141 VM

May 1998

update

In this issue

VM Update

Published by
Xephon
27-35 London Road
Newbury
Berkshire RG14 1JL
England
Telephone: 01635 38030
From USA: 01144 1635 38030
E-mail: xephon@compuserve.com

North American office
Xephon/QNA
1301 West Highway 407, Suite 201-405
Lewisville, TX 75067
USA
Telephone: 940 455 7050

Editorial panel
Articles published in *VM Update* are reviewed by our panel of experts. Members of the panel include John Illingworth (UK), Reinhard Meyer (Germany), Philippe Taymans (Belgium), Romney White (USA), Martin Wicks (UK), and Jim Vincent (USA).

Subscriptions and back-issues
A year's subscription to *VM Update*, comprising twelve monthly issues, costs £175.00 in the UK; $265.00 in the USA and Canada; £181.00 in Europe; £187.00 in Australasia and Japan; and £185.50 elsewhere. In all cases the price includes postage. Individual issues, starting with the January 1990 issue, are available separately to subscribers for £14.50 ($22.50) each including postage.

Editor
Robert Burgess

Disclaimer
Readers are cautioned that, although the information in this journal is presented in good faith, neither Xephon nor the organizations or individuals that supplied information in this journal give any warranty or make any representations as to the accuracy of the material it contains. Neither Xephon nor the contributing organizations or individuals accept any liability of any kind howsoever arising out of the use of such material. Readers should satisfy themselves as to the correctness and relevance to their circumstances of all advice, information, code, JCL, EXECs, and other contents of this journal before making any use of it.

VM Update on-line
Code from *VM Update* can be downloaded from our Web site at http://www.xephon.com; you will need the user-id shown on your address label.

Contributions
Articles published in *VM Update* are paid for at the rate of £170 ($250) per 1000 words for original material. To find out more about contributing an article, without any obligation, please contact us at any of the addresses above and we will send you a copy of our *Notes for Contributors*.

Printed in England.

Checking a REXX file for correct pairs

When you have written a complex REXX program, runtime errors can often occur because of the wrong coding of recursive loops. This can be avoided by indenting ‘do loops’, or you can try using a procedure that checks the accuracy of your code. The following XEDIT macro performs this task. When an error is encountered, the file is displayed with only the ‘do/select/end’ statements shown, to enable a better survey of the loops. You get back to the complete file display by entering the ALL command.

The tokens ‘do’, ‘select’, ‘end’, ‘do;’, ‘select;’, and ‘end;’ are recognized. These must not contain capitals and must be delimited by at least one space.

DOENDS XEDIT

```
/*********************************************************************/
/* Check a REXX file for pairs of 'do/end' or 'select/end'.          */
/* 'do', 'select', and 'end' (plus 'do;' 'select;' and 'end;') are   */
/* recognized; these tokens must be delimited by at least one blank  */
/* and may not be coded with capitals.                               */
/* Comments with embedded do/end or select/end are ignored           */
/*********************************************************************/
dos   = 0
ends  = 0
depth = 0

'EXTRACT /WRAP/SHADOW'
'SET WRAP OFF'
'SET CMSTYPE HT'
'SET MSGMODE OFF'
'SET SHADOW OFF'

signal on syntax name error

do i = 1
   'LOCATE/do /|/do;/|/select /|/select;/|/end /|/end;/'
   if rc ¬= 0 then leave i
   'STACK'
   parse pull line
   line = strip(line)
   if (wordpos('do',line)     > 0 | wordpos('do;',line)     > 0  | ,
       wordpos('select',line) > 0 | wordpos('select;',line) > 0)   ,
```

```
       & ¬(left(line,2) = '/*'    & right(line,2) = '*/')    then do
      dos = dos + 1
      depth = depth + 1
   end
   if (wordpos('end',line)    > Ø | wordpos('end;',line)    > Ø)   ,
       & ¬(left(line,2) = '/*'    & right(line,2) = '*/')    then do
      ends = ends + 1
      depth = depth - 1
   end
   if depth < Ø then do
      'EXTRACT /LINE'
      leave i
   end
end
'SET MSGMODE ON'
'SET WRAP' wrap.1
'SET CMSTYPE RT'
if dos = ends then do
   'SET SHADOW' shadow.1
   'TOP'
   'MSG File is OK'
end
else do
   if dos < ends then do
     'MSG Too many END statements (' (-depth) 'items ) continue with
ALL'
     ':Ø ALL/do /|/do;/|/select /|/select;/|/end /|/end;/'
     ':'line.1
     'CURSOR FILE' line.1 '1'
   end
   else do
     'MSG END statements missing (' depth 'items ) continue with ALL'
     ':Ø ALL/do /|/do;/|/select /|/select;/|/end /|/end;/'
   end
end
exit

error:
'SET MSGMODE ON'
'SET WRAP' wrap.1
'SET SHADOW' shadow.1
'SET CMSTYPE RT'
'MSG Error in procedure DOENDS (line =' sigl ')'
':Ø ALL'
```

Dr Reinhard Meyer (Germany)

Finding critical variables

This utility can be very helpful when you have been asked to fix a REXX program, but have no prior knowledge of it. The utility enables you to find the critical variables relevant to the problem.

This EXEC shows REXX variables that are defined by their names or their values. The definition may use the asterisk (*) as a wild-card character.

The EXEC is called by:

```
call LOOKVARS [<fvar>],[<fval>],[<out>]
```

Where:

- ‘fvar’ is the filter for variable names (* by default).
- ‘fval’ is the filter for values (* by default).
- ‘out’ is the output pipeline stage(s) (CONSOLE by default).

Note: filters may contain ‘*’ as a wild-card character, which is case-insensitive.

The most likely form of call will be the manual call from the REXX application being traced by TRACE ?R. For example, in a year 2000-related situation:

```
Call lookvars '*','*96*'
```

will show all the variables having values including ‘96’.

```
Call lookvars '*','*96','| > VARS OUTPUT A'
```

will write all variables whose values end with ‘96’ to the file VARS OUTPUT A.

The third parameter is automatically appended to the end of the pipeline, allowing the pipeline to have more than one stage.

The layout of the output is:

```
<varname> : <value>
```

LOOKVARS EXEC

```
if arg(1) <> '' then varFilter = arg(1); else varFilter = '*'
if arg(2) <> '' then valFilter = arg(2); else valFilter = '*'
if arg(3) <> '' then Output    = arg(3); else output    = 'CONSOLE'

upper varFilter valFilter output

                                          /* Analyse varname filter */
len = length(varFilter)
AddProcessing= Ø
Select
  when varFilter = '*' then filter = ''          /*no filter        */
  when pos('*',varFilter) = Ø                    /*no wildchars     */
    then Filter = 'Locate 1.'len+2 ,
       '/'varFilter'/'
  when right(varFilter,1) = '*' ,                /*only 1 at the end*/
     & pos('*',substr(varFilter,1,len-1)) = Ø
    then do
     s = substr(varFilter,1,len-1)
     filter = 'Locate 1.'length(s) '/'s'/'
    end

  Otherwise
    parse var varFilter s1'*'s2'*'
    Select
     when s1 s2 = '' then filter = ''
     when s1    = '' then filter = 'LOCATE /'s2'/'
     when    s2 = '' then filter = 'LOCATE /'s1'/'
     otherwise           filter = 'ALL /'s1'/ & /'s2'/'
    end
    AddProcessing=1
end
if Filter <> '' then filter = '|Zone FS : F 1 ' Filter  /*var : value*/

                                          /* Analyse value filter    */
len = length(valFilter)
Select
  when valFilter = '*' then filter2= ''          /*no filter         */
  when pos('*',valFilter) = Ø                    /*no wildchars      */
    then Filter2= 'Locate 2.'len+2  ,
       '/'valFilter'/'
  when right(vaLFilter,1) = '*' ,                /*only 1 at the end */
     & pos('*',left(valFilter,len-1)) = Ø
    then do
     s = left(valFilter,len-1)
     filter2= 'Locate 2.'length(s) '/'s'/'
    end

  Otherwise
```

```
    parse var valFilter s1'*'s2'*'
    Select
     when s1 s2 = '' then filter2= ''
     when s1    = '' then filter2= 'LOCATE /'s2'/'
     when    s2 = '' then filter2= 'LOCATE /'s1'/'
     otherwise           filter2= 'ALL /'s1'/ & /'s2'/'
    end
    AddProcessing=1
end
if Filter2<> ''
 then filter2= '| CaseI Zone FS : F 2 ' Filter2 /*var : value */

p=,
'PIPE '           ,
 'REXXVARS 1'     ,    /* extract caller's vars       */
 '| DROP 1  '     ,    /* skip header in 1st line     */
 '| SPEC 3-*'     ,    /* skip leading "n" or "v"     */
 '| JOIN 1 / : /' ,    /* merge varname and value     */
    Filter        ,    /* apply var filter, if any    */
    Filter2            /* apply val filter, if any    */

address command
If ¬AddProcessing
 then do;p '|' OUTPUT; exit;end

p '| STEM ARRAY.'/* preliminary filtering to reduce further processing*/
outCount=Ø
do candidate=1 to array.Ø
  parse var array.candidate varName' : 'Value
  if ¬passes(varname,varFilter) then iterate candidate
  if ¬passes(value  ,valFilter) then iterate candidate
  outCount = outCount+1              /* passed both filters, output */
  out.outCount = array.candidate
end candidate
out.Ø = outCount
'PIPE STEM OUT. |' output            /* perform output              */
return rc

Passes: Procedure   /*returns logical: (value meets filter?) */
arg value,filter

Do forever
  if filter='*' then return 1           /* any value goes */
  if pos('*',filter) = Ø & filter = value then return 1
  if left(filter,1) = '*' & filter<>'*'
   then do
      filter= substr(filter,2)
      anyPositionGood   = 1
   end
   else anyPositionGood = Ø   /* only 1st position good     */
```

```
  parse var filter piece'*'
  if piece='' then
    return (filter = '' & value = '' | filter = '*')
  position = pos(piece,value)
  Good = ,
      ( anyPositionGood   & (position <> Ø)   )   ,
   |  (¬anyPositionGood   & (position =  1)   )
  if ¬good then return Ø
  parse var filter (piece) filter
  Select
    when filter='*'                       /* any value goes  */
     then return 1
    when filter=''                        /* end of filter   */
     then          /* value should end with last non-star filter part */
        return (right(value,length(piece)) = piece)
    otherwise                           /* filter<>'', take next piece  */
     parse var value  (piece) value
   end
end
```

Vadim Rapp
Vadim Rapp Ltd (USA)

