data from a single tree. LOCATION, PLOT, and TREE variables make up the unique key to associate individual records with actual physical entities on the ground. In some cases, trees entered the study as they matured enough to be "counted" in the samples; in others, trees died and no further data were to be collected on them. In any event, there should be only one record for any [LOCATION, PLOT, TREE] triple found in the database. How does one find suspected duplicate records and what does one do with them once found? It is not efficient to traverse the full database using the edit or browse facility in an attempt to find and repair all such problems. Shouldn't we let a program do most of that work for us?

The scientists involved in the study requested that I devise a program that would directly handle simple duplicates and flag more complex problems for more detailed review. The SCATTER, GATHER, and DIMENSION commands allow one to write the code in such a way that only the name of the database, the index file, and the number of fields used in indexing are required inputs. All other information related to the problem, such as field names, field

(continues on page 3)

— INSIDE —

Editorial 2
Bit Testing 5
Subscriber Forum 6
Three Screen Procedures 8
The Magic of Publicity (Part 2) 11
Deduping 13
Recap of the Developers Conference 18
Faster Search Program 20

PINNACLE PUBLISHING, INC.

foxtalk™ *A comprehensive monthly guide for users of FoxBASE+™*

foxtalk (ISSN#1042-6302) is published monthly (12 times per year) by Pinnacle Publishing, Inc., 28621 Pacific Highway South, Federal Way, WA 98003. Cost of domestic subscriptions: 12 issues, $99.00; 24 issues, $180.00. Outside USA: 12 issues, $119.00; 24 issues, $210.00. Single copy price: $10.00; outside USA, $12.00. Individual source code disks: $10.00; 12 source code disks, $60.00. All funds must be in U.S. currency. Back issues are available upon request, for the same price as a single copy. Second class postage pending at Auburn, WA. POSTMASTER: Send address changes to **fox**talk PO Box 8099, Federal Way, WA 98003.

Address all editorial correspondence, requests for special permission, or bulk orders to The Editor, **fox**talk PO Box 8099, Federal Way, WA 98003. Phone: 1 800 231-1293 or 206/941-2300.

Questions of a technical nature can be directed to Glenn A. Hart, Editor, care of Pinnacle Publishing, Inc., PO Box 8099, Federal Way, WA 98003.

FoxBASE+ and FoxPro are trademarks of Fox Software. Other brand and product names are registered trademarks or trademarks of their respective holders.

Copyright © 1989 Pinnacle Publishing, Inc. All Rights Reserved. No part of this periodical may be used or reproduced in any fashion whatsoever (except in the case of brief quotations embodied in critical articles and reviews) without the prior written consent of Pinnacle Publishing, Inc.

Publisher, David M. Johnson
Editor, Glenn A. Hart
Managing Editor, Brent P. Smith
Layout and Design, Brett Borders
Editorial Assistant, Curt Malloy

This publication is intended as a general guide. It covers a highly technical and complex subject and should not be used for making decisions concerning specific products or applications. This publication is sold as is, without warranty of any kind, either express or implied, respecting the contents of this publication, including but not limited to implied warranties for the publication, quality, performance, merchantability, or fitness for any particular purpose. Pinnacle Publishing, Inc. shall not be liable to the purchaser or any other person or entity with respect to any liability, loss or damage caused or alleged to be caused directly or indirectly by this publication.

Volume I, Issue 8

EDITORIAL

We've seen the future, and it works! FoxPro is its name. Officially introduced at the Fox Developers Conference in Toledo (recapped by yours truly this month), FoxPro should be reaching dealer shelves right around the time you're reading this. To say you'll be impressed may qualify as the understatement of the eon.

Of course, **fox**talk will be providing you with lots of FoxPro code as our increasing roster of top developers and programmers begins to plumb the depths of this rich programming environment. Tom Rettig said at the Developers Conference that it may take two years for everyone to really investigate FoxPro's possibilities. I don't know if it will take *that* long, but we're getting started here in **fox**talk already. Pat Adams uses FoxPro's powerful new SCATTER MEMVAR BLANK and GATHER MEMVAR syntax to greatly shorten the code in her examination of the always challenging topic of duplicate records. Pat advocates and illustrates techniques for avoiding entry of duplicate records in the first place, which is obviously desirable.

Jim Colbert also deals with duplicate records. He too uses SCATTER and GATHER, but within FoxBASE+. A comparison of his code and Pat's may be instructive. Also, there are various programs available via shareware for attempting to deal with identifying, and, sometimes, removing duplicate records which already exist in a database; perhaps we'll review them at some point in the future.

New contributor Len Levy supplies three useful and attractive screen procedures written in FoxBASE+. Len was at the Developers Conference and was most enthusiastic about FoxPro. His routines simulate in some respects screen elements which are standard or easy in FoxPro, and the lengths Len goes in FoxBASE+ to accomplish similar objectives is a fine illustration of why FoxPro will make such screen elements so much easier to create and use.

We welcome several other new contributors this month. Herman Rohr provides an interesting fast search method using multiple indexes. Chris Connelly offers a bit testing routine which is extremely useful for working with data collected from lab instruments. Cavan F.E. Adolphe's calendar routine is neat and sweet, and should be handy as an appealing addition to many applications. John Bauman contributes some fine code to make handling memo fields easier in FoxBASE+ (of course, memo fields are *infinitely* superior in FoxPro).

And last, but far from least, the second half of David Irwin's article on publicity finally sees the light of day.

I find it especially gratifying to welcome our new contributors. **fox**talk is growing very rapidly (it's by far the largest of Pinnacle's technical publications), and adding new authors to our regulars is an important component of providing you with high quality code and ideas you can use. As we say from time to time, we very much welcome

(continues on page 22)

Commands Cont.

types, and database contents, are external to this program.

We decided to compare records pairwise; that is, to deal with records in pairs and make decisions relative to those pairs even though there may be more than two records with duplicate index triples. Next, if two records were exact duplicates, it was decided to forego further review and remove one of the pair.

If there were differences between two records in fields other than those used in the matching process, but one of these records merely contained the default data for that field, it was decided that the nondefault data would be retained in a composite record and a copy of the initial contents of the pair of records would be saved in an auxiliary database. Then, the initial contents could be compared with the composite record.

In the final case, where nondefault data are located in the same field on different records, no change was made to the original database, but those records were flagged by copying them to the auxiliary database for later review and editing outside of the program.

The main program opens the auxiliary databases, sets up the temporary memory arrays and variables, and does the search for matches on the primary database.

```
********              DEL_RECS.PRG              *********
*                                                       *
*    This program deleted records with duplicate        *
*    index fields if the other data are duplicate or    *
*    default; it retains essential data from records    *
*    to be removed.                                     *
*********************************************************

Private cntr,match,Fld_Cnt,Indx_Cnt

*  Set up data bases for copies of the altered records.

select a

* >> The following line should contain the database and
*    associated index(es) that will need review and
*    possible modification. <<

use DATABASE index IDX_FLDS

copy structure extended to template.001
select b
create Dup_recs from template.001
select c
create New_recs from template.001

*  Set database record size (number of fields) to memory
*  variable.

Fld_Cnt = Fcount()

*  Create the memory arrays for use in test logic and
*  data transfers.

Dimension Default(Fld_Cnt), Frst_Rec(Fld_Cnt);
        ,Sec_Rec(Fld_Cnt), Fnl_Rec(Fld_Cnt)

*  Set index field count
*  (number of fields in a record used in the index).

Indx_Cnt = 3

* Load Default array with the default value for each field.

select d
create temp from template.001
append blank
scatter to Default
delete file temp.dbf
delete file template.001

* Begin the review of the database ->

select a
goto top

*  NOTE: This program assumes that the fields used in the
*        indexing of records are the first fields in each
*        record.

Do while .not. eof()
   scatter to Frst_Rec
   skip
   scatter to Sec_Rec
   match = .T.
   Cntr = Indx_Cnt

* Since the third of the indexing fields changes the most
* rapidly in this case, we check it first.

   Do while (Cntr >= 1) .and. match
      If Frst_Rec(Cntr) <> Sec_Rec(Cntr)
         match = .F.
         endif
      cntr = cntr - 1
      enddo
   If match
      do rebuild
      endif
   enddo

pack
close databases
return
```

The Dup_recs database stores a copy of all pairs that are found to have nonduplicate data. New_recs contains a new record if one is composed from those in Dup_recs, or a copy of the Defaults where no composite record can be constructed. Use of default-filled records assures that composites can be accurately associated with the source data. There will be two records in Dup_recs for each record in New_recs.

Once a pair of records with duplicate indexes is found, the subprogram REBUILD is called. REBUILD.PRG reviews all of the remaining fields in a pair of records, decides what action is needed as a result of tests performed, and carries out copying of data and setting delete flag on those records to be removed.

```
*********                REBUILD.PRG              *********
*                                                         *
* This program is used to decide what data should be kept *
* from two records with duplicate entries in the fields   *
* used to index records.  It copies the two records in    *
* question to another database, transfers any nondefault  *
* data in either record to an output record in the original*
* database, and copies the newly constructed record to a  *
* second auxiliary database.                              *
***********************************************************

Private counter,required,differnt

*  Copy the needed data from the first of the two records.

skip -1
scatter to Fnl_rec
```

(continues)

```
skip 1

*  Set flag for non-duplicate records—"different" indicates
*  need to rebuild, "required" indicates the need to retain
*  both records for additional review by humanware.

required = .F.
differnt = .F.

*  Test for non-duplicate data: this process assumes that
*  if the data in a field differ between two records and
*  one is the default then the data from the other record
*  is to be saved.  If neither is the default, both are
*  saved and copied; and in this case a blank record is
*  appended to New_Recs database to maintain record spacing.

counter = Indx_Cnt + 1

Do while counter <= Fld_Cnt
   If Frst_Rec(counter) # Sec_Rec(counter)
      differnt = .T.
      If Default(counter) = Frst_Rec(counter)
         Fnl_Rec(counter) = Sec_Rec(counter)
      Else
         If Default(counter) # Sec_Rec(counter)
            required = .T.
            endif
         endif
      endif
   counter = counter + 1
   enddo

If .not. differnt
   skip -1
   delete
   skip 1
Else
* Copy records to Dup_recs database.
   select b
   append blank
   gather from Frst_Rec
   append blank
   gather from Sec_Rec
   select c
   If required
      append blank
      gather from Default
      select a
   Else
      append blank
      gather from Fnl_Rec
      select a
      skip -1
      delete
      skip
      gather from Fnl_Rec
      endif
   endif

return
```

With minimal modification, this program might be used under different circumstances. For example, I have made no attempt to consider case in character fields or values in numeric fields in the logic for deciding what to do with duplicate records once detected. One may also want to consider more complex decision logic to deal with situations where there are more than two records with the same index values. DIMENSION, SCATTER, and GATHER are the essential ingredients to the efficiency of this code.

Without the SCATTER and GATHER commands, one would be required to reference fields by name and use the REPLACE <field name> WITH < memory variable>... syntax. Thus, aside from adding substantially to the amount of initial code, the programmer would be required to add to or repeatedly modify the code with each new database to be examined.

This program uses only the name of the database, index file, and the number of entries in the database that compose the index. Thus, this routine could be written as a three-parameter procedure. Note that one also might generalize the procedure using the FCOUNT( ) and FIELD ( ) functions so that index fields are not required to be at the beginning of each record.

But the real power of the DIMENSION command can be fully realized with the complimentary use of the SCATTER and GATHER commands as demonstrated. I have used this code together with a menu-driven data entry system and report facility to allow biological field technicians with little knowledge of computers and database systems to enter, edit, and correct database files. We have found that this is an economical use of personnel and minimizes the need for review and intervention by programmers or other scientists.

About the Author:

Jim Colbert is a research mathematician with the USDA Forest Service's Northeastern Forest Experiment Station at Morgantown, WV. He became interested in writing FoxBASE+ software as a result of volunteer work with local youth soccer programs. He received a Ph.D. in mathematics from Washington State University in 1975 and currently is working on the development of computer software related to forest pest problems.

EOF

Source Code Disk Subscription Program Saves Time and Money

Attention **fox**talk subscribers: As you can see, **fox**talk is filled with very valuable and useful source code. If you would like to receive a source code disk each month with your publication, please seriously consider the following offer.

Pinnacle Publishing regularly offers a source code disk subscription program. For just $5.00 per month*, subscribers to this service automatically receive a disk each month containing all of the source code found in that month's issue of **fox**talk. This program is designed to save subscribers hours of laborious rekeying of published source code.

Call us at **1 800 231-1293** or **206/941-2300**; or write to us at **Pinnacle Publishing, Inc., PO Box 8099, Federal Way, WA 98003**

* Based on one-year source code subcription for $60.00. Individual disks are $10.00 each.

Bit Testing

By Chris Connelly, CCP

Although FoxBASE+ and other dbase dialects provide a number of data types that can be manipulated via standard commands or functions, the BIT (Binary InTeger) is not included in this group. The program BITTEST.PRG demonstrates a simple technique that can be used for determining the bit settings of a character and, through multiple calls, a string of characters.

Today, FoxBASE, dBASE, and Clipper are being used to record testing information from a variety of peripherals. Such instrumentation as oscilloscopes, data loggers, and digital volt ohmmeters are being attached to personal computers via standard communication ports with dbase language interfaces provided by software products from vendors such as Pinnacle Publishing and SilverWare.

Many of these peripherals will return a status byte or bytes that indicate the current operating conditions. These status bytes usually indicate the presence or absence of a condition by a bit being on or off (0 or 1). For example, a digital volt ohmmeter could indicate that it is measuring volts in the .001 volt range by sending a status byte of "0." The binary representation of the value of the ASCII character "0" is 00110000. Consider the following chart:

BIT	Meaning
0	Meter in LOCAL mode
1	Readings scaled by 1
2	Readings scaled by 10
3	Readings scaled by 100
4	Readings scaled by 1000
5	Ohms
6	Volts
7	Amperes

The bits in this chart are numbered least significant (right most) to most significant. Bits labeled in this fashion correspond to the power of two represented, but not all manufacturers choose this representation and may indicate bit zero being the most significant bit.

In the program BITTEST, logical operators are established for each bit using this representation. A program could then use the information to send a message to the screen, or to perform another task:

```
IF BIT0
     @ 20,10 SAY "VOM IN LOCAL MODE"
     WAIT
ELSE
     @ 20,10 SAY "VOM UNDER COMPUTER CONTROL"
ENDIF
```

FoxBASE+ provides arrays to store like data types. Usage of an array of logical operators would be more machine efficient than the macro substitution (BIT&), and might be necessary for an instrumentation system. The code could be easily changed to accommodate this feature.

Although bit testing may be more common in process control situations, any number of FoxBASE+ applications utilize information downloaded from mainframes. Some old mainframe systems (where disk storage used to be very dear) used bits instead of bytes for flags in employee files. Indications of such things as sex (e.g., bit3 on = male), security clearances, and distribution lists were common in data files. The ability to examine these indicators can save many requests for reformatting from a data processing department.

```
*****************************************************************
*    BITTEST.PRG     Determines whether a bit is on (.T.rue)
*                    Input mybyte a single character
*                    output bit0 thru bit7 (bit 7 most significant
*                    logically true if bit is on
*****************************************************************

****GET A CHARACTER FROM KEYBOARD TO DEMONSTRATE PROCESS*********

set device to screen
clear
set talk off
store ' ' to mybyte
@ 12,20 say 'Enter character: ' get mybyte
read

*********HEART OF THE PROCESS STARTS HERE**********

store .F. to bit0,bit1,bit2,bit3,bit4,bit5,bit6,bit7
store 0 to mynum
store 128 to mypower
store 7 to mym
store ' ' to mymacro
store asc(mybyte) to mynum
do while mym > -1
     store str(mym,1,0) to mymacro
     if mynum > mypower .or. mynum = mypower
          store .T. to bit&mymacro
          store mym - 1 to mym
          store mynum - mypower to mynum
          store mypower / 2 to mypower
     else
          store mym - 1 to mym
          store mypower / 2 to mypower
     endif
enddo

*************PROCESS ENDS HERE********************

***************DISPLAY RESULTS FOR DEMONSTRATION PURPOSES********

? bit7
? bit6
? bit5
? bit4
? bit3
? bit2
? bit1
? bit0

* End of Program
```

About the Author:

Chris Connelly, BA, CCP, is currently Vice President and senior aerospace consultant with Doorway to Memory, a custom software house in Pasadena, California. He has participated in the development of a wide variety of systems over the past two decades ranging from board level embedded software to super computing. A member of the Society of Manufacturing Engineers for over 10 years, his comments and writings have appeared in such publications as ***Datamation***, ***InfoSystems***, *and* ***Mini Micro Systems***.

EOF

Subscriber Forum

Reader John Bauman sends the following program, which makes editing memo fields with FoxBASE+ 2.10 easier. The code is pretty much self-documenting. The structure for the sample database used is:

```
Field  Field Name  Type         Width    Dec
    1  CHAR        Character       10
    2  NUM         Numeric          5
    3  LOG         Logical          1
    4  MEMFIELD    Memo            10
** Total **                        27

*****************************************************************
* memodemo.prg      test memo field editing
*
* problem:          Foxbase Plus 2.1 doesn't allow for easy use
*                   of the memo field editor from within a program.
*
* logic:            This module is a sample editing session of
*                   testmemo.dbf which contains 4 fields, the 4th
*                   of which is a memofield, memfield.
*
*                   After editing other variables, the user is
*                   prompted as to whether or not memo field
*                   editing is desired.
*
*                   If desired, memedit.fmt is created using the
*                   SET ALTERNATE commands, and then set as the
*                   format file.
*
*                   memedit.fmt:
*                       @ 10, 10 say MSG
*                       set color to ,b/b
*                       @ 15,15 GET MEMFIELD    VALID MEMRET(.T.)
*                       set color to ,n/w
*
*                   The initial "set color" statement sets the
*                   foreground = background so that the "memo"
*                   block of the memo field is NOT displayed.
*                   The second "set color"  statement sets
*                   things back the way they were.
*
*                   The variable msg is initially defined as "",
*                   the null string, as we want the user to get
*                   straight into the memo edit session.
*                   Subsequently, however, msg is changed by the
*                   udf memret:
*
*                     PARA RET
*                     MSG = "Press:  <Ctrl Pgdn> to resume " + ;
*                         "editing  or <ESC> to quit"
*                     RETURN RET
*
*                   The dummy variable ret is just used to fulfill
*                   syntax requirements of FoxBASE+ udf's.
*
*                   Now, when the editing session is through, the
*                   user is returned to the format file screen with
*                   the read still active, but the prompt, msg, has
*                   been changed so that users now know they have
*                   the option to resume editing or quit. Actually,
*                   any of the READKEY() keys will exit the
*                   editing, except the ctrl pgdn set.
*****************************************************************

* set up working environment:

SET STEP OFF
SET TALK OFF
SET HELP OFF
SET STAT OFF
SET SCOR OFF
SET SAFETY OFF
SET BELL OFF
SET DELI OFF
SET HEAD OFF
SET ECHO OFF
SET DEBUG OFF
SET TITLE OFF

SET COLOR TO W+/B,N/W,B

notice = "copyright (c) 1989 John M. Bauman; all rights reserved"

SET PROCEDURE TO MEMODEMO

USE TESTMEMO             && .dbf containing memofield named memfield

IF RECCOUNT() = 0        && be sure we have a record to work with
   APPEND BLANK
ENDIF

SCATTER TO MEM           && sample non-memo fields

@ 3,20 SAY "MEMODEMO.PRG:  Demo of memo field editing"

@ 10,10 SAY "Enter character data:        " GET MEM(1)
@ 12,10 SAY "Enter numeric data:          " GET MEM(2)
@ 14,10 SAY "Enter logical data:          " GET MEM(3)
@ 23,10 SAY NOTICE

READ
GATHER FROM MEM

* prompt for editing of the memo field:

YN = "Y"
@ 16,10 SAY "Enter memo field data now, Y/N?" GET YN ;
  PICT "!" VALID YN $ "YN "
READ

IF YN = "Y"

   IF .NOT. FILE("MEMEDIT.FMT")
     * build a temporary .fmt file:

     SET CONSOLE OFF
     SET ALTERNATE TO MEMEDIT.FMT
     SET ALTERNATE ON

     ? "@ 10, 10 say MSG"
     ? "set color to ,b/b"
     ? "@ 15,15 GET MEMFIELD    VALID MEMRET(.T.)"
     ? "set color to ,n/w"

     SET ALTERNATE OFF
     SET ALTERNATE TO
     SET CONSOLE ON

   ENDIF .not. file("memedit.fmt")

   * stuff keyboard with <ctrl pgdn> to auto enter memo editing:

   KEYBOARD CHR(30)

   MSG = ""              && initially no message to display
   SAVE SCREEN           && optional save of previous screen

   SET FORMAT TO MEMEDIT
   READ
   SET FORMAT TO

   RESTORE SCREEN        && optional screen restore

   WAIT                  && just a pause to show restored screen

ENDIF yn = "y"

CLEAR
SET PROCEDURE TO
* kill fmt file or comment this out and leave it
! DEL MEMEDIT.FMT > NUL
CLEAR ALL
RETURN
* eof() memodemo.prg

PROCEDURE MEMRET
*************************************************************
*                         UDF MEMRET()
*************************************************************
* memret.prg is a "dummy" udf in a valid clause for memo field
*          editing from a .fmt file.  The only purpose of this
*          valid clause is to change the value of the variable
*          msg from null ("") to the message below, DURING A
*          READ, so that it may be displayed after editing the
*          memo field
```

(continues)

```
PARA RET
MSG = "Press:  <Ctrl Pgdn> to resume editing or <ESC> to quit"
RETURN RET  && .T.
* eof() memret.prg/udf
* eop() memodemo.prg
```

foxtalk foxtalk foxtalk foxtalk foxtalk foxtalk foxtalk

Cavan F.E. Adolphe sends a nifty little calendar routine that can be used in a wide variety of applications:

Here's a short, fast program to write a calendar on screen at coordinates you specify. The calendar is erased by hitting any key, and the underlying screen is restored.

Usage is "DO CALDRIVE WITH <date expr>,n,m". The first parameter is a date expression, and this date determines the month and year of the calender. One common call would use the character to date function, for example, CTOD("09/01/89"). The passed date is set blinking in the calendar. The second and third parameters are the screen coordinates at which you would like the calendar display to begin. The colors can easily be customized by experimenting with the color references in the program. A subroutine names CALENDAR; a UDF named EOM is also used.

Just the job for a "HOTKEY" invocation!

```
**************************************************************
*
* PROGRAM: CALDRIVE - WRITES A CALENDAR ON SCREEN
* USEAGE: "DO CALDRIVE WITH <date expression>,n,m"
* (n,m - positive integers, earliest valid date is 12/01/1582)
*
* AUTHOR: CAVAN F.E. ADOLPHE
*
* PROGRAMMER:  Cavan Adolphe
* ADDRESS: 3895 LAKE GARDEN DRIVE, FALLBROOK, CA 92028
* PHONE: (619) 723 1764 (w)
*
**************************************************************
*
* CALL WITH:
* PARAM1=DATE EXPRESSION
* PARAM2=CALENDER TOP LINE (0->15)
* PARAM3=CALENDER LEFT COLUMN (0->6)
*
PARAMETER DT,RW,COL
DT=IIF(DT<CTOD('12/01/1582'),CTOD('12/01/1582'),DT)
DTSAVE=DT               && KEEP PASSED DATE FOR BLINKING REFERENCE
RW=IIF(RW>15,15,RW)     && CHECK ON ROW BOUNDS
COL=IIF(COL>6,6,COL)    && CHECK ON COLUMN BOUNDS
SAVE SCREEN TO CSAVE
CURSE=SYS(2002)         && CURSOR OFF
DT=DT-DAY(DT)+1         && GET 1ST OF PASSED MONTH
CATTRIB=SYS(2001,'COLOR') && SAVE CURRENT COLOR
SET COLOR TO W+/N+
@ RW,COL CLEAR TO RW+9,COL+73
@ RW,COL TO RW+9,COL+73 DOUBLE
DO CALENDAR WITH DT-1,RW+1,COL+3          && PREVIOUS MONTH
SET COLOR TO W+/R
DO CALENDAR WITH DT,RW+1,COL+27           && PASSED MONTH
SET COLOR TO W+/N+
DO CALENDAR WITH DT+31,RW+1,COL+51        && NEXT MONTH
WAIT ''
CURSE=SYS(2002,1)                         && CURSOR ON
SET COLOR TO &CATTRIB                     && RESTORE ENTERING COLOR
RESTORE   SCREEN   FROM   CSAVE
RETURN
*     CALDRIVE
```

foxtalk foxtalk foxtalk foxtalk foxtalk foxtalk foxtalk

```
**************************************************************
*
* PROGRAM: CALENDAR - SUBROUTINE OF CALDRIVE
* PURPOSE: WRITES OUT 1 MONTH FOR PASSED DATE, ON SCREEN
* AT PASSED ROW, COLUMN
*
**************************************************************
*
PARAMETER DT,SROW,SCOL      && PASSED DATE, POSITION OF CALENDER
STORE DAY(EOM(DT)) TO ENDDAY
STORE SCOL+DOW(DT-DAY(DT)+1)*3-3 TO CURCOL,STCOL  && START COLUMN
STORE SROW+2 TO CURROW,STROW                      && START ROW
@ SROW,SCOL-2 CLEAR TO SROW+7,SCOL+21
@ SROW,SCOL SAY UPPER(CMONTH(DT))
@ SROW,SCOL+16 SAY STR(YEAR(DT),4)
@ SROW+1,SCOL SAY 'SU MO TU WE TH FR SA'
NDAY=1
DO WHILE NDAY<=ENDDAY
DO WHILE CURCOL<=SCOL+19.AND.NDAY<=ENDDAY
@ CURROW,CURCOL SAY STR(NDAY,2)        && WRITE OUT DAY NUMBERS
CURCOL=CURCOL+3
NDAY=NDAY+1
ENDDO
CURROW=CURROW+1
CURCOL=SCOL                   && RESET TO FIRST COLUMN OF CALENDER
ENDDO

IF MONTH(DT)=MONTH(DTSAVE)
DOWMS=DOW(DT)
BLDAY=DAY(DTSAVE)
COLCAL=MOD(BLDAY+DOWMS-2,7)+1
ROWCAL=(BLDAY+DOWMS-2)/7
SET COLOR TO *W+/R
@ STROW+ROWCAL,SCOL+COLCAL*3-3 SAY STR(BLDAY,2)
* WRITE OUT DAY NUMBER
SET COLOR TO W+/R
ENDIF

RETURN
```

foxtalk foxtalk foxtalk foxtalk foxtalk foxtalk foxtalk

```
* GETS END-OF-MONTH DATE OF PASSED DATE [UDF]
* LAST UPDATE 1/3/89
  PARAMETER PDATE                           && PASSED DATE
  CMONTH=MONTH(PDATE)
  PDATE=IIF(DAY(PDATE)<=28,PDATE+28-DAY(PDATE),PDATE)
  * 28TH OF MONTH OR GREATER
  DO WHILE MONTH(PDATE)=CMONTH
    * INCREMENT DATE WHILE SAME MONTH
   PDATE=PDATE+1
  ENDDO
RETURN PDATE-1                              && END-OF-MONTH
```

EOF

Three Screen Procedures

By Len Levy

As programmers, we have developed, and are continuing to develop, our own individual creative styles. We 'borrow' code from many sources, such as our superb monthly issue of **fox**talk, and spend endless hours copying code, analyzing results, rewriting, and rewriting again. All the while, our repertoire of programming techniques continues to grow and our creative skills increase; much the same as a musician, artist or sculptor. We programmers *are* creative artists, and the flexible medium in which we work is FoxBASE+ and the even richer, more exciting, and eagerly awaited FoxPro.