REXX tracking system re-visited – part 2

This month we continue the code for the Problem Tracking Facility (PTF), which has been re-written to be Year 2000 compatible.

```
IF CPFKEY = 'PFØ8' & PAGENBR < 7 THEN DO
   INCNBR = INCNBR+1
   PAGENBR = PAGENBR+1
   CALL FULLIST3
   SIGNAL CKRESP ; END

IF CPFKEY = 'PFØ7' THEN DO
   MESSAGE = 'Already at the beginning...'
   CALL DISPØ7 ; SIGNAL CKRESP ; END

IF CPFKEY = 'PFØ8' THEN DO
   MESSAGE = 'No more entries to display...'
   CALL DISPØ7 ; SIGNAL CKRESP ; END

IF CPFKEY ¬= 'PFØ3' THEN DO
   CALL DISPØ7 ; END
```

```
ELSE
   IF PSCREEN = 'PTFØØ' THEN DO
      P7TYPE = ' ' ; CALL DISPØØ ; END
RETURN

/*********************************************************************/
/***  GENERATE NEW INCIDENT AND PLACE USER INTO XEDIT MODE        ***/
/*********************************************************************/
BUILDNEW:

/*********************************************************************/
/***  GENERATE NEW INCIDENT NUMBER                                 ***/
/*********************************************************************/

CALL GETLNS                                 /* NBR LINES IN DIRECT.  */
WORK  = 1ØØØØØ1 + LNS                       /* INCR BY 1             */
INCID = SUBSTR(P64,1,2) || SUBSTR(DATE(J),1,2) || SUBSTR(WORK,4,4)

/*********************************************************************/
/***  DISPLAY & PROCESS DESCRIPTION PANEL                          ***/
/*********************************************************************/

 DESC1  = '' ; DESC2 =''
 COMP = '' ; VEND = '' ; VREF = '' ; V = ''
 CTIME = TIME() ; CUSER = USERID()
 UTIME = TIME() ; UUSER = USERID()

 UDATE    = WORD(DATE(N),3)
 UDATE    = UDATE || SUBSTR(DATE(U),1,2)
 UDATE    = UDATE || SUBSTR(DATE(U),4,2)
 CDATE    = UDATE

 S = 'O' ; PFRET = '' ; CALL SETCNTL

/*********************************************************************/
/***  BUILD DIRECTORY ENTRY AND UPDATE DIRECTORY                   ***/
/*********************************************************************/

WORKØ = INCID || ' ' || S || ' ' || CDATE || ' ' || TIME()
WORK1 = ' ' || SUBSTR(USERID(),1,8) || ' ' || UDATE
WORK2 = ' ' || TIME() || ' ' || SUBSTR(USERID(),1,8)
WORK3 = ' ' || COMP || ' ' || VEND || ' ' || SUBSTR(VREF,1,8)
WORK4 = ' ' || V || ' ' || DESC1 || DESC2
WORK5 = WORKØ || WORK1 || WORK2 || WORK3 || WORK4

ADDRESS COMMAND 'EXECIO 1 DISKW' DIRFN DIRFT INCFM 'Ø F 2ØØ (STRING'
WORK5

IF RC¬=Ø THEN DO
   CALL DIRERR
```

```
   RETURN ; END

/*********************************************************************/
/***  BUILD INCIDENT MEMBER                                        ***/
/*********************************************************************/

 IF P7Ø = 'NO' THEN
    NOP
 ELSE DO
    WORKA = COPIES(" ",7Ø)
    WORK  = WORKA
    CALL INCWRT
    WORK  = ' 2       DESCRIPTION: 'DESC1
    CALL INCWRT
    WORK  = ' 2                    'DESC2
    CALL INCWRT
    WORK  = WORKA
    CALL INCWRT
    ADDRESS COMMAND 'FINIS' INCID INCFT INCFM
    END

/* CALL INCWRT                                                      */
/* WORK  = '          TRACKING:'                                    */
/* CALL INCWRT                                                      */
/* WORK  = WORKA                                                    */
/* CALL INCWRT ; CALL INCWRT ; CALL INCWRT; CALL INCWRT ; CALL INCWRT*/
/* WORK  = '          RESOLUTION:'                                  */
/* CALL INCWRT                                                      */
/* WORK  = WORKA ; CALL INCWRT                                      */

ADDRESS COMMAND 'COPYFILE' P68 P69 '*' INCID INCFT INCFM '(APPEND'
EDCTRL = 'Y' ; CALL EDPRINT
ADDRESS COMMAND 'XEDIT' INCID INCFT INCFM '(PROF' P62
ADDRESS XEDIT
ADDRESS
EDCTRL = '' ; ADDRESS COMMAND 'ERASE' INCID EDPRTFT EDPFM
RETURN

INCWRT:

ADDRESS COMMAND 'EXECIO 1 DISKW' INCID INCFT INCFM ' (STRING' WORK
IF RC¬=Ø THEN SIGNAL INCERR
RETURN

/*********************************************************************/
/***  WORK WITH A SINGLE INCIDENT                                  ***/
/*********************************************************************/
WORK:

SVDATE   = DATE   ; SVCOMP = COMP  ; SVS    = S    ; SVVREF = VREF
```

```
SVAUTHOR = AUTHOR ; SVVEND = VEND  ; SVSRCH = SRCH

ADDRESS COMMAND 'STATE' INCID INCFT INCFM
IF RC¬=Ø THEN DO
   MESSAGE = P33 'not found'
   SET EMSG ON
   DATE   = SVDATE   ; COMP = SVCOMP  ; S    = SVS  ; VREF = SVVREF
   AUTHOR = SVAUTHOR ; VEND = SVVEND  ; SRCH = SVSRCH
   RETURN ; END

SET EMSG ON ; CALL GETPARSE
PFRET = ''  ; CALL SETCNTL
IF PFRET = 'Y' THEN DO
   DATE   = SVDATE   ; COMP = SVCOMP  ; S    = SVS    ; VREF = SVVREF
   AUTHOR = SVAUTHOR ; VEND = SVVEND  ; SRCH = SVSRCH
   RETURN ; END

IF ACCSW ¬= 'R/O' THEN CALL UPDDIR
EDCTRL = 'Y' ; CALL EDPRINT
ADDRESS COMMAND 'XEDIT' INCID INCFT '* (PROF' P62
ADDRESS XEDIT
ADDRESS
EDCTRL = ''
ADDRESS COMMAND 'ERASE' INCID EDPRTFT EDPFM
SET EMSG ON

DATE   = SVDATE   ; COMP = SVCOMP  ; S    = SVS   ; VREF = SVVREF
AUTHOR = SVAUTHOR ; VEND = SVVEND  ; SRCH = SVSRCH
RETURN

/*********************************************************************/
/***  ERROR DURING DIRECTORY UPDATE                                ***/
/*********************************************************************/
DIRERR:
MESSAGE='Error during directory update'
RETURN

/*********************************************************************/
/***  ERROR DURING INCIDENT CREATION                               ***/
/*********************************************************************/
INCERR:
MESSAGE = 'Error while creating member. Contact Tech Support.'
SIGNAL INIT

/*********************************************************************/
/***  DISPLAY THE FULLIST PANEL                                    ***/
/*********************************************************************/
FULLIST:

CALL GETLNS
```

```
INCNBR  = 1        /*  NUMBER THAT APPEARS ON LINE-ITEM 1, 2, 3...  */
PAGENBR = 1        /*  CURRENT PAGE BEING DISPLAYED                 */

FULLIST2:
ACTR1 = 1 ; ACTR2 = 15
PLNS.PAGENBR      = WLNS              /* PREVIOUS PAGE STARTER     */
PINCNBR.PAGENBR   = INCNBR            /* PREVIOUS PAGE STARTER     */

DO 14                                 /* CLEAR ITEM ARRAY          */
  Z.ACTR1=''  ; Z.ACTR2=''
  N.ACTR1=''  ; I.ACTR1=''  ; A.ACTR1='' ; D.ACTR1=''
  F1.ACTR1='' ; F2.ACTR1=''
  ACTR1 = ACTR1+1 ; ACTR2 = ACTR2+1
END

INCNBR2   = 1      /*  SEQUENTIAL LINE NUMBER OF PANEL (1-14)       */
ZCTRA     = 1      /*  COUNTER FOR FIRST Z-ITEM IN LINE             */
ZCTRB     = 2      /*  COUNTER FOR SECOND Z-ITEM IN LINE            */

/* FVGDIRFT = DIRFT ; FVGWLNS = WLNS ; FVGLNS = LNS   */
DO UNTIL D.14 ¬= ''

  CALL GETDIR
  IF (RETCD = '2' | HIT = 'N') & INCNBR = 1 THEN DO
     MESSAGE = 'No matching entries found'
     SIGNAL INIT ; END

  IF RETCD = '2' | HIT = 'N' THEN DO
     LEAVE ; END

  DX2.1 = INCIDENT
  DX2.2 = LUPDDATE
  DX2.3 = OPENUSER
  DX2.4 = COMPID
  DX2.5 = VENDID
  DX2.6 = VREFID
  DX2.7 = SEV
  DX2.8 = OPENDATE

  F1.INCNBR2 = DX2.P12
  F2.INCNBR2 = DX2.P13

  Z.ZCTRA   = '_'
  N.INCNBR2 = INCNBR
  I.INCNBR2 = INCIDENT
  Z.ZCTRB   = STATUS
  D.INCNBR2 = SUBSTR(ENTRY,95,5Ø)
  WRK = INCNBR2+1
  D.WRK     = SUBSTR(ENTRY,145,5Ø)
```

```
  IF D.WRK  = '' THEN DO
     INCNBR  = INCNBR+1
     INCNBR2 = INCNBR2+1
     ZCTRA   = ZCTRA+2
     ZCTRB   = ZCTRB+2
     NOP ; END
  ELSE DO
     INCNBR  = INCNBR+1
     INCNBR2 = INCNBR2+2
     ZCTRA   = ZCTRA+4
     ZCTRB   = ZCTRB+4
     NOP ; END
END
/* DIRFT = FVGDIRFT ; WLNS = FVGWLNS ; LNS = FVGLNS  */

Z1  = Z.1 ; Z15 = Z.15 ; N1  = N.1 ; I1  = I.1 ; A1  = A.1 ;D1  = D.1
Z2  = Z.2 ; Z16 = Z.16 ; N2  = N.2 ; I2  = I.2 ; A2  = A.2 ;D2  = D.2
Z3  = Z.3 ; Z17 = Z.17 ; N3  = N.3 ; I3  = I.3 ; A3  = A.3 ;D3  = D.3
Z4  = Z.4 ; Z18 = Z.18 ; N4  = N.4 ; I4  = I.4 ; A4  = A.4 ;D4  = D.4
Z5  = Z.5 ; Z19 = Z.19 ; N5  = N.5 ; I5  = I.5 ; A5  = A.5 ;D5  = D.5
Z6  = Z.6 ; Z2Ø = Z.2Ø ; N6  = N.6 ; I6  = I.6 ; A6  = A.6 ;D6  = D.6
Z7  = Z.7 ; Z21 = Z.21 ; N7  = N.7 ; I7  = I.7 ; A7  = A.7 ;D7  = D.7
Z8  = Z.8 ; Z22 = Z.22 ; N8  = N.8 ; I8  = I.8 ; A8  = A.8 ;D8  = D.8
Z9  = Z.9 ; Z23 = Z.23 ; N9  = N.9 ; I9  = I.9 ; A9  = A.9 ;D9  = D.9
Z1Ø = Z.1Ø; Z24 = Z.24 ; N1Ø = N.1Ø; I1Ø = I.1Ø; A1Ø = A.1Ø;D1Ø = D.1Ø
Z11 = Z.11; Z25 = Z.25 ; N11 = N.11; I11 = I.11; A11 = A.11;D11 = D.11
Z12 = Z.12; Z26 = Z.26 ; N12 = N.12; I12 = I.12; A12 = A.12;D12 = D.12
Z13 = Z.13; Z27 = Z.27 ; N13 = N.13; I13 = I.13; A13 = A.13;D13 = D.13
Z14 = Z.14; Z28 = Z.28 ; N14 = N.14; I14 = I.14; A14 = A.14;D14 = D.14

F11  = F1.1 ; F18  = F1.8
F12  = F1.2 ; F19  = F1.9
F13  = F1.3 ; F11Ø = F1.1Ø
F14  = F1.4 ; F111 = F1.11
F15  = F1.5 ; F112 = F1.12
F16  = F1.6 ; F113 = F1.13
F17  = F1.7 ; F114 = F1.14

F21  = F2.1 ; F28  = F2.8
F22  = F2.2 ; F29  = F2.9
F23  = F2.3 ; F21Ø = F2.1Ø
F24  = F2.4 ; F211 = F2.11
F25  = F2.5 ; F212 = F2.12
F26  = F2.6 ; F213 = F2.13
F27  = F2.7 ; F214 = F2.14

ZCSR = 'Z1'                          /* DEFAULT CURSOR POSITION   */
CALL DISPØ1
RETURN
/*********************************************************************/
```

```
/***  DISPLAY THE LIST PANEL                                          ***/
/*********************************************************************/
FULLIST3:

IF P7TYPE = '17' THEN DO
Z1  = INCNBR; XID1  = CID.INCNBR; XDS1  = CDS.INCNBR; INCNBR =INCNBR+1
Z2  = INCNBR; XID2  = CID.INCNBR; XDS2  = CDS.INCNBR; INCNBR =INCNBR+1
Z3  = INCNBR; XID3  = CID.INCNBR; XDS3  = CDS.INCNBR; INCNBR =INCNBR+1
Z4  = INCNBR; XID4  = CID.INCNBR; XDS4  = CDS.INCNBR; INCNBR =INCNBR+1
Z5  = INCNBR; XID5  = CID.INCNBR; XDS5  = CDS.INCNBR; INCNBR =INCNBR+1
Z6  = INCNBR; XID6  = CID.INCNBR; XDS6  = CDS.INCNBR; INCNBR =INCNBR+1
Z7  = INCNBR; XID7  = CID.INCNBR; XDS7  = CDS.INCNBR; INCNBR =INCNBR+1
Z8  = INCNBR; XID8  = CID.INCNBR; XDS8  = CDS.INCNBR; INCNBR =INCNBR+1
Z9  = INCNBR; XID9  = CID.INCNBR; XDS9  = CDS.INCNBR; INCNBR =INCNBR+1
Z1Ø = INCNBR; XID1Ø = CID.INCNBR; XDS1Ø = CDS.INCNBR; INCNBR =INCNBR+1
Z11 = INCNBR; XID11 = CID.INCNBR; XDS11 = CDS.INCNBR; INCNBR =INCNBR+1
Z12 = INCNBR; XID12 = CID.INCNBR; XDS12 = CDS.INCNBR; INCNBR =INCNBR+1
Z13 = INCNBR; XID13 = CID.INCNBR; XDS13 = CDS.INCNBR; INCNBR =INCNBR+1
Z14 = INCNBR; XID14 = CID.INCNBR; XDS14 = CDS.INCNBR
END
ELSE DO