This delicate balance, form, and symmetry, is attained through the judicious use of contrast and repetition. The programs that we develop are subject to the same rules. Our clients judge our work, not only on the basis of "Does it do what it's supposed to do," but also, "is it appealing to work with?" Balance, symmetry, contrast, and repetition are integral factors in our finished product. What better way to gain consistency in the programs we develop than through use of procedures and functions?

We see fewer "plain vanilla," black and white, clinical-looking screens. The appealing and colorful (when available) input and display screens seen in a running application are certainly more "user–friendly," and considerably less inhibitive to data entry people. Why not concentrate to a greater degree on the output?

I use the following three procedures in most of the applications I write.

Title

'TITLE' is simple and short. When you run 'DO title WITH "<whatever>",' the screen is cleared and a frame appears around the screen edges with your <whatever> centered on the top line between ► and ◄ [Alt–16] or 'CHR(16)' and [Alt–17] or 'CHR(17)'.

Rather than clearing the screen, I find it more effective to do a partial clearing and keep the title and borders intact while in the same procedure, as in:

```
@ 1,1 CLEAR TO 23,78
```

If you use this partial screen clear, make certain your color parameters are the same as when you initially called the 'TITLE' procedure!

Simple, but effective and consistent.

```
*************************************************************
*                             TITLE                          *
*************************************************************

PARAMETERS msg
* Len Levy, Data Management Systems
***   Creates double frame in window with centered message
***   MSG = Message on top line to be centered
***      between '►' and '◄'
***
***   Syntax Example:
***
***   DO title WITH "ADD TO INVENTORY FILE"
***
*************************************************************
PRIVATE l,msg,old_col
l = (80-LEN(msg))/2
* capture original screen colors
STORE SYS(2001,"COLOR") TO old_col
SET COLOR TO gr+/b              && set your own colors here
CLEAR
@ 0,0 TO 24,79 double
@ 0,l-2 SAY "►"+REPLICATE(" ",LEN(msg))+"◄"
SET COLOR TO w+/b
@ 0,l SAY msg
SET COLOR TO &old_col           && restore original colors
RETURN

*************************************************************
```

Frame

'FRAME' is the first of two procedures that automate the display of a centered, framed, and shadowed box containing your message of up to 264 characters.

An error trap is built in, to warn you of starting too low on the screen and exceeding the 254 character limit.

The shadow produced is *narrow* and appears in opaque black, unless the background upon which the frame appears is black. In that case, the shadow appears grey.

In FoxPro, this procedure will be rewritten to allow to for a 1024 character message and a *transparent* shadow!

'FRAME' is called with six parameters:

1. The starting row position, which could be designated by a numeric value of your choice, 1 to 23, or the relative position, 'ROW( )'.

2. The message to be displayed can be passed directly as in 'DO frame WITH 12, "Hit Any Key",...', or you you can store the message in a variable and pass the variable to the procedure, as in:

   ```
   'STORE "Hit Any Key" TO msg'
   'DO frame WITH 12,msg,..............'
   ```

   Use 'STORE REPLICATE("$",250) TO msg' to try it out.

3. The color of box frame, typed in quotes, as in

   ```
   'DO frame WITH 10, "Hit Any Key","r/bg",...........'
   ```

   My personal preference for my title screens are Yellow text on blue background, (gr+/b) and the cyan frames are attractive.

4. The color of text to be displayed in the frame, as in:

   ```
   'DO frame WITH 10, "Hit Any Key","r/bg","w+/bg",...'
   ```

   This produces a bright white text on cyan background.

5. The color of 'Y' or 'N' prompt which appears on the line below the last line of the message *only* if the 6th

(continues)

parameter passed to the procedure is ".t.". Since there *must* be a value passed, a null ("") will satisfy the call, as in:

```
'DO frame WITH 10, "Hit Any Key","r/bg","w+/bg",""..
```

6. The final parameter is the True/False flag, ".t." or ".f.", which tells the procedure whether or not to display a prompt line below the message: 'Choice (Y/N)? '

If parameter 6 is ".t.", you must either initialize the variable 'yn' with 'PUBLIC yn' or 'yn=[Y]' before the call. All parameters with the exception of the first, must be sent between quotes, as in:

```
STORE "Y" to yn DO frame WITH 5, "Quit?", "r/bg",;
"w+/bg", "b/w",  ".t."

IF yn - "Y"  ... etc.
```

I regularly use the 'FRAME' procedure with macros either stored in a 'MEM' file or intitialized in their master procedure as a public variable. As an example:

```
PUBLIC color,yn
STORE " " to yn
STORE "[r/bg],[w+/bg],[b/g]" TO color
DO title WITH "ADD ITEM TO INVENTORY FILE"
    < your input procedure goes here >
    @ 1,1 CLEAR TO 23,78 && Clear screen but leave borders
DO frame with 10,"Add Another Item?",&color,".t."
    IF yn - "Y" ... etc.


*************************************************************
*                          FRAME                            *
*************************************************************
PARAMETERS ROW,msg,fr_color,tx_color,hi_color,getflag
* Len Levy, Data Management Systems
***
*** ROW - Row on which message is to appear
***   MSG - Any message to be centered and framed up to 254
***   characters
***      FR_COLOR - Color of frame,
***      in QUOTES (Ex.:  "r/bg")
***         TX_COLOR - Color to text to be displayed,
***         in QUOTES
***            HI_COLOR - Color of "Y" or "N" prompt
***            if getflag - ".T."
***            otherwise pass NULL to routine  (Ex.: "")
***               GETFLAG - Indicator of whether 'Y' or 'N'
***               prompt is to appear
*************************************************************
***   Syntax Example:

***   DO frame WITH 12,"Do You Wish To Quit?","r/bg",;
***      "w+/bg","b/w",".t."

***      Will display centered and framed query on line 12
***      plus 'Choice  (Y/N)? ' on line 13

***   IMPORTANT!! Initialize variable 'YN' to " " before
***   calling routine or declare PUBLIC yn

***   DO frame WITH row(),"Hit Any Key to Continue",;
***      "g+/n","b+/n","",".f."
***      Will display center and framed message on the
***      current line for 2 seconds before restoring
***      original screen

*************************************************************
PRIVATE ROW,msg,fr_color,tx_color,hi_color,getflag
PRIVATE l_msg,no_rows,xx,m1,m2,m3,m4,start
* Get source screen background color for generating 'shadow'
*    effect:
old_back=SUBSTR(SYS(2001,"color"),AT("/",SYS(2001,"color"));
   +1,AT(",",SYS(2001,"COLOR"));
   -(AT("/",SYS(2001,"COLOR"))+1))
IF ISCOLOR()
   sh_col="n/"+old_back     && Shadow color - BLACK on screen
   IF UPPER(old_back)="N"  &&    with OTHER THAN black back-
      sh_col="W/n"         &&    ground, otherwise WHITE
   ENDIF
ELSE
   fr_color="n/w"
   tx_color="n/w+"
   hi_color="w+/n"
   sh_col="b+/n"
ENDIF
old_col=SYS(2001,"COLOR")                   && Save PRIOR colors
SAVE SCREEN
STORE LEN(msg) TO l_msg
IF (LEN(msg)/72+1)+ROW>24
   IF ISCOLOR()
      SET COLOR TO R/W
   ELSE
      SET COLOR TO N/W
   ENDIF
   @ 11,12 CLEAR TO 13,67
   @ 11,12 TO 13,67 DOUBLE
   SET COLOR TO B/W
   @ 12,14 SAY "Starting Row Position is TOO LOW... ";
      +"Please Re-Select"
   *** Narrow SHADOW ***
   SET COLOR TO &sh_col
   @ 14,14 SAY REPLICATE(CHR(219),56)
   @ 13,68 SAY REPLICATE(CHR(219),2)
   @ 12,68 SAY REPLICATE(CHR(219),2)
   ? SYS(2002)                                 && turn cursor OFF
   xx=INKEY(2)
   ? SYS(2002,1)                               && turn cursor ON
   RESTORE SCREEN
   SET COLOR TO &old_col
   RETURN TO MASTER
ENDIF
IF l_msg>72
   LEFT - 2            && set left and right edges for frame
   RIGHT - 77
   DO CASE
   CASE l_msg>=217
      no_rows=4
      STORE LTRIM(TRIM(LEFT(msg,72))) TO m1
      IF AT(" ",m1)>0      && if at least one space in line
         STORE 0 TO COUNT
         DO WHILE COUNT <=LEN(m1)
            IF SUBSTR(m1,LEN(m1)-COUNT ,1)=" "
               STORE LEFT(m1,LEN(m1)-COUNT ) TO m1
               STORE RIGHT(msg,LEN(msg)-LEN(m1)) TO m2
               EXIT
            ENDIF
            STORE COUNT +1 TO COUNT
         ENDDO
         STORE LTRIM(TRIM(LEFT(m2,72))) TO m2
      ELSE
         STORE LTRIM(TRIM(RIGHT(msg,LEN(msg)-LEN(m1))));
            TO m2
         STORE LTRIM(TRIM(LEFT(m2,72))) TO m2
      ENDIF
      IF AT(" ",m2)>0
         STORE 0 TO COUNT
         DO WHILE COUNT <=LEN(m2)
            IF SUBSTR(m2,LEN(m2)-COUNT ,1)=" "
               STORE LEFT(m2,LEN(m2)-COUNT ) TO m2
               STORE RIGHT(msg,LEN(msg)-(LEN(m1)+LEN(m2)));
                  TO m3
               EXIT
            ENDIF
            STORE COUNT +1 TO COUNT
         ENDDO
         STORE LTRIM(TRIM(LEFT(m3,72))) TO m3
      ELSE
         STORE LTRIM(TRIM(RIGHT(msg,LEN(msg);
            -(LEN(m1)+LEN(m2))))) TO m3
         STORE LTRIM(TRIM(LEFT(m3,72))) TO m3
      ENDIF
      IF AT(" ",m3)>0
```

(continues)

```
      STORE 0 TO COUNT
      DO WHILE COUNT <=LEN(m3)
         IF SUBSTR(m3,LEN(m3)-COUNT ,1)=" "
            STORE LEFT(m3,LEN(m3)-COUNT ) TO m3
            STORE RIGHT(msg,LEN(msg);
               -(LEN(m1)+LEN(m2)+LEN(m3))) TO m4
            EXIT
         ENDIF
         STORE COUNT +1 TO COUNT
      ENDDO
      STORE LTRIM(TRIM(m4)) TO m4
   ELSE
      STORE LTRIM(TRIM(RIGHT(msg,LEN(msg);
         -(LEN(m1)+LEN(m2)+LEN(m3))))) TO m4
      STORE LTRIM(TRIM(LEFT(m4,72))) TO m4
   ENDIF
   SET COLOR TO &fr_color
   IF .NOT. &getflag                && No GET
      @ ROW-1,LEFT CLEAR TO ROW+4,RIGHT
      @ ROW-1,LEFT TO ROW+4,RIGHT DOUBLE
      *** SHADOW ***
      SET COLOR TO &sh_col
      @ ROW+5,LEFT+1 SAY REPLICATE(CHR(223),76)
      @ ROW+4,RIGHT+1 SAY CHR(219)
      @ ROW+3,RIGHT+1 SAY CHR(219)
      @ ROW+2,RIGHT+1 SAY CHR(219)
      @ ROW+1,RIGHT+1 SAY CHR(219)
      @ ROW,RIGHT+1 SAY CHR(219)
      @ ROW-1,RIGHT+1 SAY CHR(220)
   ELSE
      @ ROW-1,LEFT CLEAR TO ROW+5,RIGHT
      @ ROW-1,LEFT TO ROW+5,RIGHT DOUBLE
      *** SHADOW ***
      SET COLOR TO &sh_col
      @ ROW+6,LEFT+1 SAY REPLICATE(CHR(223),76)
      @ ROW+5,RIGHT+1 SAY CHR(219)
      @ ROW+4,RIGHT+1 SAY CHR(219)
      @ ROW+3,RIGHT+1 SAY CHR(219)
      @ ROW+2,RIGHT+1 SAY CHR(219)
      @ ROW+1,RIGHT+1 SAY CHR(219)
      @ ROW,RIGHT+1 SAY CHR(219)
      @ ROW-1,RIGHT+1 SAY CHR(220)
   ENDIF
   SET COLOR TO &tx_color,&hi_color
   @ ROW,(80-LEN(m1))/2 SAY m1
   @ ROW+1,(80-LEN(m2))/2 SAY m2
   @ ROW+2,(80-LEN(m3))/2 SAY m3
   @ ROW+3,(80-LEN(m4))/2 SAY m4
   IF &getflag
      STORE " " TO yn
      @ ROW+4,31 SAY "Choice  (Y/N)? " GET yn PICTURE ;
         "!" VALID yn$"YN"
      READ
      SET COLOR TO &old_col
      RESTORE SCREEN
   ELSE
      xx=INKEY(2)
      RETURN
   ENDIF
CASE l_msg>=145.AND.l_msg<=216
   STORE 3 TO endline
   no_rows=3      && But POSSIBLY '4' after parsing!!
   STORE LTRIM(TRIM(LEFT(msg,72))) TO m1
   IF AT(" ",m1)>0
      STORE 0 TO COUNT
      DO WHILE COUNT <=LEN(m1)
         IF SUBSTR(m1,LEN(m1)-COUNT ,1)=" "
            STORE LEFT(m1,LEN(m1)-COUNT ) TO m1
            STORE RIGHT(msg,(LEN(msg)-LEN(m1))) TO m2
            EXIT
         ENDIF
         STORE COUNT +1 TO COUNT
      ENDDO
      STORE LTRIM(TRIM(LEFT(m2,72))) TO m2
   ELSE
      STORE LTRIM(TRIM(RIGHT(msg,LEN(msg)-LEN(m1))));
         TO m2
      STORE LTRIM(TRIM(LEFT(m2,72))) TO m2
   ENDIF
   IF AT(" ",m2)>0
      STORE 0 TO COUNT
      DO WHILE COUNT <=LEN(m2)
         IF SUBSTR(m2,LEN(m2)-COUNT ,1)=" "
            STORE LEFT(m2,LEN(m2)-COUNT ) TO m2
            STORE RIGHT(msg,LEN(msg)-(LEN(m1)+LEN(m2)));
               TO m3
            EXIT
         ENDIF
         STORE COUNT +1 TO COUNT
      ENDDO
      STORE LTRIM(TRIM(m3)) TO m3
      IF LEN(m3)>72          && Gotta add a 4th line!
         STORE LEFT(m3,72) TO m3
         STORE 0 TO COUNT
         DO WHILE COUNT <=LEN(m3)
            IF SUBSTR(m3,LEN(m3)-COUNT ,1)=" "
               STORE LEFT(m3,LEN(m3)-COUNT ) TO m3
               STORE RIGHT(msg,LEN(msg)-(LEN(m1)+LEN(m2);
                  +LEN(m3))) TO m4
               EXIT
            ENDIF
            STORE COUNT +1 TO COUNT
         ENDDO
         STORE LTRIM(TRIM(m4)) TO m4
         STORE endline+1 TO endline
      ENDIF
   ELSE
      STORE LTRIM(TRIM(RIGHT(msg,LEN(msg);
         -(LEN(m1)+LEN(m2))))) TO m3
      STORE LTRIM(TRIM(LEFT(m3,72))) TO m3
   ENDIF
   SET COLOR TO &fr_color
   IF .NOT. &getflag                 && No GET
      @ ROW-1,LEFT CLEAR TO ROW+endline,RIGHT
      @ ROW-1,LEFT TO ROW+endline,RIGHT DOUBLE
      *** SHADOW ***
      IF endline=3
         SET COLOR TO &sh_col
         @ ROW+4,LEFT+1 SAY REPLICATE(CHR(223),76)
         @ ROW+3,RIGHT+1 SAY CHR(219)
         @ ROW+2,RIGHT+1 SAY CHR(219)
         @ ROW+1,RIGHT+1 SAY CHR(219)
         @ ROW,RIGHT+1 SAY CHR(219)
         @ ROW-1,RIGHT+1 SAY CHR(220)
      ELSE
         *** SHADOW ***
         SET COLOR TO &sh_col
         @ ROW+5,LEFT+1 SAY REPLICATE(CHR(223),76)
         @ ROW+4,RIGHT+1 SAY CHR(219)
         @ ROW+3,RIGHT+1 SAY CHR(219)
         @ ROW+2,RIGHT+1 SAY CHR(219)
         @ ROW+1,RIGHT+1 SAY CHR(219)
         @ ROW,RIGHT+1 SAY CHR(219)
         @ ROW-1,RIGHT+1 SAY CHR(220)
      ENDIF
   ELSE
      IF endline = 3
         @ ROW-1,LEFT CLEAR TO ROW+endline+1,RIGHT
         @ ROW-1,LEFT TO ROW+endline+1,RIGHT DOUBLE
         SET COLOR TO &sh_col
         @ ROW+5,LEFT+1 SAY REPLICATE(CHR(223),76)
         @ ROW+4,RIGHT+1 SAY CHR(219)
         @ ROW+3,RIGHT+1 SAY CHR(219)
         @ ROW+2,RIGHT+1 SAY CHR(219)
         @ ROW+1,RIGHT+1 SAY CHR(219)
         @ ROW,RIGHT+1 SAY CHR(219)
         @ ROW-1,RIGHT+1 SAY CHR(220)
      ELSE
         @ ROW-1,LEFT CLEAR TO ROW+endline+2,RIGHT
         @ ROW-1,LEFT TO ROW+endline+2,RIGHT DOUBLE
         *** SHADOW ***
         SET COLOR TO &sh_col
         @ ROW+6,LEFT+1 SAY REPLICATE(CHR(223),76)
         @ ROW+5,RIGHT+1 SAY CHR(219)
         @ ROW+4,RIGHT+1 SAY CHR(219)
         @ ROW+3,RIGHT+1 SAY CHR(219)
         @ ROW+2,RIGHT+1 SAY CHR(219)
         @ ROW+1,RIGHT+1 SAY CHR(219)
         @ ROW,RIGHT+1 SAY CHR(219)
         @ ROW-1,RIGHT+1 SAY CHR(220)
      ENDIF
   ENDIF
```

(continues)

```
    SET COLOR TO &tx_color,&hi_color
    @ ROW,(80-LEN(m1))/2 SAY m1
    @ ROW+1,(80-LEN(m2))/2 SAY m2
    @ ROW+2,(80-LEN(m3))/2 SAY m3
    IF endline=4
       @ ROW+3,(80-LEN(m4))/2 SAY m4
    ENDIF
    IF &getflag
       STORE " " TO yn
       IF endline=3
          @ ROW+3,31 SAY "Choice  (Y/N)?  " GET yn;
             PICTURE "!" VALID yn$"YN"
          READ
          SET COLOR TO &old_col
          RESTORE SCREEN
          RETURN
       ELSE
          @ ROW+4,31 SAY "Choice  (Y/N)?  " GET yn;
             PICTURE "!" VALID yn$"YN"
          READ
          SET COLOR TO &old_col
          RESTORE SCREEN
          RETURN
       ENDIF
    ELSE
       xx= INKEY(2)
    ENDIF
 CASE l_msg>=73.AND.l_msg<=144
    STORE 2 TO endline
    no_rows=2            && But POSSIBLY 3 after parsing!!
    STORE LTRIM(TRIM(LEFT(msg,72))) TO m1
    IF AT(" ",m1)>0
       STORE 0 TO COUNT
       DO WHILE COUNT <=LEN(m1)
          IF SUBSTR(m1,LEN(m1)-COUNT ,1)=" "
             STORE LEFT(m1,LEN(m1)-COUNT ) TO m1
             STORE RIGHT(msg,(LEN(msg)-LEN(m1))) TO m2
             EXIT
          ENDIF
          STORE COUNT +1 TO COUNT
       ENDDO
       STORE LTRIM(TRIM(m2)) TO m2
       IF LEN(m2)>72              && Gotta add a 3rd line!
          STORE 0 TO COUNT
          STORE LEFT(m2,72) TO m2
          DO WHILE COUNT <=LEN(m2)
             IF SUBSTR(m2,LEN(m2)-COUNT ,1)=" "
                STORE LEFT(m2,LEN(m2)-COUNT ) TO m2
                STORE RIGHT(msg,LEN(msg);
                   -(LEN(m1)+LEN(m2))) TO m3
                EXIT
             ENDIF
             STORE COUNT +1 TO COUNT
          ENDDO
          STORE LTRIM(TRIM(m3)) TO m3
          STORE endline+1 TO endline
       ENDIF
    ELSE
       STORE LTRIM(TRIM(RIGHT(msg,LEN(msg);
          -LEN(m1)))) TO m2
       STORE LTRIM(TRIM(LEFT(m2,72))) TO m2
    ENDIF
    SET COLOR TO &fr_color
    IF .NOT. &getflag
       IF endline = 2
          @ ROW-1,LEFT CLEAR TO ROW+endline,RIGHT
          @ ROW-1,LEFT TO ROW+endline,RIGHT DOUBLE
          *** SHADOW ***
          SET COLOR TO &sh_col
          @ ROW+3,LEFT+1 SAY REPLICATE(CHR(223),76)
          @ ROW+2,RIGHT+1 SAY CHR(219)
          @ ROW+1,RIGHT+1 SAY CHR(219)
          @ ROW,RIGHT+1 SAY CHR(219)
          @ ROW-1,RIGHT+1 SAY CHR(220)
       ELSE
          @ ROW-1,LEFT CLEAR TO ROW+endline+1,RIGHT
          @ ROW-1,LEFT TO ROW+endline+1,RIGHT DOUBLE
          *** SHADOW ***
          SET COLOR TO &sh_col
          @ ROW+4,LEFT+1 SAY REPLICATE(CHR(223),76)
```

(continues on page 23)

The Magic of Publicity — Part 2

By David Irwin
With Emile Barrios

"I don't care what they say about me, as long as it isn't true."
— Katherine Hepburn

In the first part of this series, as you may recall, we set about debunking the Black Art of publicity.

To sum up: We talked about how some clever footwork can generate great publicity for an essentially worthless product — but how the public will sooner or later catch on — and boy will they be mad!

As Honest Abe said, you can't fool all the people all of the time, so resist the temptation to oversell your product to the press. And never forget David Irwin's First Rule of Public Relations: Don't start promoting your product until you have a product to promote. Hyping "vaporware" to the press is the express route to bankruptcy court.

Good publicity looks natural; it appears as if the press just happened to stumble upon your product and, after being suitably impressed, is writing about it in glowing terms. The effort that goes into this "natural" publicity should, at its best, be transparent. That's why the best single type of publicity is word of mouth — you just can't beat it.

And remember that those software buyers out there are smarter than you think. They want to be convinced before they spend their money. They're tired of hype and hoopla. They want to know how they'll benefit from your technology. That's what you must tell them.

But how can you make all the necessary contacts with editors, write all the press releases, keep track of all the editorial calendars, monitor the competition, and do all the other hard work that good public relations requires while continuing to develop your product?

Read on, McDuff, and I'll tell you.

The Flack Jacket

If getting the right kind of publicity sounds like a full-time job to you, you're right. And I don't say that because it's what I do for a living.

Remember we said last time that the "Magic" of publicity isn't magic at all. It is searching out and exploiting all the opportunities available to you, while putting the best possible face on your product and your company. You already have a job. You don't need another one. If you're serious about mounting a publicity campaign, you're going to need some help.

Before we go any further, let's talk a little about the difference between "Publicity" and "Public Relations."

(continues)

Publicity is getting your name in the paper as many times as possible. Public Relations is associating the right image with that name.

Publicity, as the name implies, is making sure your achievements and/or products are publicized faithfully in the trade papers and other media. Publicity is frequently nothing more than good reporting — reporting facts about you and your product to the media.

Public Relations takes those facts and dramatizes them in creative ways. Public Relations, in general, looks beyond today's facts and puts them into a larger context — as a part of the ongoing story of your company's success.

For example, let's suppose that a controversy erupts in the industry in which you're involved. You're asked to enter the fray by commenting in the press.

If Publicity is what you're after, you go for it. Damn the torpedoes, full speed ahead. Just spell the name right.

From a Public Relations standpoint, however, the situation may be quite different. As Confucius said, people who sling mud often get dirty. This may be a good time to shut up and let the whole thing blow over. Good PR is knowing when to make noise, as well as knowing when to be quiet.

That's a good reason to enlist the aid of an experienced PR professional — not your Uncle Murray who sells used cars. Not your brother-in-law Raoul who sets up "L'Eggs" displays at the supermarket. You wouldn't want those guys messing around with your code. They shouldn't mess with your image, either.

Public Relations is about mapping a path to the future. Where do you think you'll be with your product next year? In five years? What kind of markets do you want to get into? And where do you go to get publicity in those markets? You start by deciding where you want to be, then you set about developing a plan that will take you there.

Public Relations starts with those kinds of questions. Publicity, on the other hand, asks simply "what's new?" and then attacks the media with this new-found knowledge. That's where publicists got the nickname "Flack"; put enough ordnance into the air and you're bound to hit something — even if it is one of your own planes.

Massaging the Medium

There is a word that defines the success of any public relations campaign. It also defines any successful Public Relations Practitioner.

That word is CREDIBILITY.

The people who work in the various computer-related media are hard-pressed for time and resources. They rely heavily on outside sources for the information they need to fill up all those pages of text between the ads. If you can become a trusted source for quality information, you'll become a valuable commodity indeed. On the other hand, if you burn these people with bogus information, they will never believe you again. It's that simple.

So when you deal with the media, do not lie to them, either in fact or by omission. You'll find that the media as a whole has a cynical, skeptical attitude. This is because it's their job to sift the facts from the hype — and when it comes to hype, they've probably heard it all.

For this reason, it's relatively difficult for you, as a developer, to get the attention of these people. Picture it this way: An editor is sitting at his/her desk in a cramped office. The desk is piled with magazines, books, and hundreds and hundreds of discarded software diskettes. And even more discarded Press Releases that all say "Here's a product that will revolutionize the industry!" He/she opens your envelope. He/she doesn't know you or your product or your company. Your Press Release says "Here's a product that will revolutionize the industry," because that's the way you feel about the technology you've created.

The editor flings your Press Release against the wall and says: "Next!"

Your best bet is to work with someone who has an ongoing relationship with the editor or editors in question. Someone who knows what these editors are looking for. Someone with the magic word: CREDIBILITY.

I hate to repeat myself, but this is a PR practitioner's stock in trade.

Doing Your own PR

Of course, not everyone can afford to hire a professional PR person. If you're in a situation where you have to do your own PR, then are a few simple rules that will help you:

- Don't get overly extravagant in describing what your product will do or has done.

- Keep your adjectives down to a minimum. If you're going to praise your product, do it in the words of impartial users. If you're like most developers, your application is like your child — you can't really be objective in describing it. Remember, you're striving for CREDIBILITY.

- Don't send one magazine's review to another magazine. If your product has already been treated by the competition, chances are the target publication will be much less interested.

- Don't expect overnight results. It takes, on average, 90-120 days for the seeds of PR to blossom into print.

(continues)

- Don't get discouraged. If you believe in your product and its possibilities, keep plugging away at the PR process. Remember that there are a lot of other people out there trying to do exactly what you are doing.