Z1  = INCNBR; XID1  = VID.INCNBR; XDS1  = VDS.INCNBR; INCNBR =INCNBR+1
Z2  = INCNBR; XID2  = VID.INCNBR; XDS2  = VDS.INCNBR; INCNBR =INCNBR+1
Z3  = INCNBR; XID3  = VID.INCNBR; XDS3  = VDS.INCNBR; INCNBR =INCNBR+1
Z4  = INCNBR; XID4  = VID.INCNBR; XDS4  = VDS.INCNBR; INCNBR =INCNBR+1
Z5  = INCNBR; XID5  = VID.INCNBR; XDS5  = VDS.INCNBR; INCNBR =INCNBR+1
Z6  = INCNBR; XID6  = VID.INCNBR; XDS6  = VDS.INCNBR; INCNBR =INCNBR+1
Z7  = INCNBR; XID7  = VID.INCNBR; XDS7  = VDS.INCNBR; INCNBR =INCNBR+1
Z8  = INCNBR; XID8  = VID.INCNBR; XDS8  = VDS.INCNBR; INCNBR =INCNBR+1
Z9  = INCNBR; XID9  = VID.INCNBR; XDS9  = VDS.INCNBR; INCNBR =INCNBR+1
Z1Ø = INCNBR; XID1Ø = VID.INCNBR; XDS1Ø = VDS.INCNBR; INCNBR =INCNBR+1
Z11 = INCNBR; XID11 = VID.INCNBR; XDS11 = VDS.INCNBR; INCNBR =INCNBR+1
Z12 = INCNBR; XID12 = VID.INCNBR; XDS12 = VDS.INCNBR; INCNBR =INCNBR+1
Z13 = INCNBR; XID13 = VID.INCNBR; XDS13 = VDS.INCNBR; INCNBR =INCNBR+1
Z14 = INCNBR; XID14 = VID.INCNBR; XDS14 = VDS.INCNBR
END

IF P7UPIP = 'Y' THEN
   RETURN

CALL DISPØ7
CALL SETCFG17
RETURN

/*********************************************************************/
/***  EXECIO ERROR                                                   ***/
/*********************************************************************/
CIOERR:
MESSAGE = 'EXECIO error encountered. RC='RETCD'. Last screen ='LSCREEN
SIGNAL INIT
```

```
/*********************************************************************/
/***  GET DIRECTORY ENTRY                                          ***/
/***                                                               ***/
/***  NEWER INCIDENTS RESIDE AT THE END OF THE DIRECTORY.          ***/
/***  THEREFORE, THE DIRECTORY IS READ IN LIFO SEQUENCE            ***/
/***  TO OBTAIN THE NEWEST INCIDENTS FIRST. LIFO CONTROL IS        ***/
/***  PROVIDED BY THE "LNS" VALUE INSTEAD OF ON THE EXECIO CMD.    ***/
/***                                                               ***/
/***  EACH EXECUTION OF THIS ROUTINE CAUSES THE DIRECTORY TO       ***/
/***  BE READ CONTINUOUSLY FROM THE LAST SELECTED RECORD UNTIL     ***/
/***  A RECORD MATCHING THE SEARCH CRITERIA IS FOUND.              ***/
/*********************************************************************/
GETDIR:

HIT = 'N'
DO FOREVER
VALID = 'Y'
IF WLNS=Ø THEN DO
   SVGDIRFT = DIRFT ; SVGWLNS = WLNS ; SVGLNS = LNS
   DIRWK = SUBSTR(DIRFT,7,2)
   IF CPFKEY = 'PFØ7' THEN DO
      DIRWK = DIRWK + 1
      IF DIRWK = 1ØØ THEN
         DIRWK = ØØ
      ELSE
         NOP
      END
   ELSE DO
      IF DIRWK = Ø THEN
         DIRWK = '99'
      ELSE
         DIRWK = DIRWK - 1
      END
   DIRFT = X2C(C489998583A3) || DIRWK
   CALL GETLNS
   IF LNS = Ø THEN DO
      DIRFT = SVGDIRFT ; WLNS = SVGWLNS ; LNS = SVGLNS ; LEAVE ; END
   ELSE DO
      NOP ; END
   END
IF WORK = 'Y' THEN DIRFT = X2C(C489998583A3) || SUBSTR(INCID,3,2)
ADDRESS COMMAND 'EXECIO 1 DISKR' DIRFN DIRFT INCFM WLNS '(VAR ENTRY'

RETCD=RC
IF RETCD  = 2 THEN ITERATE                  /* EOF               */
IF RETCD ¬= Ø THEN SIGNAL CIOERR            /* OTHER ERROR       */

PARSE VAR ENTRY INCIDENT STATUS OPENDATE OPENTIME OPENUSER LUPDDATE
LUPDTIME LUPDUSER COMPID VENDID VREFID SEV DESC .
```

```
IF WORK = 'Y' THEN
   RETURN
ELSE
   WLNS=WLNS-1
IF STATUS = 'D' THEN ITERATE         /* IGNORE DELETED INCIDENTS */

/*********************************************************************/
/***  VERIFY DATE CRITERIA IF PRESENT                              ***/
/*********************************************************************/

IF DATE = '' THEN
   NOP
ELSE
   IF S = 'C' & STATUS = 'C' & (LUPDDATE = DATE | LUPDDATE> DATE) THEN
      NOP
   ELSE
      IF OPENDATE = DATE | OPENDATE > DATE THEN
         NOP
      ELSE
         VALID = 'N'

/*********************************************************************/
/***  VERIFY AUTHOR CRITERIA IF PRESENT                            ***/
/*********************************************************************/

WORK5 = ''
IF INDEX(AUTHOR,'*',1) > Ø THEN DO     /* ASTERISKS IN SRCH WORD?  */
   ARG1  = AUTHOR ; ARG2 = OPENUSER    /* USR, DIRECTORY VALUES    */
   CALL EXTRACT ; END                  /* DELIMIT SEARCH WORD      */
IF AUTHOR = '' | AUTHOR = OPENUSER | WORK5 > Ø THEN
   NOP
ELSE
   VALID = 'N'

/*********************************************************************/
/***  VERIFY STATUS CRITERIA                                       ***/
/*********************************************************************/

IF S = '' | S= 'A' | S = STATUS THEN
   NOP
ELSE
   VALID = 'N'
/*********************************************************************/
/***  VERIFY COMPONENT CRITERIA                                    ***/
/*********************************************************************/

WORK5 = ''
IF INDEX(COMP,'*',1) > Ø THEN DO       /* ASTERISKS IN SRCH WORD?  */
   ARG1  = COMP ; ARG2 = COMPID        /* USR, DIRECTORY VALUES    */
   CALL EXTRACT ; END                  /* DELIMIT SEARCH WORD      */
```

```
IF COMP = '' | COMP = COMPID | WORK5 > Ø THEN
   NOP
ELSE
   VALID = 'N'

/*********************************************************************/
/***  VERIFY VENDOR NAME CRITERIA                                  ***/
/*********************************************************************/

WORK5 = ''
IF INDEX(VEND,'*',1) > Ø THEN DO       /* ASTERISKS IN SRCH WORD?  */
   ARG1  = VEND ; ARG2 = VENDID        /* USR, DIRECTORY VALUES    */
   CALL EXTRACT ; END                  /* DELIMIT SEARCH WORD      */

IF VEND = '' | VEND = VENDID | WORK5 > Ø THEN
   NOP
ELSE
   VALID = 'N'

/*********************************************************************/
/***  VERIFY VENDOR REFERENCE NUMBER CRITERIA                      ***/
/*********************************************************************/

WORK5 = ''
IF INDEX(VREF,'*',1) > Ø THEN DO       /* ASTERISKS IN SRCH WORD?  */
   ARG1  = VREF ; ARG2 = VREFID        /* USR, DIRECTORY VALUES    */
   CALL EXTRACT ; END                  /* DELIMIT SEARCH WORD      */

IF VREF = '' | VREF = VREFID | WORK5 > Ø THEN
   NOP
ELSE
   VALID = 'N'

/*********************************************************************/
/***  VERIFY SEARCH-WORD CRITERIA                                  ***/
/*********************************************************************/

IF SRCH = '' THEN
   NOP
ELSE
   CALL SEARCH
/*********************************************************************/
/***  TABULATE RESULTS                                             ***/
/*********************************************************************/

IF VALID = 'Y' THEN DO
   HIT = 'Y'
   LEAVE ; END
ELSE DO
   ITERATE ; END
```

```
END
RETURN

/*********************************************************************/
/***  CLEAR PANEL VARIABLES (PTFØ1)                                ***/
/*********************************************************************/
CLRLST:
DATE = ''     ; COMP = ''   ; S = ''      ; VREF = ''
AUTHOR = ''   ; VEND = ''   ; SRCH = ''

CLRLST2:

Z1=' ' ;  Z2=' ' ;  Z3=' '  ;  Z4=' '  ;  Z5=' '  ;  Z6=' '  ; Z7=' '
Z8=' ' ;  Z9=' ' ;  Z1Ø=' ' ;  Z11=' ' ;  Z12=' ' ;  Z13=' ' ; Z14=' '
Z15=' ';  Z16=' ';  Z17=' ' ;  Z18=' ' ;  Z19=' ' ;  Z2Ø=' ' ; Z21=' '
Z22=' ';  Z23=' ';  Z24=' ' ;  Z25=' ' ;  Z26=' ' ;  Z27=' ' ; Z28=' '
N1=' ' ;  N2=' ' ;  N3=' '  ;  N4=' '  ;  N5=' '  ;  N6=' '  ; N7=' '
N8=' ' ;  N9=' ' ;  N1Ø=' ' ;  N11=' ' ;  N12=' ' ;  N13=' ' ; N14=' '
I1=' ' ;  I2=' ' ;  I3=' '  ;  I4=' '  ;  I5=' '  ;  I6=' '  ; I7=' '
I8=' ' ;  I9=' ' ;  I1Ø=' ' ;  I11=' ' ;  I12=' ' ;  I13=' ' ; I14=' '
A1=' ' ;  A2=' ' ;  A3=' '  ;  A4=' '  ;  A5=' '  ;  A6=' '  ; A7=' '
A8=' ' ;  A9=' ' ;  A1Ø=' ' ;  A11=' ' ;  A12=' ' ;  A13=' ' ; A14=' '
D1=' ' ;  D2=' ' ;  D3=' '  ;  D4=' '  ;  D5=' '  ;  D6=' '  ; D7=' '
D8=' ' ;  D9=' ' ;  D1Ø=' ' ;  D11=' ' ;  D12=' ' ;  D13=' ' ; D14=' '

RETURN

/*********************************************************************/
/***  DETERMINE NUMBER OF LINES IN THE DIRECTORY                   ***/
/*********************************************************************/
GETLNS:

 SET EMSG OFF
'ACC' INCADDR INCFM                    /* RE-ACCESS INCIDENT DISK  */
 SET EMSG ON
ADDRESS COMMAND 'STATE' DIRFN DIRFT INCFM

IF RC ¬= Ø THEN DO
   LNS  = Ø ; WLNS = Ø
  RETURN ; END
ELSE DO
  ADDRESS COMMAND 'LISTFILE' DIRFN DIRFT INCFM '(STACK DATE'
  PARSE PULL ENTRY                     /*   THE DIRECTORY       */
  PARSE VAR ENTRY . . DIRFM . . LNS .
  DESBUF
  WLNS = LNS
  RETURN ; END

/*********************************************************************/
/***  UPDATE DIRECTORY INFORMATION                                 ***/
```

```
/*********************************************************************/
UPDDIR:
IF ACCSW='R/O' THEN RETURN
WLNS2 = SUBSTR(INCID,5,4)

ADDRESS COMMAND 'EXECIO 1 DISKR' DIRFN DIRFT INCFM WLNS2 '(VAR ENTRY'
RETCD=RC
IF RETCD ¬= Ø THEN SIGNAL CIOERR            /* OTHER ERROR       */

PARSE VAR ENTRY INCIDENT .

IF INCIDENT ¬= INCID THEN DO
  MESSAGE = 'Internal mismatch Ø1. Contact Tech Support.'
  RETURN ; END

WORK1 = SUBSTR(ENTRY,1,9) || S || SUBSTR(ENTRY,11,28)
WORK2 = UDATE   || ' ' || TIME() || ' ' || SUBSTR(USERID(),1,8)
WORK3 = ' ' || SUBSTR(COMP,1,8) || ' ' || SUBSTR(VEND,1,8)
WORK4 = ' ' || SUBSTR(VREF,1,8) || ' ' || V
WORK5 = ' ' || SUBSTR(DESC1,1,5Ø) || SUBSTR(DESC2,1,5Ø)
WORK6 = WORK1 || WORK2 || WORK3 || WORK4 || WORK5

ADDRESS COMMAND 'EXECIO 1 DISKW' DIRFN DIRFT INCFM WLNS2 '(STRING' WORK6
IF RC¬=Ø THEN CALL DIRERR
RETURN

/*********************************************************************/
/***  SET OR UPDATE INCIDENT CONTROL INFO                          ***/
/*********************************************************************/
SETCNTL:

IF ACCSW='R/O' THEN RETURN
HIT = 'N'
DO UNTIL HIT = 'Y'
PFRET = '' ; VALID = 'Y' ; CALL DISPØ3
IF CPFKEY = 'PFØ4' THEN SIGNAL EXIT
IF CPFKEY ¬= 'PFØ3' THEN DO
   NOP ; END
ELSE
   IF PSCREEN = 'PTFØ1' THEN DO
      PFRET = 'Y'
      RETURN ; END
   ELSE DO
      CALL DISPØØ
      SIGNAL CKRESP ; END
 IF CPFKEY = 'PFØ7' THEN DO
   INCNBR = 1
   PAGENBR = 1
   LSCREEN = 'PTFØ7'
   P7TYPE = '17'
```

```
   CALL FULLIST3
   CALL SAVEZ
   CALL CKSETFØ7
   CALL REPLACEZ
   END

IF CPFKEY = 'PFØ8' THEN DO
   INCNBR = 1
   PAGENBR = 1
   LSCREEN = 'PTFØ7'
   P7TYPE = '18'
   CALL FULLIST3
   CALL SAVEZ
   CALL CKSETFØ7