Damage Control

Finally, let's talk briefly about a worst-case scenario.

There may come a time when the press gets hold of a story that you'd rather not see in print. Maybe your product has been found to be very buggy. Maybe you're being sued by some megacorporation. Maybe your number-one programmer has defected to Bulgaria and joined the KGB.

The press may come to you and ask if the story is true. What do you say?

The best thing you (and/or your PR person) can do is level with the media — even though it may be painful and somewhat damaging at the moment. Your honesty in the midst of crisis will be of great benefit to your relationship with the media. Your CREDIBILITY.

On the other hand, if you deny the facts, or (even worse) say "No Comment," chances are the media will get the story anyway. In that case you'll be in big trouble, because they will publish the story and take special care to make you look like a schmuck (use Ashton-Tate as an example here).

In the long run, your best interests are served by being honest with the media, as difficult as that may be.

Note: *Special thanks to Ben Irwin (who's forgotten more about Public Relations than I'll ever know) for his assistance in preparing these two columns.*

About the Author:

David Irwin is President of Irwin Ink, a microcomputer marketing and consulting company. David was the Technical Editor of ***Data Based Advisor*** *and President and Chief Executive Officer of Data Based Solutions, the parent company of* ***Data Based Advisor*** *and a dBASE-language software publisher. David now writes for* ***DBMS Views*** *(a marketing publication covering the dBASE Community) and* ***IDBUG Journal***.

EOF

Deduping

By Pat Adams

One of my oft-told stories concerns a young colleague who came to see me one day and excitedly showed me a routine he had written to cull duplicate records from a database. While it was an interesting bit of code, I had to explain to him that good systems design stops duplicate records from entering the system in the first place. If a system is designed properly, there should be no need for routines that later cull for duplicates. However, if the number of telephone calls I receive from people looking for such routines, and the number of such routines I see posted on BBSs are any indication, there are lots of xbase systems out there which do not avoid duplicate record creation up front.

While the philosophy of deduping is a simple one, it is not always one that is easy to execute. Certainly the easiest implementation is with systems where each record has a unique identifier. In such instances, duplication can be avoided by doing searches on the unique key or field prior to an APPEND. There is another aspect to this that often trips up inexperienced programmers — doing duplicate searches if the key field is changed during an EDIT.

The following code illustrates up front deduping for a patient database that uses the patient's Social Security Number as the unique identifier. Notice that the new FoxPro SCATTER MEMVAR BLANK and GATHER MEMVAR syntax is used to create blank memory variables; then, if no duplicate is found, this is used to replace the data in the newly APPENDed record. If a duplicate record is found during the APPEND routine, the KEYBOARD command is used to stuff a PgDn into the keyboard. A logical .T. is returned from the VALID and the PgDn then takes the routine out of the READ.

If the user is EDITing a record and changes the Social Security Number to one that already exists in the database, he/she is so advised and the Social Security Number is returned to the original value. For both the APPEND and EDIT sequences, all activity is performed against memory variables, so data are not input to the database unless the duplicate screening process is concluded satisfactorily (yet another example of why you might want to do data entry and editing against memvars rather than directly against the database).

```
*********************************************
        PROCEDURE Padd
*********************************************
*
*  Add patient record to database.
*   Duplicate search is performed on unique
*   key of patient's social security number.
*  If duplicate record found APPEND is aborted.
*
```

(continues)

```
*  Author: Pat Adams, DB Unlimited
*
* * * * * *
*
SET TALK OFF
SET CONFIRM ON
SET DELETED OFF
SELECT 2
USE ZIP INDEX ZIP
SELECT 1
USE PATIENT INDEX SSNO
* ------------------------
* Create blank memory variables with
* the same name as the fields in the
* PATIENT.DBF file
* ------------------------
SCATTER TO MEMVAR BLANK     && <-FoxPro command
SET COLOR TO W/n
CLEAR
SET COLOR TO W+/B
@ 6,12,16,70 BOX"╔═╗║╝═╚║ "
@ 7,14 SAY "SOCIAL SECURITY #"
@ 9,31 SAY "."
@12,14 SAY "ADDRESS:"
@14,58 SAY ","
SET COLOR TO W/B, N/W
@10,15 SAY "First Name"+SPACE(11)+"Last Name"
@ 7,32 GET M->SSNO PICTURE [999-99-9999] VALID;
   Dupcheck(ssno, "ADDING")
@ 9,14 GET M->FIRST
@ 9,30 GET M->MI PICTURE [!A]
@ 9,33 GET M->LAST
@12,23 GET M->ADDRES1
@13,23 GET M->ADDRES2
@14,23 GET M->CITY
@14,60 GET M->STATE PICTURE [!A]
@14,64 GET M->ZIP PICTURE [99999] VALID ;
   Zipcheck(city, state, zip)
READ

IF READKEY() - 12 .OR. READKEY() - 268
* ----------------------------------------
* Return to main menu if ESC pressed during READ
* ----------------------------------------
    RETURN TO MASTER
ENDIF readkey() - 12, etc.

IF LEN(TRIM(CITY)) <> 0
* ------------------------
* If no data in CITY field the routine
* has been aborted.  However, if data exists,
* add new record to database and enter info
* from the memory variables.
* ------------------------
   APPEND BLANK
   GATHER MEMVAR        && <-FoxPro command
   * ------------------------
   * GATHER MEMVAR is a FoxPro command.
   * If using FoxBASE+ a REPLACE must be done.
   * ------------------------
ENDIF len(trim(city)) <> 0
RETURN
* END Procedure Padd

****************************************
PROCEDURE Dupcheck
****************************************
*$ Check PATIENT.DBF file for duplicate
*  Social Security number.  This routine is
*  used by APPEND and EDIT routines.
*
*  The passed parameters are:
*     xssno - Social Security number
*       xaction - Either ADDING or EDITING will
*                 be passed to determine action
* * *
*
PARAMETERS xssno, xaction
CLEAR TYPEAHEAD
IF LEN(TRIM(xssno)) - 0
* ----------------------------
*$ SS # required.  If user had not
*  entered the data pop up warning.
* ----------------------------
    SET COLOR TO GR+*/R
    @ 5,45,10,66 BOX"╔═╗║╝═╚║ "
    SET COLOR TO W+/R
    @ 6,46 CLEAR TO  9,65
    @ 6,47 SAY "You MUST enter a"
    @ 7,47 SAY "Social Security #!"
    SET COLOR TO W/R
    @ 9,47 SAY "Press any key..."
    WAIT ""
    RETURN .F.
ENDIF len(trim(xssno)) - 0

IF xaction - "ADDING"
* --------------------------
* In process of adding new record.
* Do simple check for duplicate SS No.
* --------------------------
   SEEK xssno

   IF FOUND()
   * ----------------
   *$ Advise user if duplicate SS # found
   * ----------------
      DO Dupfind1
      KEYBOARD CHR(3)
   ENDIF found()

   RETURN .T.
ELSE
* --------------------------
*$ User is editing data.  Check
*  to see if SS # has been changed
*  in the memvar differs from that
*  in the database field.  If so,
*  search for possible duplicate.
*  EDIT routine has already stored the
*  patient RECNO() to a memvar at the
*  calling level.  Memvar name is kurrent.
* --------------------------
   IF M->ssno # patient->ssno
      SEEK xssno

      IF FOUND()
         DO Dupcheck2
         GO kurrent
         * ------------------
         * NOTE: This code to return the
         * M->SSNO memvar to its original
         * information will only work in FoxPro.
         * KEYBOARD must be used in FoxBASE+
         * ------------------
         STORE patient->ssno TO M->ssno
         RETURN .F.
      ELSE
         RETURN .T.
      ENDIF found()
   ELSE
   * ------------------------
   * SS # has not been changed so
   * return a logical .T.
   * ------------------------
      RETURN .T.
   ENDIF M->ssno # patient->ssno
ENDIF xaction
* END Procedure Dupcheck


********************************
PROCEDURE Dupfind1
********************************
*$ Pop up warning box to tell user
*  SS # already exists
* * * * * * *
*
PRIVATE savescrn
```

(continues)

```
SAVE SCREEN TO savescrn
SET COLOR TO GR+/R
?? CHR(7)
@ 7,47,16,63 BOX "▄▄▄█▀▀▀█ "
* ---------------------------
* After screen has been saved
* paint message box
* ---------------------------
SET COLOR TO W+/R
@ 8,48 CLEAR TO 15,62
@10,49 SAY "That Social"
@11,49 SAY "Security #"
@12,49 SAY "is already in"
@13,49 SAY "the database."
@15,49 SAY "Press any key"
SET COLOR TO GR+*/R
@ 8,53 CLEAR TO  8,59
@ 8,53 SAY "SORRY!"
WAIT ""
RESTORE SCREEN FROM savescrn
RETURN
* End Procedure Dupfind1

*************************************
PROCEDURE Dupfind2
*************************************
*& Advise user s/he has changed the SS #
*  to one that already exists in the database
* * * * * * *
*
PRIVATE savescrn
SAVE SCREEN TO savescrn
SET COLOR TO GR+*/R
@ 8,40,21,62 BOX "▄▄▄█▀▀▀█ "
SET COLOR TO W+/R
* ----------------------------
* After screen has been saved,
* paint message box
* ----------------------------
@ 9,41 CLEAR TO 20,61
@11,42 SAY "A record already"
@12,42 SAY "exists with the new"
@13,42 SAY "SS # you entered."
@14,42 SAY "Duplicate are not"
@15,42 SAY "permitted."
@17,42 SAY "The original SS #"
@18,42 SAY "will be restored."
@20,42 SAY "Press any key..."
SET COLOR TO GR+*/R
@ 9,49 CLEAR TO  9,54
@ 9,49 SAY "SORRY!"
WAIT ""
RESTORE SCREEN FROM savescrn
RETURN
* END Procedure Dupfind2
```

The above routine is fairly simplistic and relies on the presence of a unique identifier for the duplication avoidance check. However, life is not often so simple. All too frequently there will be no unique identifier. An example might be a variation on our PATIENT database where a Social Security Number is not used and the system assigns a patient ID number at the time the new patient record is entered; other examples are the systems I designed for the International Dbase Users Group membership and the NYPC Consultants SIG membership. All three of these systems contain addresses and, at first light, it might be thought the addresses could be used as part of a duplication avoidance check. However, once again reality rears its ugly head.

Patients — just like other people — move from time to time and might not be seen for months or years between visits. Therefore, a deduping process that utilizes the address may not always catch duplicate records. The same situation applies for the IDBUG Membership System and the Consultants SIG Membership System, but with yet one more consideration — inactive membership files must be searched as well as active membership files. Another problem is that these systems may very well have three patients with the same name or several members with the same name. About the only alternative this leaves is to search by the name of the individual, and let the user compare any matches to determine, based upon additional information, whether the matching record is the same person or not.

Searching by last name and first name often becomes useless, since a user may input a first name as Robert one time and Bob another time — and may yet come up with other variations on this theme. Searching by last name creates other problems, not the least of which is the constant misspelling of last names. What we really need here is a good phonetic fuzzy search technology that can adequately deal with misspellings of names (SOUNDEX( ) is simply not up to the task). Proximity Technology offers a Developer's Version of their Friendly Finder that can be used for this purpose, but it is relatively slow, very expensive, requires royalty payments, and has a counter type of copy protection. Therefore, I have foregone the use of Proximity's product except in one instance where the client's need left no alternative. I've been advising Korenthal Associates in Manhattan (the creators of the excellent 4PRINT utility to print multiple pages on one sheet with a laser printer) on their development of a fuzzy search technology currently code named PhDbase, and hope this will be on the market soon. But until PhDbase arrives, I'll have to continue to use the routines I've developed over the years to contend with these problems.

The logic I use is applicable primarily in searching and duplication avoidance where a person's name is the key. It is effective in systems where the database will remain relatively small — under 5,000 or 6,000 records. Otherwise, it's necessary to resort to the SIMILAR( ) UDF discussed in this publication some months ago. For the small database situations mentioned, the primary search utilizes a SEEK based upon the first three or four letters of the last name input by the user. If a match is found, the user is informed via a pop-up box that contains the full name from the record as well as address information. The user is asked if this is the same person and, if not, the program iterates the search and pop-up routine until all possible matches have been exhausted. If the user indicates that a match is the same person, the APPEND routine is aborted, of course. Obviously, this approach is not suitable for very large databases.

(continues)

```
*!*******************************************************
*!
*!      Procedure: MADD
*!
*!      Called by: MMENU          (procedure in IDBUG.PRG)
*!
*!          Calls: SAME           (procedure in IDBUG.PRG)
*!               : ZIPCHECK       (procedure in IDBUG.PRG)
*!
*!           Uses: ACTIVE.DBF         Alias: AC
*!               : INACTIVE.DBF       Alias: IN
*!               : ZIPDATA.DBF        Alias: ZIP
*!               : TEMP.DBF
*!
*!        Indexes: ANAME.IDX
*!               : A_IDNO.IDX
*!               : AZIP.IDX
*!               : I_IDNO.IDX
*!               : ZIP.IDX
*!
*!   Memory Files: MEMIDNO.MEM
*!
*!*********************************
PROCEDURE madd
**********************************
*
*
*& Add new IDBUG member
*
*  Routine includes a duplicate avoidance
*  search keyed by the last name.
*
*  Written by: Pat Adams
*
* * * * * * * * * * * * * * * * * * * *
*
SET ESCAPE ON
ON ESCAPE RETURN TO MASTER
SET DELETED OFF
SET SAFETY OFF
SET BELL Off
SET CONFIRM ON
SET TALK Off
PUBLIC mfirst, mmi, mlast, same, match
PRIVATE okay, more, last_no, good, savescrn, wwait
STORE " " TO same, okay
SET FUNCTION 7 TO CHR(29)
SET FUNCTION 8 TO CHR(21)
SET FUNCTION 9 TO CHR(23)
STORE " " TO okay, more
* -----------------------------
*$  Open databases for use
* -----------------------------
SELECT 1
USE ACTIVE INDEX aname, a_idno, azip ALIAS ac
SELECT 2
USE inactive INDEX i_idno ALIAS in
SELECT 3
USE zipdata INDEX zip ALIAS zip

DO WHILE UPPER(more) # "N"
   * -----------------------
   *$ Continue adding new members
   *  until user indicates otherwise
   * -----------------------
   DO WHILE UPPER(okay) # "Y"
      * -----------------------------
      *$ Get user input for new member name
      * -----------------------------
      SET COLOR TO W/N
      CLEAR
      @ 0,1 SAY "ESC to"
      @ 1,1 SAY "return to"
      @ 2,1 SAY "Main Menu"
      STORE SPACE(16) TO mfirst
      STORE SPACE(1) TO mmi
      STORE SPACE(24) TO mlast
      SET COLOR TO +BG/N
      @ 10,20 SAY "PLEASE ENTER THE NAME OF THE NEW MEMBER"
      @ 13,36 SAY "."
      SET COLOR TO +BR/BG
      @ 00,27,02,52 BOX"┌┐│┘─└│"
      SET COLOR TO +W/BG
      @ 01,28,01,51 BOX "             "
      @ 01,29 SAY "ADD A NEW IDBUG MEMBER"
      SET COLOR TO W/N
      @ 14,21 SAY "(First)                  (Last)"
      SET COLOR TO +GR/N, +W/BR
      ?? SYS(2002,1)
      @ 13,17 GET mfirst
      @ 13,35 GET mmi PICTURE "@A!"
      @ 13,38 GET mlast

      IF READKEY() = 12 .OR. READKEY() = 286
         RETURN TO MASTER
      ENDIF readkey() = 12, etc.

      SAVE SCREEN TO savescrn
      ?? SYS(2002)
      * -----------------------
      *$ User validation of input
      * -----------------------
      SET COLOR TO +B/B
      @ 10,05,16,38 BOX "▄▄█▀▀▀█ "
      SET COLOR TO +W/B
      @ 11,06,15,37 BOX "         "
      @ 11,07 SAY "You entered:"
      qquery = IIF(mmi = " ", TRIM(mfirst) + " " +;
        TRIM(mlast), TRIM(mfirst) + " " +;
        mmi + ". " + TRIM(mlast))
      @ 13,07 SAY qquery
      SET COLOR TO *+W/B
      @ 15,07 SAY "IS THIS CORRECT? (Y/N)"
      wwait = INKEY(0)

      DO WHILE .NOT. CHR(wwait) $ "YyNn"
         * --------------------------------
         *  Error trapping to ensure Y or N response
         * --------------------------------
         IF wwait = 27
            RETURN TO MASTER
         ELSE
            wwait = INKEY(0)
         ENDIF wwait = 27
      ENDDO while .not. chr(wwait) $

      okay = UPPER(CHR(wwait))

      IF okay = "N"
         * --------------------------
         *$ Loop to permit re-entry of
         *  member name if error made
         * --------------------------
         LOOP
      ELSE
         RESTORE SCREEN FROM savescrn
      ENDIF okey = "N"
   ENDDO while upper(okay) # "Y"

   SET COLOR TO +GR/N
   @ 22,16 SAY " " + CHR(16) + " " + CHR(16) +;
      " " + CHR(16) + "  SEARCHING FOR POSSIBLE DUPLICATE  " +;
      CHR(17) + " " + CHR(17) + " " + CHR(17)
   * ----------------------------------------------
   *$ First search ACTIVE database for possible duplicate
   * ----------------------------------------------
   SELECT ac
   GO TOP
   match = UPPER(mlast)
   SEEK match

   IF FOUND()
      * -----------------------
      *$  If possible duplicate found
      *   in ACTIVE database, inform user
      * -----------------------
      DO same
   ENDIF found()
```

(continues)

```
IF UPPER(same) # "Y"
   * ---------------------------
   *$ If no duplicate found in ACTIVE
   *  database, search INACTIVE
   * ---------------------------
   SELECT in
   GO TOP
   SEEK match

   IF FOUND()
      * ---------------------------
      *$  If possible duplicate found
      *   in INACTIVE database, inform user
      * ---------------------------
      STORE " " TO same
      SAVE SCREEN TO savescrn

      DO WHILE UPPER(in->last) = match
         DO same

         IF UPPER(same) = "Y"
            * ---------------------------
            *$ Move inactive record to ACTIVE database
            * ---------------------------
            SET COLOR TO *+B/W
            @ 05,37,10,68 BOX "▄▄█▀▀█▄█ "
            SET COLOR TO B/W
            @ 06,38,09,67 BOX "         "
            @ 06,42 SAY "MOVING RECORD TO ACTIVE"
            @ 07,44 SAY "MEMBERSHIP DATABASE"
            @ 09,39 SAY "Sorry to keep you waiting..."
            COPY TO temp
            DELETE
            PACK
            SELECT ac
            APPEND FROM temp
            * ---------------------------
            *$ Present record for user editing
            * ---------------------------
            ** NOTE: Code for editing goes here
            EXIT
         ELSE
            SKIP
         ENDIF upper(same), etc.
      ENDDO while upper(in->, etc.
   ENDIF found()

** Code to APPEND the record follows here


*!*************************************************
*!
*!      Procedure: SAME
*!
*!      Called by: MADD     (procedure in IDBUG.PRG)
*!
*!*************************************************
PROCEDURE same
**************************************************

*   The SAME procedure is called when a
*   possible matching record is found in
*   in the ACTIVE or INACTIVE databases.
*
*   Assumes a PUBLIC memvar named SAME has
*   been declared at the calling level.
*
* * * * * * * * * * * * * * * * * * * * *
*& Show data on possible matching record
*
PRIVATE savescrn, wwait
SET ESCAPE ON
ON ESCAPE RETURN TO MASTER
STORE " " TO same
SAVE SCREEN TO savescrn

DO WHILE UPPER(LEFT(LAST,4) = LEFT(match,4)
   SET COLOR TO +B/B
   @ 02,05,11,48 BOX   "▄▄█▀▀█▄█ "
   SET COLOR TO +W/B
   @ 03,06,10,47 BOX "         "
   @ 03,08 SAY "The following record exists "
   @ 6,8 SAY TRIM(first) + " " + TRIM(LAST)

   IF hstreet1 # SPACE(35)
      * ---------------------------
      *  Display partial info on home
      *  address if it is in record
      * ---------------------------
      @ 07,08 SAY TRIM(hstreet1)
      @ 08,08 SAY TRIM(hcity) + ", " + hstate +;
         TRANSFORM(hzip, "@R 99999-9999")
   ELSE
      @ 7,8 SAY TRIM(ostreet1)
      @ 8,8 SAY TRIM(ocity) + ", " + ostate + ;
         TRANSFORM(ozip, "@R 99999-9999")
   ENDIF hstreet # space(35)

   SET COLOR TO *+W/B
   @ 10,08 SAY "IS THIS THE SAME PERSON? (y/n)"
   wwait = INKEY(0)

   DO WHILE .NOT. CHR(wwait) $ "YyNn"
      * ---------------------------------------
      *  Error trapping to ensure Y or N response
      * ---------------------------------------
      wwait = INKEY()

      IF wwait = 27
         RETURN TO MASTER
      ENDIF wwait = 27
   ENDDO while .not. chr, etc.

   same = UPPER(CHR(wwait))

   IF same = "N"
      SKIP
   ELSE
      EXIT
   ENDIF same = "N"
ENDDO

RETURN
* END Procedure SAME
```

As you can see, there is no one method of deduping that is right for all situations or all systems. It is necessary to analyze each system and each database in the system to ascertain what fields are key to avoidance record duplication. Once that determination has been made, it's possible to design the dedupe routine(s). As shown in the above example, it may not always be possible for the computer to deal with things effectively without additional analysis and input from the user. Where data input is achieved via electronic transfer from tape, telecommunications, or mainframe transfer, entirely different methodologies will be required. The important principle is to prevent duplicate records from entering the system.

About the Author:

Pat Adams is an independent consultant, headquartered in Brooklyn, New York. She is the author of two books on dBASE III and dBASE III+, and numerous articles on the family of dbase languages. She is also founding treasurer of the International dBASE Users Group, and founder and chair of the NYPC Consultants SIG. **EOF**

A Recap of the Fox Developers Conference

By Glenn A. Hart

The Fox Developers Conference is history. This rather cliched phrase isn't, for once, hyperbole. For there really *was* history made at the Conference — on several levels.

There were two stars of the show. FoxPro literally blew away the audience at the jam-packed opening session. Dr. David Fulton's FoxPro demonstration was interrupted constantly by appreciative clapping, cheering, and whistling. The gathering of developers, corporate MIS executives, reporters, and users reached a fever pitch by the end of the two-hour, in-depth demo. Feature after feature elicited oohs and aahs, as the assemblage began to grasp the incredible power which would soon be available to them.

The other star was Dr. Dave himself. While he attends some trade shows and an occasional user group meeting, for the most part Dave stays anchored in bucolic Perrysburg, which is hardly a thriving media center. Recent publicity in leading database journals has widened his exposure, but this was the first opportunity for many to see the Fox guru in action. Dave projects a quiet brilliance and obvious intellect, but his gentle and dry wit may have been something of a surprise to some.

While the demo was intended as a continuous affair, with several feedback sessions scheduled throughout the Conference for questions and answers, several of the attendees couldn't restrain themselves. Questions peppered the demo, and were handled pungently by Dr. Fulton. Strangely enough, it was this aspect of the session that most won over the developers. For example, a member of the audience suggested an interesting and clever approach to handling one facet of FoxPro debugging. When Dave tried it, with excellent results, his whoop of delight conveyed more than any planned demo could. The developers and programmers instantly realized that Dave is one of *them*. Not some cold corporate executive motivated by bottom lines, head counts, stock prices, or other irrelevances, but a *user* who understands and shares the ineffable pleasures of superior software.

Dr. Fulton also announced the pricing for FoxPro. Single user FoxPro is priced at $795.00, with multi-user FoxPro/LAN at $1,095.00. The FoxPro runtime is $500.00. The runtime automatically configures itself to handle single-user or multi-user applications, so there's no longer a separate version for each base package. Registered FoxBASE+ and FoxBASE+/386 users can purchase FoxPro for only $195.00, and FoxBASE+/LAN owners can buy FoxPro/LAN for $250.00. These special prices were announced as effective through December 31, 1989.

The Conference readily answered the question of whether anyone would come to Toledo, Ohio for *anything*. Pinnacle Publishing, co-sponsor of the event (and publisher of your favorite Fox-oriented publication), cut off attendance at 650, and turned away another 250. Given the early cutoff announcement, it's likely that many, many more would have attended if possible. Toledo proved a pleasant surprise for many attendees. The Conference hotels were lovely and professional, the convention facilities were attractive and competently run, and the surrounding areas of downtown Toledo were pretty, clean, and safe. If Toledo does close down a mite early for programmer types with all-night work habits, no city can have everything!

The Conference kicked off Tuesday night with a lavish welcoming reception. Mounds of seafood and other delectables proved conclusively that there *is* fine food available between New York and Chicago. Associating faces with names and disembodied voices was the order of the evening, as readers met writers, users met Fox tech support and development folks, and denizens of the electronic mailwaves on CompuServe and IDBUG met each other.

Following the Wednesday morning FoxPro rollout, at lunch hour David Irwin gave an entertaining slide show chronicling the triumphs and follies of the dbase community since its inception nearly 10 years ago. David's engaging manner and the intrinsic humor of some of the silly marketing ploys attempted over the years were most amusing.

The intensive seminar schedule at the Conference covered a wide gamut of FoxPro, FoxBASE+, and FoxBASE+/Mac topics. Most attendees reported that their toughest decision was which seminars to choose from the long list, even though several were repeated more than once. The FoxBASE luminaries conducting seminars included Pat Adams, Luis Castro, Jim Davis of SBT, Bill French, George Goley, Tom Gottheimer of SBT, Richard Grossman, John Hawkins, Jon Henderson of 3D Graphics, Stephen Hochschild of Novell, David Irwin, Walt Kennamer, Jordan Powell, Tom Rettig, Randy Wallin, Jerry Whittaker (see, writing for **fox**talk *can* make you a star!), yours truly, and others.

Later that day, a four-hour vendor trade show gave attendees a chance to examine the wares of leading third-party software vendors like SBT, Korenthal Associates, Clear Software, Concentric Software, SCO, Wallsoft, and others.

Wednesday evening, consultant/columnist Pat Adams hosted an exclusive IDBUG party. Most of the speakers, Dave and Amy Fulton, Dick LaValley, Fox Chairman Richard Ney and other Foxies, as well as other leading lights made a heavy dent in the headquarters hotel's stock of fine beverages (including The Glenlivet, perhaps the best Scotch in the world, and a brand I helped introduce into this country in the early seventies) until the wee hours. A most pleasant end to a *very* long day!

(continues)

Thursday was devoted primarily to seminars. The highlight of the day was a luncheon speech by the father of the dbase community, Wayne Ratliff. Wayne shared some humorous stories on the origination and early marketing of dBASE II. This was the first time many had seen and met the man to whom we all owe so much, and Wayne's quiet charm was most ingratiating.

Thursday evening Fox and Pinnacle hosted an elegant cocktail party at the Toledo Museum of Art. The Museum is widely regarded as one of the best in the country, and provided a cosmopolitan backdrop for the festivities. A special exhibition was displayed for the Conference, and a sophisticated jazz trio generated suitably refined aural accompaniment.