   CALL REPLACEZ
   END

IF CPFKEY = 'PF1Ø' & P63 ¬= '' THEN DO
   CALL EDPRINT
   MESSAGE = INCID 'sent to' P63'...'
   ITERATE ; END
IF CPFKEY ¬= ' ' THEN ITERATE
IF DESC1 = '' & DESC2 = '' THEN DO
   VALID = 'N'
   MESSAGE = P35 'must be entered' ; END
ELSE DO
   DESC1 = SUBSTR(DESC1,1,5Ø,' ')
   DESC2 = SUBSTR(DESC2,1,5Ø,' ') ; END

IF COMP = '' | VEND = '' THEN DO
   VALID = 'N'
   MESSAGE = P34 'and' P36 'must be entered' ; END
ELSE DO
   COMP = SPACE(COMP,Ø)    ; VEND = SPACE(VEND,Ø)
   COMP = SUBSTR(COMP,1,8) ; VEND = SUBSTR(VEND,1,8) ; END
IF VREF = '' THEN DO
   VREF = 'ØØØØØØØØ' ; END
ELSE DO
   VREF = SPACE(VREF,Ø) ; VREF = SUBSTR(VREF,1,8) ; END

IF V = '' THEN DO
   V = '4' ; END
ELSE
   IF V = '1' | V = '2' | V = '3' | V = '4' THEN DO
      NOP ; END
   ELSE DO
      VALID = 'N'
      MESSAGE = 'Invalid' P38'.' ; END
IF S = 'O' | S = 'C' THEN DO
   NOP ; END
```

```
ELSE DO
   VALID = 'N'
   MESSAGE = 'Status must be O for open or C for closed' ; END
IF VALID = 'Y' THEN DO
   HIT = 'Y' ; END
ELSE DO
   ITERATE ; END
END
RETURN

/*********************************************************************/
/***  SAVE Z VARIABLES IN CASE FULLIST PANEL WILL BE RE-DISPLAYED ***/
/*********************************************************************/
   SAVEZ:
   W.1  = Z.1  ;  W.5  = Z.5  ;  W.9  = Z.3   ;  W.13 = Z.13
   W.2  = Z.2  ;  W.6  = Z.6  ;  W.10 = Z.10  ;  W.14 = Z.14
   W.3  = Z.3  ;  W.7  = Z.7  ;  W.11 = Z.11
   W.4  = Z.4  ;  W.8  = Z.8  ;  W.12 = Z.12
   RETURN

/*********************************************************************/
/***  REPLACE Z VARIABLES                                          ***/
/*********************************************************************/
   REPLACEZ:

   Z1  = W.1  ;  Z5  = W.5  ;  Z9  = W.3   ;  Z13 = W.13
   Z2  = W.2  ;  Z6  = W.6  ;  Z10 = W.10  ;  Z14 = W.14
   Z3  = W.3  ;  Z7  = W.7  ;  Z11 = W.11
   Z4  = W.4  ;  Z8  = W.8  ;  Z12 = W.12
   RETURN

/*********************************************************************/
/***  EDIT RESPONSES FROM PTF07 PANEL WHILE IN SETCNTL             ***/
/*********************************************************************/
CKSETF07:

DO UNTIL CPFKEY = 'PF03'
IF CPFKEY = 'PF04' THEN SIGNAL EXIT
IF CPFKEY = 'PF07' & PAGENBR > 1 THEN DO
   PAGENBR = PAGENBR-1
   INCNBR = (PAGENBR * 14) - 13
   CALL FULLIST3
   ITERATE ; END

IF CPFKEY = 'PF08' & PAGENBR < 7 THEN DO
   INCNBR = INCNBR+1
   PAGENBR = PAGENBR+1
   CALL FULLIST3
   ITERATE ; END
```

```
IF CPFKEY = 'PFØ7' THEN DO
   MESSAGE = 'Already at the beginning...'
   CALL DISPØ7 ; ITERATE ; END

IF CPFKEY = 'PFØ8' THEN DO
   MESSAGE = 'No more entries to display...'
   CALL DISPØ7 ; ITERATE ; END

CALL DISPØ7
END
RETURN

/*********************************************************************/
/***  SCAN DESCRIPTION FOR SEARCH WORDS                            ***/
/*********************************************************************/
SEARCH:

CTR = WORDS(SRCH)
IF CTR = Ø THEN
   RETURN

PARSE VAR SRCH W.1 W.2 W.3 W.4 W.5 W.6 W.7 W.8 W.9 W.1Ø W.11 W.12

SUB   = 1
DWORK = SUBSTR(ENTRY,95,1ØØ)
DWORK = TRANSLATE(DWORK)

DO CTR
   SWORD = TRANSLATE(W.SUB)
   IF POS(SWORD,DWORK) = Ø THEN
      VALID = 'N'
   ELSE
      SUB = SUB+1
END
IF VALID = 'N' THEN NOP
   ELSE RETURN

/*********************************************************************/
/***  SCAN ACTUAL MEMBER FOR SEARCH WORDS...                       ***/
/*********************************************************************/

SUB = Ø
VALID = 'Y'
DO CTR
  SUB = SUB + 1
  ADDRESS COMMAND 'EXECIO * DISKR' INCIDENT INCFT INCFM '1 (LOCATE
\'W.SUB'\ SKIP NOTYPE'
  IF RC = Ø THEN ITERATE
  UPPER W.SUB
  ADDRESS COMMAND 'EXECIO * DISKR' INCIDENT INCFT INCFM '1 (LOCATE
```

```
\'W.SUB'\ SKIP NOTYPE'
  IF RC ¬= Ø THEN VALID = 'N'
  IF RC ¬= Ø THEN LEAVE
END
RETURN

/*********************************************************************/
/***  GENERATE CONTROL MEMBER FOR XEDIT-VIEWING OR PRINTING...   ***/
/*********************************************************************/
EDPRINT:
IF ACCSW = 'R/W' THEN DO
   EDPFM   = INCFM
   EDPADDR = INCADDR ; END             /* USE INCIDENT DISK        */
ELSE DO
   EDPFM   = 'A1'
   EDPADDR = '191' ; END               /* USE USERS A DISK         */
PWORK='1'
 CALL EDPRINT2
PWORK=' '
 CALL EDPRINT2
PWORK=' 2'SUBSTR(P33,1,21,' ')INCID
 CALL EDPRINT2
PWORK=' '
 CALL EDPRINT2
PWORK=' 2'SUBSTR(P34,1,17,' ')'==> 'SUBSTR(COMP,1,8)
 CALL EDPRINT2
PWORK=' '
 CALL EDPRINT2
PWORK=' 2'SUBSTR(P35,1,17,' ')'==> 'DESC1
 CALL EDPRINT2
PWORK=' 2                 ==> 'DESC2
 CALL EDPRINT2
PWORK=' '
 CALL EDPRINT2
PWORK=' 2'SUBSTR(P36,1,17,' ')'==> 'SUBSTR(VEND,1,8) ' Created
'SUBSTR(CDATE,1,8)'  'SUBSTR(CTIME,1,8)'  by  'SUBSTR(CUSER,1,8)
 CALL EDPRINT2
PWORK=' '
 CALL EDPRINT2
PWORK=' 2'SUBSTR(P37,1,17,' ')'==> 'SUBSTR(VREF,1,8) ' Last Update
'SUBSTR(UDATE,1,8)'  'SUBSTR(UTIME,1,8)'  by  'SUBSTR(UUSER,1,8)
 CALL EDPRINT2
PWORK=' '
 CALL EDPRINT2
PWORK=' 2'SUBSTR(P38,1,17,' ')'==> 'V'         Status   ==> 'S
 CALL EDPRINT2
PWORK=' '
 CALL EDPRINT2

 SET IMSG OFF ; SET EMSG OFF
```

```
'FINIS' INCID EDPRTFT EDPFM
'ACC' EDPADDR EDPFM                    /* RE-ACCESS DISK           */
 IF EDCTRL = 'Y' THEN DO
    SET IMSG ON ; SET EMSG ON; RETURN ; END
 ELSE DO
    NOP ; END

ADDRESS COMMAND 'COPYFILE' INCID INCFT INCFM INCID EDPRTFT EDPFM
'(APPEND'
'SP PRINTER CLASS A FORM STANDARD COPY' P67 'CHARS NULL'
'SPOOL PRINTER CLASS U DEST' P63 'COPY' P67 'CHARS ST12'
 IF P66 = 'DUPLEX' THEN
    EXT = 'NORMAL'
 ELSE
    EXT = ''

'PSF' INCID EDPRTFT EDPFM '(CC' P66 EXT 'FORMDEF (F1A1Ø11Ø FDEF38PP)
PAGEDEF ('LPI' PDEF38PP) TRC DATACK BLKC)'
 ADDRESS COMMAND 'ERASE' INCID EDPRTFT EDPFM
 SET IMSG ON
 RETURN
 EDPRINT2:
ADDRESS COMMAND 'EXECIO 1 DISKW' INCID EDPRTFT EDPFM 'Ø (STRING' PWORK
 RETURN

/*********************************************************************/
/***  PROCESS DELETE REQUEST                                       ***/
/*********************************************************************/
 DELETE:
IF PWD2OK = 'Y' THEN
   NOP
ELSE DO
   PWDNULL = 'D' ; PC2 = '==>'
   PWDC3   = ' '   ; PWDC4   = ' '
   PC1     = ' ' ; PC2     = '==>' ; PC3     = ' ' ; PC4     = ' '
   CALL SETPWD
   ZCSR = 'Z1'
   IF PWD2OK = 'Y' THEN
      NOP
   ELSE DO
      MESSAGE = 'Delete request ignored due to invalid password.'
      RETURN ; END
   END

   D.1 = D1   ;   D.6  = D6   ;   D.11 = D11
   D.2 = D2   ;   D.7  = D7   ;   D.12 = D12
   D.3 = D3   ;   D.8  = D8   ;   D.13 = D13
   D.4 = D4   ;   D.9  = D9   ;   D.14 = D14
   D.5 = D5   ;   D.1Ø = D1Ø
```

```
   DLTDSC1 = D.ICTR          /* SET FIRST LINE OF DESCRIPTION      */
   ZCTR = ZCTR+2             /* LOOK-AHEAD AT NEXT ENTRY TO SEE IF */
   ICTR = ICTR+1             /* THIS IS A 1 OR 2-LINE DESCRIPTION  */
   IF Z.ZCTR = '' THEN
      DLTDSC2 = D.ICTR
   ELSE
      DLTDSC2 = ''

 DELETE2:
   DO FOREVER
      CALL DISPØ4
      IF CPFKEY = 'PFØ6' THEN LEAVE
      IF CPFKEY = 'PFØ3' THEN RETURN
      IF CPFKEY = 'PFØ4' THEN SIGNAL EXIT
   END

   SET IMSG OFF ; ADDRESS COMMAND 'ERASE' INCID INCFT INCFM
   SET IMSG ON ; SVS = S ; S = 'D' ; CALL UPDDIR ; S = SVS
 RETURN

/*********************************************************************/
/***  SET-UP TO GET DIRECTORY ENTRY, THEN "PARSE" ITEMS FURTHER  ***/
/*********************************************************************/
GETPARSE:
  WORK = 'Y'
  WLNS = SUBSTR(INCID,5,4)
  CALL GETDIR
  WORK = ''

  COMP  = COMPID                  ; CDATE = OPENDATE
  VEND  = VENDID                  ; CTIME = OPENTIME
  VREF  = VREFID                  ; CUSER = OPENUSER
  V     = SEV                     ; UDATE = LUPDDATE
  DESC1 = SUBSTR(ENTRY,95,5Ø)     ; UTIME = LUPDTIME
  DESC2 = SUBSTR(ENTRY,145,5Ø)    ; UUSER = LUPDUSER
  S     = STATUS
RETURN

/*********************************************************************/
/***  DISPLAY PANELS...                                          ***/
/*********************************************************************/
 DISPØØ:
   FULLIST = ''
   ACCOUNT = ''
   LSCREEN = 'PTFØØ'
   PSCREEN = 'PTFØØ'
   ADDRESS ISPEXEC 'DISPLAY PANEL(PTFØØ)'
   MESSAGE  = ''
 RETURN
```

```
DISPØ1:
  LSCREEN = 'PTFØ1'
  ADDRESS ISPEXEC 'DISPLAY PANEL(PTFØ1) CURSOR('ZCSR')'
  MESSAGE  = ''
RETURN

DISPØ2:
  LSCREEN = 'PTFØ2'
  ADDRESS ISPEXEC 'DISPLAY PANEL(PTFØ2)'
  MESSAGE  = ''
RETURN

DISPØ3:
  IF SVZCTR = 'X' THEN RETURN
  LSCREEN = 'PTFØ3'
  ADDRESS ISPEXEC 'DISPLAY PANEL(PTFØ3)'
  MESSAGE  = ''
RETURN

DISPØ4:
  LSCREEN = 'PTFØ4'
  ADDRESS ISPEXEC 'DISPLAY PANEL(PTFØ4)'
  MESSAGE  = ''
RETURN

DISPØ6:
    MESSAGE  = 'Overtype reversed fields and press ENTER...'
    ZCSR = 'P61'
  DO FOREVER
    PWDNULL = 'C'
    PWDC3   = ' '   ; PWDC4   = ' '
    PC1     = '==>' ; PC2    = ' ' ; PC3     = ' ' ; PC4     = ' '
    LSCREEN  = 'PTFØ6'
    ADDRESS ISPEXEC 'DISPLAY PANEL(PTFØ6) CURSOR('ZCSR')'
    MESSAGE  = ''
    IF CPFKEY = 'PFØ2' THEN CALL CONFIGØ6
    IF CPFKEY = 'PFØ3' | CPFKEY = ' ' THEN CALL UPDCFGØ6
    IF CPFKEY = 'PFØ4' THEN DO
       IF ACCSW = 'R/W' THEN
          CALL UPDCFGØ6
       ELSE
         MESSAGE = 'Permanent updates not allowed in READ/ONLY mode.'
    END
  ZCSR = 'P61'
    IF CPFKEY = 'PFØ5' THEN CALL DISP13
    IF CPFKEY = 'PFØ6' THEN CALL DISP1Ø
    IF CPFKEY = 'PFØ7' THEN CALL DISP11
    IF CPFKEY = 'PFØ8' THEN P7TYPE = '17'
    IF CPFKEY = 'PFØ8' THEN P7ID   = P34
    IF CPFKEY = 'PFØ8' THEN CALL DISP17
```

```
    IF CPFKEY = 'PFØ9' THEN P7TYPE = '18'
    IF CPFKEY = 'PFØ9' THEN P7ID   = P36
    IF CPFKEY = 'PFØ9' THEN CALL DISP17
    IF LSCREEN = 'PTFØ6' & MESSAGE = '' & CPFKEY='PFØ3' THEN LEAVE
    ITERATE
  END
P7TYPE=' '
RETURN

DISPØ7:
  IF P7TYPE = '17' THEN
     P7ID = P34
  ELSE
     P7ID = P36
  ADDRESS ISPEXEC 'DISPLAY PANEL(PTFØ7)'
  MESSAGE  = ''
RETURN

DISP1Ø:
  LSCREEN  = 'PTF1Ø'
  MESSAGE  = 'Overtype reversed fields and press ENTER...'
  ZCSR = 'PØ1'
  DO FOREVER
    ADDRESS ISPEXEC 'DISPLAY PANEL(PTF1Ø) CURSOR('ZCSR')'
    MESSAGE  = ''
    IF CPFKEY = 'PFØ2' THEN CALL CONFIGØØ
    IF CPFKEY = 'PFØ3' THEN LEAVE
    IF CPFKEY = 'PFØ4' THEN DO
       IF ACCSW = 'R/W' THEN
          CALL UPDCFGØØ
       ELSE
         MESSAGE = 'Permanent updates not allowed in READ/ONLY mode.'
    END
  ZCSR = 'PØ1'
  END
  ZCSR='P61'
  CALL SETHDR
RETURN
```

Editor's note: this article will be continued next month.

Steve Bernard
Senior Systems Programmer (USA)

Transferring code from the Web to a mainframe

Editor's note: although this article was written by an MVS Update *subscriber, and the solution is MVS-specific, a similar macro could be written for XEDIT.*

When a colleague of mine recently downloaded an *MVS Update* article from the Xephon Web site to his PC and then uploaded it to his MVS system, he found to his disappointment that the program code would not run properly.

It was a REXX program, and, when he executed it, he received the following message:

```
IRX0013I Error running XXXXXXXX, line nn: Invalid character in program
```

This was rather puzzling, but a quick look at the code revealed that the offending character was a REXX 'not' (that is ^, in a ^= expression), which should be a hex value X'5F', but was instead a X'B0'. The REXX interpreter was rejecting this value. Another odd character turned out to be the '|' operator, which should be X'4F', but was X'6A'.

Having discovered this, it was easy to code an ISPF edit macro to fix this and to cater for it in future uploads:

```
ISREDIT MACRO
ISREDIT CHANGE ALL X'B0' X'5F'
ISREDIT CHANGE ALL X'6A' X'4F'
EXIT
```

The PC was running IBM Personal Communications 3270 Version 4.1 for Windows with an IEEE 802.2 connection to the host, code page 037. The upload was achieved using the IBM 3270 PC File Transfer Program for MVS/TSO Release 1.1.1 using the following command:

```
IND$FILE PUT XEPHFILE.TEXT ASCII CRLF RECFM(V) LRECL(133)
```

It seems that the ASCII to EBCDIC conversion taking place works fine for alphanumeric characters, but is suspect for unusual ones. Readers should be aware of this when transferring code.

Patrick Mullen
MVS Systems Consultant (Canada)

Fast copying of fixed length files

GENERAL DESCRIPTION

Copying large files can take up a sizeable part of database development and maintenance time. FCOPY reduces the time needed for file processing and, because database support uses a lot of processor power, optimizes the overall system load.

FCOPY is written in Assembler and runs under CMS with VM/SP Release 5.

MEMORY REQUIREMENTS

The size of FCOPY is 2093 bytes. At execution time, a buffer area size of 1024KB is allocated. For successful execution, the VM size must be at least 3MB.

FCOPY USAGE

FCOPY primarily accelerates the copying of fixed-length files, but it can also process variable-length files. FCOPY is invoked as shown below:

```
FCOPY <source> <target> [(<options>]
```

Where:

- 'Source' is the filename (FN), filetape (FT), and filemode (FM) of the source file.
- 'Target' is the FN, FT, and FM of the target file. To replace the source FN, FT, or FM, the place holder '=' may be used.
- 'Options' refers to the following supported keywords:
  - 'Fn' specifies that the target file is to be written with fixed-record format and a record size of n bytes.
  - 'Vn' specifies that the target file is to be written with variable-record format and a record size of n bytes.

– ‘P[os] i’ specifies that only part of each source file record, starting from byte i, is to be written to the target file.

– ‘FR[om] p’ specifies that part of the source file, beginning at record p, is to be written to the target file.

– ‘FO[r] q’ specifies that only q records are to be copied to the target file.

– ‘R[ep]’ specifies that the existing target file is to be replaced.

– ‘A[pp]’ specifies that copied records must be added to the end of the existing file.

When parsing ‘options’, the following substitutions are made:

- ‘Vn’ replaces ‘Fn’.
- ‘A[pp]’ replaces ‘R[ep]’.

If a fixed record format file is copied to a variable format file, then trailing blanks are removed from each written record. If ‘P[os] i’ is specified for the variable record file, then all records having a length less than ‘i’ will not be copied to the target file.

Examples of FCOPY usage are:

- Copying file FN FT A to disk B:

```
FCOPY FN FT A = = B
```

- Appending 1,000 records, beginning from the 10,000th record of a source file, to an existing file:

```
FCOPY FN FT A FN_OLD FT_OLD A (A FR 10000 FO 1000
```

- Replacing an existing file with a new one containing a field length of 10 bytes, starting from the 7th byte of the first 12 records of the source file:

```
FCOPY FN FT A FN_OLD FT_OLD A (R F 10 P 7 FR 1 FO 12
```

PERFORMANCE OF FCOPY

The performance of FCOPY can be compared to that of CMS COPY by looking at the results obtained when copying fixed-length files,

ranging in size from 1MB to 14.5MB, to/from 3380 devices, with a 3MB per second transfer rate. The number of copied records varies between 33,825 and 2,170,336. The elapsed time was determined using the REXX function TIME with R and E options and the VTIME and TTIME were obtained from the CMS READY message. The results are shown in Figure 1.