Friday morning, Fox opened the gates for everyone to pick up their copy of FoxPro (or FoxBASE+/Mac if they preferred). With over 600 advanced power users receiving copies of FoxPro, Fox decided to delay the commercial release of FoxPro to provide the group with time to use the program exhaustively and report any remaining problems. This wise decision should result in an even more solid program when FoxPro hits the shelves sometime in early October. The multi-user FoxPro/LAN should follow about a month later.

Throughout the Conference, computer rooms stocked with PCs and Macs were available to all attendees to experiment with FoxPro and FoxBASE+/Mac, share code and otherwise assuage their computer withdrawal symptoms. Fox gathered all comments and problem reports, and Dave Fulton got a cheer on Friday when he announced that all problems reported by eight PM the previous evening had already been fixed. No wonder many of the Fox programmers weren't much in evidence during parts of the Conference!

The Conference closed with Dick LaValley updating the group on the status of the litigation between Fox Software and Ashton-Tate. At present, the discovery process, where both sides gather information, continues. The suit is not likely to come to trial, if it ever does, until late 1990 or early 1991. The legal actions, while somewhat costly, are having essentially no impact on Fox's ongoing business, other than the publicity bonanza the suit bestowed on Fox.

Dr. Fulton then discussed the future directions Fox intends to explore. Dave explained that Fox no longer conceptualizes a distinction between PC and Mac products. Rather, Fox categorizes its products as graphics mode products and character mode products. Next on the Fox agenda is synchronizing these two environments. Fox can then consider porting both types of software to other platforms. Among the possibilities: character mode OS/2 on the upcoming '386 version, graphics mode Presentation Manager, character mode Unix/Xenix, the various graphics interfaces operating over Unix, minicomputers, etc.

FoxServer and various SQL options were also covered. Dave drew interesting distinctions between performance oriented servers, likening them to math coprocessors, and connectivity oriented servers for retrieval of information from locations remote from the local workstation. Fox intends to support both server styles and a variety of implementations.

After a spirited question and answer session, Dave received a standing ovation and the Conference drew to a close. Weary yet elated, everyone rushed home to put FoxPro through its paces and enter the next generation of database management.

The reports, reviewers, and columnists attending the Conference will be sharing their thoughts with you in the months to come. Even during the Conference, though, a consensus seemed to be emerging, one with which I concur.

The importance of the Conference transcends on two scores the mere introduction of a new product. FoxPro ups the ante for *all* personal computer software, not just database managers. New interface styles, new ease of learning and ease of use, new functionality, seamless integration — and all at unprecedented speeds. Other PC software seems clumsy and sluggish. Over time, users will increasingly demand FoxPro-like interface elegance and execution speed.

Perhaps even more striking was the almost palpable sense of a company crossing a threshold, somehow coming of age before our eyes. Fox's potential is nearly unlimited. Several factors point toward mercuric growth. Fox's products are technologically superior in nearly every way to its competition. Ashton-Tate, its primary competitor, has hideously flawed products, serious management weaknesses, and financial difficulties. And, not to be underestimated, Fox is staffed with committed and *nice* people who love what they do and what they create.

Fox faces serious challenges ahead. They must consolidate and expand their technological leadership, enter new marketplaces — with many technical and marketing pitfalls to avoid, expand their organization greatly — with all the administrative responsibilities this entails, while at the same time retaining the special personal and corporate style that makes them so special. I, and the other analysts at the Conference, have little doubt that Fox can accomplish all these objectives. We all left the Conference with reinforced confirmation of the wisdom of our decision to commit to Fox products.

The Fox Developers Conference was both an end and a beginning. It marked the culmination of two years work on FoxPro. At the same time, it represented the beginning of a new phase for Fox Software. It was a remarkable event. I hope I've been able to give you a feel for what happened, but I know this is but a pale shadow of the reality of this powerful experience. Don't miss the next one.

EOF

Faster Search Program

By Herman Rohr

"If only we had a list of all the criminals in Gotham City who have red hair, were age 40-45, have a height 6' 5," wore clown outfits, and drove a 20' long car that was shaped like a banana," Alfred says to Batman.

A worried looking Batman suddenly exclaims, "I know, let's check the Bat Computer." (Like he forgot he had this computer that occupied three-fourths of the Bat Cave.)

After inputing all the search criteria into the computer, seconds later, sure enough — the culprit was identified. Now THAT was a fast search mode.

Many of us might have somewhat similar needs in a search program. The problem occurs when combining all of those variables together. Depending on the size of your database, this process could take several minutes.

I discovered a way to find all clown-dressed drivers of banana cars almost instantly. The trick is to have a separate index on each of the input fields of search criteria.

General Layout

Step 1	Collect search criteria.
Step 2	Use your database and index off of the first used field in the search criteria.
Step 3	Create a macro that can change with the amount of fields that are used in the search criteria.
Result	Data output spews forth from printer! (You may need a rag if the spewing gets out of hand.)

Spock Speed

There is also a lot of logic involved in this process. So put on your Spock ears and let's delve into the what's what.

Speed in processing your search with this program comes from knowing which fields to use and which not to use.

After determining which fields to use, set a logical variable to indicate that field is being used. Then it just becomes a matter of listing all the possible combinations of the fields to adjust the macro.

So instead of having the computer try to LOCATE for a combination of fields, you actually do a FIND on the first used field and then use the macro you've set up to search for the remaining criteria.

Wild Card Search

The program also monitors the length of the input variable and does the search off of it. For instance, if you were to enter an "S" in the first field, all records with that field beginning with an "S" will be listed.

Of course, if there is no match available, the program notifies you of this and allows you to modify your entry.

What follows is a program that my department uses. It has cut our search time for a record from what was up to several minutes, down to few seconds.

```
*:**********************************************************
*:
*:         Program: SEARCH.PRG
*:
*:          Author: HERMAN ROHR
*:        Copyright (c) 1989, HERMAN ROHR
*:   Last modified: 08/29/89     12:13
*:
*:            Uses: BIOMSTR.DBF
*:
*:         Indexes: DESCMSTR.NDX
*:                : BIOMFG.NDX
*:                : BIOMDL.NDX
*:                : DEPTMSTR.NDX
*:
*:      Documented: 08/29/89 at 12:14     FoxDoc version 1.0
*:**********************************************************

*:**********************************************************
*: Structure for database: C:\FB\BIOMSTR.DBF
*: Number of data records:     9689
*: Date of last update   : 09/22/89
*: Field  Field Name  Type       Width   Dec
*:     1  ID          Character      6
*:     2  MFG         Character     20
*:     3  SN          Character     10
*:     4  DESCR       Character     25
*:     5  MODEL       Character     20
*:     6  CTR         Character      4
*:     7  PM_NO       Character      3
*:     8  LAST_PM     Date           8
*:     9  STATUS      Character      1
*:    10  LAST_SFTY   Date           8
*: ** Total **                    106
*:**********************************************************

*:**********************************************************
*: Select area:  1, Database in Use: C:\FB\BIOMSTR.DBF
*: Master index file:   C:\FB\DESCMSTR.IDX  Key: DESCR+ID
*:        Index file:   C:\FB\DEPTMSTR.IDX  Key: CTR+ID
*:        Index file:   C:\FB\BIOMDL.IDX  Key: MODEL+ID
*:        Index file:   C:\FB\BIOMFG.IDX  Key: MFG+ID
*:**********************************************************


SET TALK OFF
SET ESCAPE ON
SET ECHO OFF
SET EXACT OFF
CLEAR

* ---------------------------------------
*$ Initialize memory variables
* ---------------------------------------

STORE SPACE(25) TO tdesc
STORE SPACE(4) TO tctr
STORE SPACE(20) TO tpm
STORE SPACE(20) TO tmfg,tmdl
STORE 0 TO af,bf,cf,df,so
STORE 1 TO x,s,y,z
STORE .F. TO a,b,c,d
```

(continues)

```
DO WHILE .T.

   * -----------------------------------------
   *$ Set up screen and do GETS
   * -----------------------------------------

   @  4,4 SAY "INPUT SEARCH CRITERIA  :"
   @  8,4 SAY "DESCRIPTION............" GET tdesc
   @ 10,4 SAY "MANUFACTURER..........." GET tmfg
   @ 12,4 SAY "MODEL NUMBER..........." GET tmdl
   @ 14,4 SAY "COST CENTER............" GET tctr ;
      PICTURE "9999"

   READ

   * -----------------------------------------
   *$ If no entry, safely returns to
   *  prior program
   * -----------------------------------------

   IF tdesc = " " .AND. tmfg = " " .AND. tmdl = " " .AND. ;
      tctr = " "
      RETURN
   ENDIF

   * -----------------------------------------
   *$ USE database, set up indexes
   * -----------------------------------------

   USE biomstr
   SET INDEX TO descmstr, biomfg, biomdl, deptmstr

   * -----------------------------------------
   *$ Looks for blank entries or mismatches.
   *  Loops back if no match. Stores recno()
   *  to z if there is a match.
   * -----------------------------------------

   IF tdesc > " "
      tdesc = TRIM(LTRIM(tdesc))
      SET ORDER TO 1
      FIND &tdesc
      IF EOF()
         @ 16,4 SAY "NO MATCH ON DESCRIPTION"
         WAIT
         @ 16,0 TO 20,50 CLEAR
         STORE SPACE(25) TO tdesc
         LOOP
      ENDIF
      STORE RECNO() TO af
      IF .NOT. EOF()
         STORE .T. TO a
      ENDIF
   ENDIF

   IF tmfg > " "
      tmfg = TRIM(LTRIM(tmfg))
      SET ORDER TO 2
      FIND &tmfg
      IF EOF()
         @ 17,4 SAY "NO MATCH ON MANUFACTURER"
         WAIT
         @ 16,0 TO 20,60 CLEAR
         STORE SPACE(20) TO tmfg
         LOOP
      ENDIF
      STORE RECNO() TO bf
      IF .NOT. EOF()
         STORE .T. TO b
      ENDIF
   ENDIF

   IF tmdl > " "
      tmdl = TRIM(LTRIM(tmdl))
      SET ORDER TO 3
      FIND &tmdl
      IF EOF()
         @ 18,4 SAY "NO MATCH ON MODEL"
         WAIT
         @ 16,0 TO 20,60 CLEAR
         STORE SPACE(20) TO tmdl
         LOOP
      ENDIF
      STORE RECNO() TO cf
      IF .NOT. EOF()
         STORE .T. TO c
      ENDIF
   ENDIF

   IF tctr > " "
      tctr = TRIM(LTRIM(tctr))
      SET ORDER TO 4
      FIND &tctr
      IF EOF()
         @ 19,4 SAY "NO MATCH ON COST CENTER"
         WAIT
         @ 16,0 TO 20,60 CLEAR
         STORE SPACE(4) TO tctr
         LOOP
      ENDIF
      STORE RECNO() TO df
      IF .NOT. EOF()
         STORE .T. TO d
      ENDIF
   ENDIF

   * -----------------------------------------
   *$ After determining which fields are to
   *  be used in the search, the logic state-
   *  ments of a,b,c & d will help set up
   *  a macro used in the search mode.
   * -----------------------------------------

   IF a .AND. b .AND. c .AND. d
      uto = "DESCR = TDESC .AND. MFG = TMFG .AND. " + ;
          "MODEL = TMDL .AND.  CTR = TCTR"
   ENDIF

   IF b .AND. c .AND. d .AND. .NOT. a
      uto = "MFG = TMFG .AND. MODEL = TMDL .AND. " + ;
          "CTR = TCTR"
   ENDIF

   IF c .AND. d .AND. .NOT. a .AND. .NOT. b
      uto = "CTR = TCTR .AND. MODEL = TMDL"
   ENDIF

   IF d .AND .NOT. a .AND. .NOT. b .AND. .NOT. c
      uto = "CTR = TCTR"
   ENDIF

   IF a .AND. b .AND. c .AND. .NOT. d
      uto = "DESCR = TDESC .AND. MFG = TMFG .AND. " + ;
          "MODEL = TMDL"
   ENDIF

   IF b .AND. c .AND. .NOT. a .AND. .NOT. d
      uto = "MFG = TMFG .AND. MODEL = TMDL"
   ENDIF

   IF c .AND. .NOT. a .AND. .NOT. b .AND. .NOT. d
      uto = "MODEL = TMDL"
   ENDIF

   IF a .AND. b .AND. .NOT. c .AND. .NOT. d
      uto = "DESCR = TDESC .AND. MFG = TMFG"
   ENDIF

   IF b .AND. .NOT. a .AND. .NOT. c .AND .NOT. d
      uto = "MFG = TMFG"
   ENDIF

   IF a .AND. .NOT. b .AND. .NOT. c .AND. .NOT. d
      uto = "DESCR = TDESC"
   ENDIF

   IF a .AND. c .AND. d .AND. .NOT. b
      uto = "DESCR = TDESC .AND. PM_NO = TPM .AND. " + ;
          "CTR = TCTR"
   ENDIF

   IF a .AND. d .AND. .NOT. b .AND. .NOT. c
      uto = "DESCR = TDESC .AND. CTR = TCTR"
   ENDIF

   IF a .AND. b .AND. d .AND. .NOT. c
      uto = "DESCR = TDESC .AND. MFG = TMFG .AND. " + ;
          "CTR = TCTR"
   ENDIF
```