Size (MB)	1	9	14.5
Records	33,825	943,713	2,170,336
Record length (bytes)	31	10	7
Elapsed time			
FCOPY	1.52	13.29	23.17
CMS COPY	9.21	142.25	327.68
CP VTIME			
FCOPY	0.03	0.19	0.30
CMS COPY	3.83	105.02	245.33
CP TTIME			
FCOPY	0.15	0.82	1.25
CMS COPY	4.13	107.70	249.57
Rate (MB/min)			
FCOPY	39.47	40.63	37.53
CMS COPY	6.51	3.79	2.65

Figure 1: Comparison of FCOPY and CMS COPY

FCOPYPRO EXEC

```
/*****************************************************************/
/***                                                  ***      ***/
/*** INSTALL       generate FCOPY MODULE              ***      ***/
/***                                                  ***      ***/
/*****************************************************************/
/***   SIZE ØØØ43  VER 1.Ø MOD ØØØ                              ***/
/*****************************************************************/

  CLRSCRN
  MESSAGE = 'user request'
  SAY ' — Start FCOPY MODULE generation - reply Y or N'
  PULL REPLY
  IF REPLY ¬= 'Y' THEN
  SIGNAL ERROR
  SET CMSTYPE HT
  STATE FCOPY MODULE A
  SAVE_RC = RC
  SET CMSTYPE RT
  IF SAVE_RC = Ø THEN
  DO
    SAY ' — FCOPY MODULE found on disk A'
    SAY ' — Replace FCOPY MODULE A - reply Y or N'
    PULL REPLY
    IF REPLY ¬= 'Y' THEN
    SIGNAL ERROR
  END
  SET CMSTYPE HT
  SIGNAL ON ERROR
  MESSAGE = 'error when assemble' FCOPY
  ASSEMBLE FCOPY
  ERASE FCOPY LISTING A
  MESSAGE = 'error when load' FCOPY
  LOAD FCOPY '(' NOMAP NOLIBE
  MESSAGE = 'error when genmod' FCOPY
  GENMOD
  ERASE FCOPY TEXT A
  SIGNAL OFF ERROR
  SET CMSTYPE RT
  SAY ' — FCOPY MODULE generated successfully'
  EXIT
ERROR:
  SET CMSTYPE RT
  SAY ' — FCOPY MODULE not generated due to' MESSAGE

       - FCOPY ASSEMBLE

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

```
**** FCOPY              CMS fast copyfile             ***         ****
****                                                  ***         ****
************************************************************************
****   SIZE 00423  VER 1.0 MOD 000                                  ****
************************************************************************
*                                                                      *
FCOPY    CSECT
         USING *,12
         ST    14,EXIT
         MVC   INDCB+8(18),8(1)
         MVC   OUTDCB+8(18),32(1)
         CLI   OUTDCB+8,C'='
         BNE   CHECKFT
         MVC   OUTDCB+8(8),INDCB+8
CHECKFT  EQU   *
         CLI   OUTDCB+16,C'='
         BNE   CHECKFM
         MVC   OUTDCB+16(8),INDCB+16
CHECKFM  EQU   *
         CLI   OUTDCB+24,C'='
         BNE   OKOUTDCB
         MVC   OUTDCB+24(2),INDCB+24
OKOUTDCB EQU   *
         CLC   INDCB+8(18),OUTDCB+8
         BE    MSG01
         LA    3,64(1)
         LA    4,SETINS
         LA    5,DIGITS
         LA    10,KEYLIST
         LA    14,8
         LA    15,ENDLIST
NEXTKEY  EQU   *
         LR    6,3
CHKPARM  EQU   *
         CLI   0(6),X'FF'
         BE    ITERATE
         CLC   0(4,6),0(10)
         BE    THATS
         CLC   0(4,6),4(10)
THATS    EQU   *
         LA    6,8(6)
         BNE   CHKPARM
SETVALUE EQU   *
         MVC   0(1,4),0(10)
         CLI   0(10),C'R'
         BE    ITERATE
         CLI   0(10),C'A'
         BE    ITERATE
         SR    1,1
         TRT   0(8,6),0(5)
```

```
         LTR   1,1
         BZ    MSGØ2
         SR    1,6
         LTR   1,1
         BZ    MSGØ2
         BCTR  1,Ø
         EX    1,PACK
         CVB   1,DOUBLE
         ST    1,4(4)
ITERATE  EQU   *
         LA    4,8(4)
         BXLE  1Ø,14,NEXTKEY
CONFIG   EQU   *
         FSSTATE FSCB=OUTDCB,FORM=E
         LTR   15,15
         BNZ   OPENWNEW
         CLI   APPEND,X'4Ø'
         BNE   OPENWMOD
         CLI   WRITOVER,X'4Ø'
         BE    MSGØ3
         FSERASE FSCB=OUTDCB
         B     OPENWNEW
OPENWMOD EQU   *
         LA    15,1
         A     15,48(1)
         ST    15,OUTDCB+X'2C'
         CLI   3Ø(1),C'V'
         BNE   SETFCONV
         MVI   VFORMAT,C'V'
         MVC   VLEN(4),32(1)
         B     OPENWNEW
SETFCONV EQU   *
         MVI   FFORMAT,C'F'
         MVC   FLEN(4),32(1)
OPENWNEW EQU   *
         FSOPEN FSCB=INDCB,FORM=E,ERROR=MSGØ4
         LA    9,1Ø24
         SLL   9,7
         LR    Ø,9
         DMSFREE DWORDS=(Ø),TYPE=USER,ERR=MSGØ5
         STM   Ø,1,RGØ1
         SLL   Ø,3
         LA    2,2
         SLL   2,15
         SR    Ø,2
         LR    3,Ø
         AR    Ø,1
         ST    Ø,OUTDCB+X'1C'
         ST    1,INDCB+X'1C'
         MVC   OUTDCB+X'24'(1),INDCB+X'24'
```

```
         MVC   OUTDCB+X'20'(4),INDCB+X'20'
         CLI   VFORMAT,X'40'
         BE    CHECKF
         MVI   OUTDCB+X'24',C'V'
         MVC   OUTDCB+X'20'(4),VLEN
         B     CHECKOFF
CHECKF   EQU   *
         CLI   FFORMAT,X'40'
         BE    CHECKOFF
         MVI   OUTDCB+X'24',C'F'
         MVC   OUTDCB+X'20'(4),FLEN
CHECKOFF EQU   *
         L     11,INDCB+X'20'
         L     10,OUTDCB+X'20'
         LR    9,11
         CLI   OFFSET,X'40'
         BE    CHECKMOV
         L     15,OFFSETV
         BCTR  15,0
         ST    15,OFFSETV
         SR    9,15
         BM    MSG06
         B     MOVE
CHECKMOV EQU   *
         CLI   INDCB+X'24',C'V'
         BE    DONTMOVE
         CLI   OUTDCB+X'24',C'V'
         BE    DONTMOVE
         CR    10,9
         BNH   DONTMOVE
MOVE     EQU   *
         MVI   DOUBLE+1,C'X'
DONTMOVE EQU   *
         CLI   START,X'40'
         BE    CHKCOUNT
         MVC   INDCB+X'2C'(4),STARTV
CHKCOUNT EQU   *
         L     8,=A(1024*1024*1024)
         CLI   COUNT,X'40'
         BE    CHECKSRC
         L     8,COUNTV
CHECKSRC EQU   *
         CLI   INDCB+X'24',C'V'
         BE    VSOURCE
         SR    2,2
         DR    2,11
         ST    3,INDCB+X'30'
         SR    2,2
         MR    2,11
         ST    3,INDCB+X'20'
```

```
         CLI   OUTDCB+X'24',C'F'
         BNE   SETLRECL
         CR    1Ø,11
         BNE   COPYF
         CLI   DOUBLE+1,C'X'
         BE    COPYF
         MVC   OUTDCB+X'3Ø',INDCB+X'3Ø'
         MVC   OUTDCB+X'2Ø',INDCB+X'2Ø'
         ST    1,OUTDCB+X'1C'
         MVI   DOUBLE,C'X'
         B     COPYF
SETLRECL EQU   *
         CR    11,1Ø
         BH    COPYF
         LR    1Ø,11
         ST    1Ø,OUTDCB+X'2Ø'
COPYF    EQU   *
         C     8,INDCB+X'3Ø'
         BH    READALL
         ST    8,INDCB+X'3Ø'
READALL  EQU   *
         FSREAD FSCB=INDCB,FORM=E
         LTR   15,15
         BNZ   FREEMAIN
         LTR   8,8
         BNP   FREEMAIN
XC1      XC    INDCB+X'2C'(4),INDCB+X'2C'
         MVC   XC1(12),=6X'Ø7ØØ'
         C     Ø,OUTDCB+X'2Ø'
         BE    CHECKDBL
         LR    3,Ø
         SR    2,2
         DR    2,11
         ST    3,INDCB+X'3Ø'
         SR    2,2
         MR    2,11
         ST    3,INDCB+X'2Ø'
CHECKDBL EQU   *
         CLI   DOUBLE,C'X'
         BNE   WRITEF
         MVC   OUTDCB+X'3Ø'(4),INDCB+X'3Ø'
         MVC   OUTDCB+X'2Ø'(4),INDCB+X'2Ø'
         LA    3,1
         B     ONLWRITE
WRITEF   EQU   *
         L     3,INDCB+X'3Ø'
         L     4,INDCB+X'1C'
         A     4,OFFSETV
VCYCF    EQU   *
         CLI   DOUBLE+1,C'X'
```

```
         BE    MOVENOWF
         ST    4,OUTDCB+X'1C'
         LR    14,4
         AR    14,1Ø
         BCTR  14,Ø
         IC    15,Ø(4)
         MVI   Ø(4),X'FF'
TRYCUT   EQU   *
         CLI   Ø(14),X'4Ø'
         BNE   CUTHERE
         BCT   14,TRYCUT
CUTHERE  EQU   *
         STC   15,Ø(4)
         SR    14,4
         LA    14,1(14)
         ST    14,OUTDCB+X'2Ø'
         B     ONLWRITE
MOVENOWF EQU   *
         L     Ø,OUTDCB+X'1C'
         LR    1,1Ø
         LR    14,4
         LR    15,9
         ICM   15,8,=X'4Ø'
         MVCL  Ø,14
ONLWRITE EQU   *
         FSWRITE FSCB=OUTDCB,FORM=E,ERROR=MSGØ7
XC2      XC    OUTDCB+X'2C'(4),OUTDCB+X'2C'
         MVC   XC2(12),=6X'Ø7ØØ'
         AR    4,11
         BCT   3,VCYCF
         S     8,INDCB+X'3Ø'
         LTR   8,8
         BZ    FREEMAIN
         B     COPYF
VSOURCE  EQU   *
         ST    1,OUTDCB+X'1C'
         ST    3,OUTDCB+X'2Ø'
         CLI   OUTDCB+X'24',C'F'
         BNE   COPYV
         SR    2,2
         DR    2,1Ø
         ST    3,OUTDCB+X'3Ø'
         MVI   DOUBLE,C'X'
COPYV    EQU   *
         L     4,RGØ1+4
         SR    5,5
         LA    6,2
         LR    7,3
         AR    7,4
         SR    7,6
```

```
         CLI   DOUBLE,C'X'
         BE    READV
         LA    4,2(4)
         ST    4,INDCB+X'1C'
READV    EQU   *
         FSREAD FSCB=INDCB,FORM=E
         LTR   15,15
         BNZ   SETEOF
XC3      XC    INDCB+X'2C'(4),INDCB+X'2C'
         MVC   XC3(12),=6X'Ø7ØØ'
         BCTR  8,Ø
         LTR   8,8
         BM    SETEOF
         LA    5,1(5)
         CLI   DOUBLE,C'X'
         BNE   FILLBUF
         LR    14,Ø
         SR    Ø,1Ø
         BNM   CHECKREC
         LPR   1,Ø
         SR    15,15
         ICM   15,8,=X'4Ø'
         AR    14,4
         LR    Ø,14
         MVCL  Ø,14
CHECKREC EQU   *
         AR    4,1Ø
         ST    4,INDCB+X'1C'
         C     5,OUTDCB+X'3Ø'
         BE    WRITEV
         B     READV
SETEOF   EQU   *
         SR    8,8
         B     WRITEV
FILLBUF  EQU   *
         SR    4,6
         STCM  Ø,3,Ø(4)
         AR    4,Ø
         LA    4,4(4)
         ST    4,INDCB+X'1C'
         CR    4,7
         BL    READV
WRITEV   EQU   *
         CLI   DOUBLE,C'X'
         BNE   WRITECV
         ST    5,OUTDCB+X'3Ø'
         SR    4,4
         MR    4,1Ø
         ST    5,OUTDCB+X'2Ø'
         LA    5,1
```

```
         B     WRITEONL
WRITECV  EQU *
         L     4,RGØ1+4
         L     6,OFFSETV
         SR    7,7
WRITEVV  EQU   *
         ICM   7,3,Ø(4)
         LR    15,7
         SR    15,6
         LA    4,2(4)
         LTR   15,15
         BNP   CANTWRIT
         CR    15,1Ø
         BNH   STORE
         LR    15,1Ø
STORE    EQU   *
         ST    15,OUTDCB+X'2Ø'
         LA    1,Ø(4,6)
         ST    1,OUTDCB+X'1C'
WRITEONL EQU   *
         FSWRITE FSCB=OUTDCB,FORM=E,ERROR=MSGØ7
XC4      XC    OUTDCB+X'2C'(4),OUTDCB+X'2C'
         MVC   XC4(12),=6X'Ø7ØØ'
CANTWRIT EQU   *
         LA    4,Ø(4,7)
         BCT   5,WRITEVV
         LTR   8,8
         BZ    FREEMAIN
         B     COPYV
ERROR    EQU   *
         LINEDIT TEXTA=FCOPYMSG,DOT=NO,COMP=NO
         MVI   RC+3,X'Ø1'
GOOUT    B     RET
         B     FREEMEM
FREEMAIN EQU   *
         XC    RC(4),RC
FREEMEM  EQU   *
         LM    Ø,1,RGØ1
         DMSFRET DWORDS=(Ø),LOC=(1)
RET      EQU   *
         FSCLOSE FSCB=INDCB
         FSCLOSE FSCB=OUTDCB
         L     14,EXIT
         L     15,RC
         BR    14
MSGØ1    EQU   *
         MVC   MESSAGE(MLEN),MSGØ1TXT
         B     ERROR
MSGØ2    EQU   *
         MVC   MESSAGE(MLEN),MSGØ2TXT
```

```
         B     ERROR
MSGØ3    EQU   *
         MVC   MESSAGE(MLEN),MSGØ3TXT
         B     ERROR
MSGØ4    EQU   *
         MVC   MESSAGE(MLEN),MSGØ4TXT
         B     ERROR
MSGØ5    EQU   *
         MVC   MESSAGE(MLEN),MSGØ5TXT
FREESTOR EQU   *
         MVC   GOOUT(4),=2X'Ø7ØØ'
         B     ERROR
MSGØ6    EQU   *
         MVC   MESSAGE(MLEN),MSGØ6TXT
         B     FREESTOR
MSGØ7    EQU   *
         MVC   MESSAGE(MLEN),MSGØ7TXT
         B     FREESTOR
PACK     PACK  DOUBLE(8),Ø(Ø,6)
EXIT     DS    F
INDCB    FSCB  FORM=E
OUTDCB   FSCB  FORM=E
DOUBLE   DS    D
KEYLIST  DC    CL4'F'
         DC    CL4'F'
         DC    CL4'V'
         DC    CL4'V'
         DC    CL4'POS'
         DC    CL4'P'
         DC    CL4'FROM'
         DC    CL4'FR'
         DC    CL4'FOR'
         DC    CL4'FO'
         DC    CL4'REP'
         DC    CL4'R'
         DC    CL4'APP'
         DC    CL4'A'
ENDLIST  EQU   *-4
SETINS   DS    ØF
FFORMAT  DC    4X'4Ø'
FLEN     DC    4X'ØØ'
VFORMAT  DC    4X'4Ø'
VLEN     DC    4X'ØØ'
OFFSET   DC    4X'4Ø'
OFFSETV  DC    4X'ØØ'
START    DC    4X'4Ø'
STARTV   DC    4X'ØØ'
COUNT    DC    4X'4Ø'
COUNTV   DC    4X'ØØ'
WRITOVER DC    4X'4Ø'
```