(continues)

```
IF b .AND. d .AND. .NOT. a .AND. .NOT. c
   uto = "MFG = TMFG .AND. CTR = TCTR"
ENDIF

IF a .AND. c .AND. .NOT. b .AND. .NOT. d
   uto = "DESCR = TDESC .AND. MODEL = TMDL"
ENDIF


* ========================================
*$ Looks for first data entry that is
*  not blank and then will SET ORDER
*  to that associated INDEX. Also sets
*  up a macro used in the DO WHILE
* ========================================

IF a
   z = af
   so = 1
   sos = "DESCR = TDESC"
ENDIF

IF b  .AND.  .NOT. a
   so = 2
   z = bf
   sos = "MFG = TMFG"
ENDIF

IF c .AND. .NOT. a .AND. .NOT. b
   so = 3
   z = cf
   sos = "MODEL = TMDL"
ENDIF

IF d .AND. .NOT. a .AND. .NOT. b .AND. .NOT. c
   so = 4
   z = df
   sos = "CTR = TCTR"
ENDIF

* ========================================
*$ After determining which is the first
*  available data entry that is not blank,
*  SET ORDER TO is set to the correct
*  position. Printer is also turned on.
* ========================================

SET ORDER TO so
GOTO z
SET DEVICE TO PRINT

DO WHILE  &sos

   * ========================================
   *$ Header is printed for your institution
   *  for each page.
   * ========================================

   @ s, 57 SAY "YOUR INSTITUTION'S NAME"
   s = s + 1
   @ s, 60 SAY "SELECTED INVENTORY"
   @ s, 1  SAY "DATE  " + DTOC(DATE())
   @ s,122 SAY "PAGE " + STR(y,3,0)

   s = s +2
   @ s,1   SAY "ID"
   @ s,8   SAY "MANUFACTURER"
   @ s,30  SAY "MODEL"
   @ s,52  SAY "SERIAL NO."
   @ s,64  SAY "DESCRIPTION"
   @ s,90  SAY "PM CODE"
   @ s,98  SAY "EST DATE"
   @ s,108 SAY "PM DATE"
   @ s,118 SAY "STATUS"
   @ s,127 SAY "CTR"
   s = s + 2

   DO WHILE &sos

      * ========================================
      *$ Here's where the work is done.  As long
      *  as there is a match on the current
      *  INDEX, the program will compare each
      *  record looking for matches in the
      *  additional fields.  If a match is
      *  found, it is printed out.
      * ========================================

      IF &uto
         @ s,1    SAY id
         @ s,8    SAY mfg
         @ s,30   SAY model
         @ s,52   SAY sn
         @ s,64   SAY descr
         @ s,90   SAY pm_no
         @ s,98   SAY last_sfty
         @ s,108  SAY last_pm
         @ s,118  SAY STATUS
         @ s,127  SAY ctr
         X = X + 1
         s = s + 1
         SKIP

         * ========================================
         *$ End of page counter.  s is the line
         *  counter and y is the page number.
         * ========================================

         IF s >= 57
            y = y + 1
            s = 1
            EJECT
            EXIT
         ENDIF
         LOOP
      ELSE
         SKIP
         LOOP
      ENDIF
      EXIT
   ENDDO

ENDDO
SET DEVICE TO SCREEN
EJECT

EXIT
ENDDO

RETURN
*: EOF: SEARCH.PRG
```

About the Author:

Herman Rohr is a FoxBASE+/dBASE programmer in the Topeka, Kansas, area who writes small business applications. He is employed as Supervisor of Clinical Engineering at St.Francis Hospital in Topeka.

EOF

EDITORIAL CONT

submissions on all aspects of FoxBASE+, FoxBASE+/Mac and, now, FoxPro. Check the author's guidelines which appear in most issues and share your best work with the Fox community. You'll make some money (and maybe more than some; several contributors report obtaining consulting assignments resulting from their **fox**talk articles) and gain a modicum of recognition (notoriety?). I look forward to seeing your submissions.

Glenn A. Hart

EOF

Three Screens Procedures Cont.

```
         @ ROW+3,RIGHT+1 SAY CHR(219)
         @ ROW+2,RIGHT+1 SAY CHR(219)
         @ ROW+1,RIGHT+1 SAY CHR(219)
         @ ROW,RIGHT+1 SAY CHR(219)
         @ ROW-1,RIGHT+1 SAY CHR(220)
      ENDIF
   ELSE
      IF endline = 2
         @ ROW-1,LEFT CLEAR TO ROW+endline+1,RIGHT
         @ ROW-1,LEFT TO ROW+endline+1,RIGHT DOUBLE
         *** SHADOW ***
         SET COLOR TO &sh_col
         @ ROW+4,LEFT+1 SAY REPLICATE(CHR(223),76)
         @ ROW+3,RIGHT+1 SAY CHR(219)
         @ ROW+2,RIGHT+1 SAY CHR(219)
         @ ROW+1,RIGHT+1 SAY CHR(219)
         @ ROW,RIGHT+1 SAY CHR(219)
         @ ROW-1,RIGHT+1 SAY CHR(220)
      ELSE
         @ ROW-1,LEFT CLEAR TO ROW+endline+2,RIGHT
         @ ROW-1,LEFT TO ROW+endline+2,RIGHT DOUBLE
         *** SHADOW ***
         SET COLOR TO &sh_col
         @ ROW+5,LEFT+1 SAY REPLICATE(CHR(223),76)
         @ ROW+4,RIGHT+1 SAY CHR(219)
         @ ROW+3,RIGHT+1 SAY CHR(219)
         @ ROW+2,RIGHT+1 SAY CHR(219)
         @ ROW+1,RIGHT+1 SAY CHR(219)
         @ ROW,RIGHT+1 SAY CHR(219)
         @ ROW-1,RIGHT+1 SAY CHR(220)
      ENDIF
   ENDIF
   SET COLOR TO &tx_color,&hi_color
   @ ROW,(80-LEN(m1))/2 SAY m1
   @ ROW+1,(80-LEN(m2))/2 SAY m2
   IF endline=3
      @ ROW+2,(80-LEN(m3))/2 SAY m3
   ENDIF
   IF &getflag
      STORE " " TO yn
      IF endline=2
         @ ROW+2,31 SAY "Choice  (Y/N)?  " GET yn;
            PICTURE "!" VALID yn$"YN"
         READ
         SET COLOR TO &old_col
         RESTORE SCREEN
         RETURN
      ELSE
         @ ROW+3,31 SAY "Choice  (Y/N)?  " GET yn;
            PICTURE "!" VALID yn$"YN"
         READ
         SET COLOR TO &old_col
         RESTORE SCREEN
         RETURN
      ENDIF
   ELSE
      xx= INKEY(2)
   ENDIF
  ENDCASE
ELSE
  no_rows=1
  m1=LTRIM(TRIM(msg))
  STORE LEN(m1) TO l_msg
  STORE (80-LEN(m1))/2 TO start
  SET COLOR TO &fr_color
  IF .NOT. &getflag
     @ ROW-1,start-2 CLEAR TO ROW+1,start+1+l_msg
     @ ROW-1,start-2 TO ROW+1,start+1+l_msg DOUBLE
     *** SHADOW ***
     SET COLOR TO &sh_col
     @ ROW+2,start-1 SAY REPLICATE(CHR(223),l_msg+4)
     @ ROW+1,start+2+l_msg SAY CHR(219)
     @ ROW,start+2+l_msg SAY CHR(219)
     @ ROW-1,start+2+l_msg SAY CHR(220)
  ELSE
     IF l_msg>21
        @ ROW-1,start-2 CLEAR TO ROW+2,start+1+l_msg
        @ ROW-1,start-2 TO ROW+2,start+1+l_msg DOUBLE
        *** SHADOW ***
        SET COLOR TO &sh_col
        @ ROW+3,start-1 SAY REPLICATE(CHR(223),l_msg+4)
        @ ROW+2,start+2+l_msg SAY CHR(219)
        @ ROW+1,start+2+l_msg SAY CHR(219)
        @ ROW,start+2+l_msg SAY CHR(219)
        @ ROW-1,start+2+l_msg SAY CHR(220)
     ELSE
        @ ROW-1,29 CLEAR TO ROW+2,50
        @ ROW-1,29 TO ROW+2,50 DOUBLE
        *** SHADOW ***
        SET COLOR TO &sh_col
        @ ROW+3,30 SAY REPLICATE(CHR(223),22)
        @ ROW+2,51 SAY CHR(219)
        @ ROW+1,51 SAY CHR(219)
        @ ROW,51 SAY CHR(219)
        @ ROW-1,51 SAY CHR(220)
     ENDIF
  ENDIF
  SET COLOR TO &tx_color,&hi_color
  @ ROW,start SAY m1
  IF &getflag
     STORE " " TO yn
     @ ROW+1,31 SAY "Choice  (Y/N)?  " GET yn;
        PICTURE "!" VALID yn$"YN"
     READ
     SET COLOR TO &old_col
     RESTORE SCREEN
     RETURN
  ELSE
     xx= INKEY(2)       && display message for 2 seconds
  ENDIF && change to xx=inkey(0) to wait
ENDIF                               &&    for use keystroke
SET COLOR TO &old_col               && restore PRIOR colors
RESTORE SCREEN
RETURN
```

Frame2

'FRAME2' is similar to 'FRAME,' however, it is shorter and simpler and displays a centered, framed, and shadowed message box of one line *only*... up to 74 characters. The shadow is the more common WIDE shadow, CHR(219), and the colors are fixed. Of course you will adjust the colors to suit your own tastes.

'DO frame2 with row( ),"« You Must Enter a Value »"' is an ideal routine to be called from a validation UDF.

'FRAME2' can be called as a data entry prompt box, as in:

```
STORE space(15) to m_last
DO frame2 with 10,"Enter Last Name ' + ;
    REPLICATE(chr(255),15)
@ 10,col()-15 GET m_last PICTURE "@X!"
READ
```

or

```
STORE " " TO yn
DO frame2 WITH 22,"Edit Another Record? "+chr(255)
0 22,col()-1 GET yn PICTURE "Y
READ


***********************************************************
*                        FRAME2                           *
***********************************************************
PARAMETERS line,msg
* Len Levy, Data Management Systems
***
***    LINE = Row on which message is to appear
***    MSG = Any message to be centered and framed up to 74
***        characters
***
***    Syntax Example:
***
***    DO frame2 WITH row(),"Hit Any Key to Continue"
```

(continues)

```
***      Will display center and framed message on the
***      current line
************************************************************
***   STORE " " TO yn
***   DO frame2 WITH 12,"Do You Wish To Quit?"+chr(255)
***   @ 12,col()-1 GET yn PICTURE "!" VALID yn$"YN"
***   READ
***      Will display centered and framed query on line 12
***
************************************************************
***   STORE space(15) TO lname
***   DO frame2 WITH 8,"Enter Last Name ";
***      +REPLICATE(chr(255),15)
***   @ 8,col()-15 GET lname PICTURE "@X!"
***   READ
***      Will display centered and framed query on line 8
***
************************************************************

PRIVATE msg_len,msg_start,msg_end,box_start,box_end
PRIVATE sha_start,sha_end,xx
old_col=SYS(2001,"COLOR")
msg_len = LEN(TRIM(msg))
IF msg_len>74                                   && Error trap
   CLEAR
   @ 11,17 TO 13,62 double
   @ 12,19 SAY "Message Length Exceeds 74 Character Limit!"
   xx=INKEY(2)
   RETURN
ENDIF
IF line>22                                      && Error trap
   CLEAR
   @ 11,12 TO 13,66 double
   @ 12,14 SAY "Starting Line is too low. "+;
      "Message will be scrolled"
   xx=INKEY(2)
   RETURN
ENDIF
?? SYS(2002)                                    && Cursor off
msg_start = (80-msg_len)/2
msg_end = msg_start+LEN(TRIM(msg))
box_start = msg_start-3
box_end = msg_end+2
sha_start = box_start+2
sha_end = box_end+2
SET COLOR TO n/n
@ line,sha_start CLEAR TO line+2,sha_end      && Draw shadow
IF ISCOLOR()
   SET COLOR TO r/bg           && Select your own box colors
ELSE
   SET COLOR TO n/w+
ENDIF
@ line-1,box_start CLEAR TO line+1,box_end    && Box outline
@ line-1,box_start TO line+1,box_end double
IF ISCOLOR()
   SET COLOR TO w+/bg      && Select your own message colors
ENDIF
@ line,msg_start SAY msg               && Print message in box
SET COLOR TO &OLD_COL
?? SYS(2002,1)                                  && Cursor on
RETURN
```

These procedures are certainly time savers, but just as important, they add consistency to your applications.

About the Author:

Len Levy is a FoxBASE+ programmer, consultant, teacher, and musician. He started as a teacher in the Yonkers, N.Y. Public Schools System in 1961, initially teaching music and later teaching computer courses in the middle schools. Since 1986 he has worked continually as Programmer and Network Administrator for Yonkers' Special Education Department. He began computing in 1977 when the Tandy Model I appeared and, since 1983, has been addicted to dBASE. His private consulting business, DATA MANAGEMENT SYSTEMS, is based in Scarsdale, N.Y.

EOF

Writer's Guidelines:

The editorial staff of **foxtalk** encourages anyone who is interested in writing for the journal to submit articles to us. We realize that our readership will be better served by a journal that presents a wide variety of viewpoints, expertise, and topics rather than just those which come from our own FoxBASE+ experience.

If you would like to write for the journal, here are some rules we must ask you to follow:

1. All material, prose, and source code must be original and must be owned by you.

2. You must inform us if other publications are considering your article.

3. Pinnacle Publishing reserves all rights to all articles published in **foxtalk**. Be aware that your articles, and, particularly your code, may be used by readers in the development and implementation of individual applications.

4. Articles may be submitted on a diskette (5 1/4," 360K or 1.2M) or via modem. The preferred format is Microsoft Word. Also acceptable is straight ASCII text.

5. *All text must be clean and unformatted,* free of indentations, tabs, control codes, and manual "eyeballing" alignment of text using the space bar. If you need to emphasize key words, or extensively format a table or diagram, please enclose a separate printout of your article that shows the appropriate emphasis. You should include a proposed title and a brief biographical sketch which may appear at the end of the article.

6. Please make certain that no line of text exceeds 60 characters.

7. The editorial staff will decide what to publish and when based on what we think is of most interest to our readers, and on the quality of material submitted. Our target audience is the serious, professional FoxBASE+ user, so please write with that individual in mind.

All material submitted for publication will receive fair and judicious treatment; however, we cannot guarantee publication to anyone. Honoraria are paid after publication.