```
PAD      DS    4X
APPEND   DC    4X'40'
DIGITS   DC    240X'FF'
         DC    10X'00'
         DC    6X'FF'
FCOPYMSG DC    AL1(FULLEN)
FCOPYID  DC    C'FCOPY message -> '
MESSAGE  DC    C'                 '
MLEN     EQU   *-MESSAGE
FULLEN   EQU   *-FCOPYID
MSG01TXT DC    C'Source = Target  '
MSG02TXT DC    C'Invalid parameter'
MSG03TXT DC    C'Target found     '
MSG04TXT DC    C'Source not found '
MSG05TXT DC    C'No free storage  '
MSG06TXT DC    C'Invalid position '
MSG07TXT DC    C'Target disk full '
         ORG   DIGITS
RG01     DS    2F
RC       DS    F
         END   FCOPY
```

FCOPY PREPARATION

INSTALL EXEC should be used to generate the executable code. FCOPY may then be invoked as an ordinary CMS command.

Dobrin Goranov
Information Services Co (Bulgaria)

The article *Transferring code from the Web to a mainframe*, published on page 28 of this issue of *VM Update*, highlights a modification to overcome problems experienced when downloading *Update* code to a mainframe. A detailed discussion of many of the problems associated with sharing mainframe code can be found on the Xephon Web site in the article *Sharing mainframe code* (http://www.xephon.com/contnote.html/).

Melinda Varian's homepage

Continuing the series of VM Web site reviews, we visit Melinda Varian's homepage, which can be accessed at http://pucc.princeton.edu:80/~melinda/.

I've known Melinda Varian for about two decades – she's in the category of friend/colleague whose initial introduction I can't remember. Since her Web page was on a colleague's list of candidates for review in this publication, I was pleased to have an excuse to visit for the first time.

Melinda has participated vigorously in much of VM's significant history, applying VM at Princeton University, exploiting it and evangelizing on its behalf, and helping IBM understand its value to the user community, the computing industry, and IBM itself. She was one of the stalwart warriors during the source code battles of the 1980s, during which IBM planned and attempted to remove one of VM's greatest strengths – the availability of the operating system's source code, which allowed customers and Independent Software Vendors (ISVs) to exploit, enhance, customize, and debug VM in ways unavailable on most platforms – and sadly lacking in today's 'modern' operating systems such as Windows and OS/2. (Unix, of course, shares the tradition of source code availability, and has similarly benefitted, although some vendor-specific Unix versions do not include source code.) Early in VM's life, Melinda authored a definitive work on installing, tailoring, enhancing, and supporting VM, called *What mother never told you about VM maintenance*. Since then, she's been a tireless and far-ranging representative of VM and its technologies, adopting and speaking on them; she's spoken on REXX in the 1980s and CMS Pipelines and the P/390 technology more recently. One of her epic works is entitled *VM and the VM community: past, present, and future*. Published as a presentation script, the material's effect on long-time and new VMers is usually profound, reminding the former of memorable events and beloved colleagues, and telling the latter the story of how VM came to be the way it is, and how present VM users and developers are standing on the shoulders of giants.

Alongside a photograph of Melinda addressing a user group conference session, there is a view of a very complex PC screen with multiple open windows, entitled babybear.jpeg, illustrating a P/390 chugging along processing multiple applications. P/390 resources are discussed below. Just below the pictures are links to versions of the *Community paper*, in various formats. Downloading and reading this paper is well worth the effort, although reading the paper is not as satisfying as hearing Melinda present it, as she occasionally does at user groups.

Below the links to the *Community paper* is an illustration of a VM logo enjoying a party – a New Year's party. But this isn't just any new year being welcomed, it's 1 January 2000. This graphic links to a page offering a countdown clock, as do many other Year 2000-related Web sites. This clock measures the time remaining in weeks, and at the time of writing there are ninety-four. Somehow that seems more alarming than counts of seconds, minutes, hours, or days, which are always too large to be meaningful, logistically or emotionally. Ninety-four weeks means ninety-four weekends! This means there is not too much time remaining, no matter where one is in planning and implementing date remediation projects. This page, described as *VM and Year 2000 at Princeton*, in fact opens more inclusively: *"The systems software on Princeton's IBM mainframes, both MVS and VM, is in the process of being made Year 2000-compliant. This page is intended to guide users in preparing their applications to run properly in the new millennium. It will be updated frequently as more is learned."*

The page is well organized, with the initial table of contents jumping to sections below. *Other Year 2000 Web pages* offers links to a small number of valuable resources worth visiting for major vendor (IBM and Microsoft) and independent views and resources:

- *Current table of Year 2000 issues and solutions for Microsoft products.*
- *Other PC-related Year 2000 information, including BIOS fixes.*
- *IBM's Year 2000 page for VM.*
- *IBM's primary Year 2000 page.*
- *The Year 2000 information centre.*
- *Gartner Group Year 2000 page.*

- *Management information on the Year 2000 computer problem.*

The Year 2000 page's main content is VM considerations related to accommodating four-digit years, and is relevant for all VM sites, with the caveat that it describes Princeton's environment, including a mix of local system enhancements and installed ISV products. It's a good template for disseminating Year 2000 information to users within any large organization, after customizing it to address local variations and requirements. Information here illustrates why a seemingly simple change – adding century digits to years stored, processed, displayed, and entered – can have obscure, pervasive, and time-consuming consequences. It's a good technical overview, and good reading material for any manager puzzled by the scope of the problem or resources required to solve it.

The VM considerations begin with a description of the basic date change and new VM date-testing facilities: *"Both CP and CMS have been enhanced to display four-digit years. Two formats are available for four-digit years, 'isodate' (yyyy-mm-dd) and 'fulldate' (mm/dd/yyyy). The default date format is unchanged and is referred to as 'shortdate' (mm/dd/yy).*

New facilities have been added to allow users to set the time and date in their own virtual machines. This will facilitate testing, such as for crossing the millennial boundary and for making sure that their applications understand that 2000 is a leap year.

We cannot emphasize too strongly that care should be taken when altering the date in one's virtual machine. In particular, timestamps on files may be set to the future, which may confuse some applications, and expiration dates may appear to have passed."

The page then links to a general reference on making VM applications Year 2000-compliant, IBM redbook publication *VM/ESA Year 2000 migration – a case study*. This publication opens by introducing the Year 2000 problem, discusses VM's Year 2000 support (including the effects of not migrating to a Year 2000-ready release), describes a migration case study, identifies application enabling program products, addresses REXX-based application issues, suggests how to locate date-related code, covers Language Environment and PL/I, outlines

Year 2000 testing procedures, and includes sample date-related routines.

Melinda's Year 2000 page tours CMS and CP commands, diagnose codes, REXX and EXEC 2 enhancements, and enhanced date formats and conversions. There's also an experimental CMS Pipelines filter that front-ends the date conversion function of the DateTimeSubtract CSL routine and can be used to convert between a variety of date formats.

Anyone involved in Year 2000 efforts has probably heard the term 'time machine' applied to a platform used for testing operating systems, software packages, and locally written applications at artificially set dates – past or future. Just as VM provided and epitomized personal computing and client/server applications long before those terms were coined, let alone trendy, VM now offers easy time travel in the comfort of individual time machines – what we've called 'virtual machines' for decades – without requiring dedicated resources or even LPAR manipulation. As the page notes, CP allows directory-authorized virtual machines to set their own TOD clocks for localized time travel.

Although CMS does not set the TOD clock, Princeton uses an add-on commercial software product, VM Timing Facility Monitors from MiraSoft, which adds several new commands that give programmers new capabilities for Year 2000 testing: the ability to detect use of system date/time calls, and to set a different date/time for virtual machines. When many organizations are buying extra mainframes for Year 2000 testing, or counting weekends available for dedicated testing on production systems, it's heartening that VM's ability to virtualize nearly everything still increases efficiency and reduces cost of ownership, even when challenged by problems such as a new millennium.

The Year 2000 page concludes by identifying and describing local Princeton and IBM supplied tools for handling some Year 2000 support chores, such as locating specific, potentially date-related, character strings for investigation and change. The next section describes Year 2000-compliant compilers for CMS, including PL/I, REXX, C, and COBOL.

The next graphic on Melinda's main page shows a venerable VM/370 logo, displayed in green-on-black 3270 format. This flanks links to two IBM VM-related Web pages: the main VM page described in *VM Update,* Issue 139, March 1998, and IBM's corporate main Web site.

The dancing bear graphic is a Sandra Hassenplug illustration drawn for the SHARE user group's VM organization. SHARE, with member organizations around the world, draws several thousand attendees to its twice-yearly meetings. The most recent meeting was in Anaheim, California, and the next meeting will be in Washington DC in August. This links to a Web page that briefly describes SHARE's VM activities, identifies advantages of user group participation, outlines the VM activities held in Anaheim, and invites taking part – attending, speaking, and volunteering – in future SHARE meetings. The page also identifies SHARE volunteers responsible for different VM areas such as CMS, performance and capacity planning, and systems management. A very important aspect of SHARE is the interaction it provides with various level of IBM staff – development, marketing, and management. Three IBM representatives are identified who have responsibilities for IBM/SHARE coordination. Though not linked from this page, SHARE's main page (http:www.share.org) provides abundant information on the organization's resources, procedures, structure, membership, and forthcoming meetings.

Melinda's next logo/link identifies the New York area's Metropolitan VM Users Association (MVMUA), described as: *"a non-profit organization whose purpose is to promote the use of the VM operating system and the education of the computer professionals who support it. It is also the oldest organized VM user group in the country, having been founded in 1974. The Association serves the New York/New Jersey/Philadelphia metropolitan area (roughly). Members also come from as far north as Boston and as far west as Millersville, PA."*

Though MVMUA is the senior group amongst local VM user groups, many other such groups exist around the United States – and around the world. A growing list of these resources with links to their Web sites is available at http://www.vmers.org:81/. No list of VM community activities would be complete without discussing the annual VM Workshop, next in the sequence of graphics/links (the Workshop logo

lettered in balloons, symbolic of the Workshop spirit and a traditional activity there) on Melinda's page. As most of the people who have attended Workshops can confirm, each event is unique – for example, there's nothing like midnight runs to the baker in Manhattan, KS, for seconds-from-the-oven doughnuts, or watching for the green sunset flash in Asilomar, CA.

The Workshop Web page gives the meetings' flavour, and encourages joining this year's gathering on 9-13 June at Marist College, Poughkeepsie, NY: *"The VM Workshop is an annual event which is essentially an 'immersion program for systems programmers'. It is 3.5 days of intensive discussion and presentations about the VM operating system and related subjects. The Workshop began in 1977 as an alternative to the more formal (and expensive) conferences such as SHARE and Guide. Over the years, it has remained a completely grass-roots effort, with its organization being handled by volunteers from various colleges, universities, and corporations. Its emphasis is on participation and the exchanging of ideas and information.*

During the day, attendees participate in discussion groups on various subjects as well as see presentations given by colleagues. Formal presentations from IBM and other experienced speakers are featured. However, informal, 'this is a neat trick I learned' presentations are also encouraged. During the evenings, dinner and activities are planned to encourage discussion and information exchanges in a more casual environment. Spouses are also welcome at the evening activities."

The 'VMW3' logo links to several dozen Web sites powered by VM – demonstrating that the mainframe, and VM, are indeed well positioned as enterprise servers, bringing VM flexibility and power to the World Wide Web, along with traditional economies of scale and robust, reliable operation. The sites listed from around the world represent government, academic, industrial, and non-profit organizations, using a variety of tools and products for bringing back-end business information and function to the front door of the Internet. The list concludes with several links to VM/Web related resources, to enable other sites to exploit the Web with their VM systems.

The next two pages, listed under *CMS Pipelines*, constitute a

masterwork compilation worthy of detailed browsing. The first link, *CMS/TSO Pipelines runtime library distribution*, is a treasure trove of papers, software, presentations, tips, tricks, tools, anecdotes, personalities, pictures, and more, related to exploring and exploiting (and enjoying!) John Hartmann's software bonanza, which is called Pipelines or CMS Pipelines for short. Originally released as obscure software categorized as a PRPQ (don't ask what that means), Pipelines has been integrated in VM for some time, and has also been ported to MVS. In brief, and as a grossly inadequate over-simplification, Pipelines offers a set of tools and building blocks for creating applications. Pipelines replaces many frequently used traditional programming practices and paradigms, offering instead the model of data flowing between processing stages. Pipelines users (self-designated as 'plumbers') find their applications simpler, easier to develop and maintain, and operationally much faster, than similar software developed with old-style utilities and programming languages as scripting glue. The diversity of information available is clear from section titles such as:

- *CMS Pipelines runtime library distribution.*
- *Distinguishing levels of Pipelines.*
- *Tools for use with the runtime distribution.*
- *The Plumbers' Workbench: CMS Pipelines and workstation synergy.*
- *BatchPipeWorks: Pipelines for TSO.*
- *Papers on CMS/TSO Pipelines.*
- *Papers by Melinda Varian.*
- *Papers by John Hartmann.*
- *Papers by Nick Laflamme.*
- *Papers by Rob van der Heij.*
- *MTREXX.*
- *HEXSORT XEDIT macro.*

- *VM development tools.*
- *Skeleton REXX Pipeline stages.*
- *Examples of using co-pipes and encoded Pipeline specifications.*
- *VPIPE, vertical Pipelines.*
- *PIPESERV, Pipeline server facility.*
- *VSAM read Pipeline stage.*
- *Rita, the Pipeline execution profiler.*
- *Tools for use in conjunction with the CMS/TSO Pipelines TCP and UDP support.*
- *PIPEDEMO, the Pipeline animator.*
- *Other Pipeline debugging tools.*
- *Pipeline stages for encoding and decoding.*
- *SHARE requirements for enhancements to CMS/TSO Pipelines.*
- *Other sources of CMS/TSO Pipelines information.*

The other Pipelines page, hosted at the Department for Medical Computersciences, General Hospital, Vienna, Austria, opens cheerfully with the notation that '*We're plumb crazy*', and a picture of the ebullient John Hartmann, the master plumber himself. This page provides similar but different plumbing resources. Among the links and information on these two Pipelines pages are resources for every level of plumber – novice to advanced – to learn more techniques, adopt more tools, and develop new powerful applications.

The next section, devoted to REXX, links to several elegant and eloquent areas, including Mike Cowlishaw's REXX page, discussed in *VM Update*, Issue 140, April 1997. Next, we see a cheerful, stylized circuit board, tagged *Personal/370 and Personal/390*. This heads three links to P/390-related Web pages, which make interesting reading for VM users, system programmers, data centre managers, and software developers. During the first 15 years or so of VM's life, it was always a system programmers dream to have a free-standing single-user workstation on which to run VM. Several IBM attempts

at this were interesting but didn't quite meet real-world needs, because they weren't truly VM-based, or their performance was inadequate, or they were marketing orphans. The P/390, follow-on to the P/370, is available in several implementations from IBM and several flavours of IBM business partners.

The first P/390 page, with links to technical and marketing information, begins (with RS/6000 and Unix references added): *"The P/390 is a co-processor for PCI bus PCs, such as the IBM PC Server product line. It adds a standalone ESA/390 processor function to a workstation running OS/2 or an RS/6000 running AIX (IBM's Unix). This creates a S/390 workstation that can run ESA mainframe applications concurrently with OS/2 or Unix applications. The P/390 is the engine featured in the following products:*

- *IBM PC Server System/390*
- *RS/6000 System and System/390 Server on Board*

The presence of both 'PC' and 'mainframe' operating systems in the same workstation allows the customer the freedom to choose the best platform on which to run each application. The P/390 truly provides the 'best of both worlds'.

The P/390 can run any ESA/390, S/370, or S/370-XA operating system and associated applications stand-alone. When installed in a fast, server-class workstation, the P/390 can support multiple users as a small departmental ESA/390 server. When installed in a smaller machine, the P/390 becomes the ideal single-user ESA/390 workstation."

The next section, *TigerNet, Princeton Online Alumni/ae Services*, links to a page which notes *"there are now 12,000 Princetonians registered in the TigerNet Directory"*. Since Princeton has used VM systems for quite some time, it's safe to assume that many of those graduates were helped and shaped by VM technology, whether they knew it or not. Melinda commented that the Princetonians listed constitute 15% of living alumni, and that she is *"quite pleased that so many of them have used this VM-based Web service. (The oldest one ever to have used it was 96 years old at the time.)"*

Near the end of this tour is a section devoted to *ADSM, the ADSTAR Distributed Storage Manager* with two links, one to IBM's ADSM page and the other to Princeton's. IBM's page reveals, below a number of resource and testimonial links, what ADSM is and offers: *"The award-winning ADSM family of software products is a comprehensive, enterprise-wide solution integrating unattended network back-up and archive with storage management and powerful disaster recovery.*

ADSM is also part of the IBM 3466 Network Storage Manager, an integrated 'drop-in' hardware and software solution providing data back-up, disaster recovery, and storage management for networked enterprises."

Princeton has for quite some time relied on ADSM for diversified back-up services for networked systems, shown by their links to information on using ADSM to back up Macintosh, Sun, Windows 3.1, Windows 95, and Windows NT systems, and general workstations and servers, totalling 5,200 clients. Melinda measures the system's success in the number of books and theses saved each year, which she calls frighteningly high.

Penultimately, and not to be ignored, Melinda provides instructions and resources for downloading and installing ZORK for CMS, one of the all-time great traditional cave/adventure/exploration games. At the bottom, there's a list of pages of eclectic interest – birding, astronomy, origami, art, and an IBM e-mail reference page, Melinda's e-mail address, and a *Powered by S/390* logo, which is available for use on S/390-hosted Web pages.

Editor's note: if you have comments on the Web sites reviewed in this series, or suggestions for relevant sites to review, please feel free to contact the author at gabe@acm.org or Xephon at any of the addresses shown on page 2.

Gabe Goldberg
Computers and Publishing (USA)

Safe Software has announced SafeSFS, a high performance security and administration solution for the VM/ESA Shared File System (SFS), which provides powerful rules and an intuitive user interface for administering SFS.

For further information contact:
Safe Sofware, 13486 Lake Shore Drive, Herndon, VA 20171, USA.
Tel: (703) 793 0777.
Jemasys, Ridgeway, Wargrave, Berkshire, RG10 8AS, UK.
Tel: (01189) 404878.

* * *

Beyond Software has announced the release of Version 1.1 of its EnterpriseWeb Secure/ VM high-performance, secure enterprise Web Server for VM.

Version 1.1 allows the Web-enabling of legacy applications using advanced security technology. The new version also comes with session identifier (ID) cacheing features, for improved performance.

EnterpriseWeb Secure offers Secure Sockets Layer (SSL) encryption, using the SecureWeb Toolkit from Spyrus. Transactions between the server and the browser have up to 168-bit encryption.

For further information contact:
Beyond Software, 1040 East Brokaw Road, San Jose, CA 95181, USA.
Tel: (408) 436 5900.
Jemasys, Ridgeway, Wargrave, Berkshire, RG10 8AS, UK.
Tel: (01189) 404878.

* * *

For OfficeVision/VM users wishing to migrate to Lotus Notes, IBM has announced software to provide a Lotus Notes Windows 95 and Windows NT client for OfficeVision/ VM. This includes support for mail and calendar services, so aiding transition to Lotus Notes.

For further information contact your local IBM representative.