129 VM

May 1997

update

In this issue

VM Update

Published by
Xephon
27-35 London Road
Newbury
Berkshire RG14 1JL
England
Telephone: 01635 38342
From USA: 01144 1635 38342
E-mail: xephon@compuserve.com

North American office
Xephon/QNA
1301 West Highway 407, Suite 201-405
Lewisville, TX 75067
USA
Telephone: 817 455 7050

Australian office
Xephon/RSM
GPO Box 6258
Halifax Street
Adelaide, SA 5000
Australia
Telephone: 088 223 1391

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*.

Editor
Trevor Eddolls

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.

Subscriptions and back-issues
A year's subscription to *VM Update*, comprising twelve monthly issues, costs £170.00 in the UK; $255.00 in the USA and Canada; £176.00 in Europe; £182.00 in Australasia and Japan; and £180.50 elsewhere. In all cases the price includes postage. Individual issues, starting with the January 1990 issue, are available separately to subscribers for £14.50 ($21.50) each including postage.

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.

Printed in England.

Making REXX/CMS object-oriented

Although nowadays there are object-powered REXX products available, CMS itself does not offer any object goodies. If a project consisting of several EXECs involves, say, the definition of the fields in a flat file, every EXEC working with this file must contain the same definitions. If one day the layout of the file gets changed, all the EXECs are affected and have to be altered appropriately.

However, this is not necessary. There is a way to keep the definitions located in one place and export them 'outside'. The method is to use the Pipelines stage command VARLOAD, which allows variables to be loaded into the environment of the EXEC from which the current EXEC was called. Let me remind you what the command looks like:

```
PIPE REXXVARS | VARLOAD 1
```

The above command loads all the variables available in the current procedure into the environment of the procedure from which the current one was called, ie the calling procedure, one level higher than the current one ('1' after VARLOAD points out this one level).

This way, different objects may be exported. There could be constants as well as parts of a statement that need to be interpreted in the outer procedure. For instance, it could be the template for parsing the string read from a file. Consider the following example. The project uses a

Columns	Contents
1-5	Sequence number
6-20	Name
21-40	Surname
41	Gender

Figure 1: File structure

flat file containing structured lines describing the employees, as shown in Figure 1.

The exporting procedure (let's name it EMPLDEF EXEC) might look like this:

```
/**/
EmployeeGenderMale = 1
EmployeeGenderFemale=2
EmployeeTableTemplate = "1 EmployeeSeqNo ",
"6 EmployeeName ",
"21 EmployeeSurname ",
"41 EmployeeGender "
"PIPE REXXVARS | VARLOAD 1"
```

Now, here's a sample of a calling procedure:

```
call EmplDef
Do forever
'EXECIO 1 DISKR EMPL TABLE * (VAR RECORD'
Interpret "Parse var Record with' EmployeeTableTemplate
If EmployeeGender = EmployeeGenderMale
then ..
else ...
end
```

Note that in the calling EXEC we have no definitions for the variable (in fact, constant) EmployeeGenderMale: it gets imported from EmplDef, as well as the template to parse the source string. This method looks very simple, but its advantages make it very powerful. In production, you can keep all the definitions in one EXEC instead of having them repeated in every program unit of the project. The outer units now can work with named constants being imported from the single source. When the business people come out with an idea of expanding the name field to 40 bytes, there's only one place in the project to customize. Well, the advantages of object methods (to which all the above, of course, is only a first step) are well known, so let me stop here.

Vadim Rapp
Systems Officer
First Chicago NBD Corporation (USA)

Translating EXEC code to REXX automatically

These routines will translate EXEC and EXEC2 code to REXX code. Enter the command EXETOREX followed by the name of the EXEC you want converted to REXX. The converted output will have the filetype of EXETOREX.

EXETOREX DOC

```
exetorex ?
REXXNAME: EXETOREX

FUNCTION: INITIALLY TRANSLATE EXEC AND EXEC2 CODE TO REXX CODE.

BASICFMT: ENTER COMMAND FOLLOWED BY THE NAME OF THE EXEC YOU WANT
          CONVERTED TO REXX.  THE CONVERTED OUTPUT WILL HAVE THE
          FILETYPE OF EXETOREX.

HOWTORUN: EXETOREX &EXECNAM  < *REPLACE >

 &EXECNAM ENTER THE FILENAME OF THE EXEC OR EXEC2 PROGRAM TO
          CONVERT TO REXX.  THE OUTPUT FILENAME WILL BE
          "&EXECNAM EXETOREX A".
 *REPLACE ENTER THIS KEYWORD IF A CONVERTED VERSION ALREADY EXISTS.
          IT WILL ERASE THE PRIOR CONVERSION AND START FRESH.

EXAMPLES: EXETOREX MINIREST

NOTES...  CAN'T HANDLE COMPOUND VARIABLES PREFIXED WITH 2 & SIGNS.
          CAN'T HANDLE QUOTES AS DATA.
          NOTHING HOOKED TO HANDLE THE &RANGE NOR "&SUBRTN... OF"
```

EXETOREX EXEC

```
/* AN MVI EXEC */
PARSE UPPER ARG
ARGSTRING;DEBUG='';$X=FIND(TRANSLATE(ARGSTRING),'*DEBUG')
IF $X ¬= Ø THEN DO; ARGSTRING = DELWORD(ARGSTRING,$X,1); TRACE I
    DEBUG = '*DEBUG'; END
IF WORD(ARGSTRING,1) = '?' THEN SIGNAL DOC
REPLACE = Ø; X = FIND(ARGSTRING,'*REPLACE')
IF X ¬= Ø THEN DO
    ARGSTRING = DELWORD(ARGSTRING,X,1)
    REPLACE = 1
    END
```

```
BEGIN:
PARSE VAR ARGSTRING EN .
IF EN = '' THEN SIGNAL ERRØØ8   /* FILENAME IS REQUIRED */
'CMSQ STATE' EN 'EXEC'
IF RC ¬= Ø THEN SIGNAL ERRØ1Ø   /* INPUT EXEC FILENAME NOT FOUND */
'CMSQ STATE' EN 'EXETOREX A' DEBUG
IF RC = Ø THEN DO
    IF ¬REPLACE THEN SIGNAL ERRØ2Ø  /* ALREADY DONE */
    END
'EXECIO 1 DISKR' EN 'EXEC * (FINIS '
IF RC ¬= Ø THEN SIGNAL ERRØ3Ø   /* ERROR READING FIRST RECORD */
PULL LINE1
IF LEFT(WORD(LINE1,1),2) = '/*' THEN SIGNAL ERRØ4Ø /* IP IS REXX */
QUEUE 'EXETOREX *FILE' DEBUG
'XEDIT' EN 'EXEC'
'CMSQ ERASE * AUTOSAVE A'
EXIT ØØØ
DOC:
SAY 'REXXNAME: EXETOREX'
SAY
SAY 'FUNCTION: INITIALLY TRANSLATE EXEC AND EXEC2 CODE TO REXX CODE.'
SAY
SAY 'BASICFMT: ENTER COMMAND FOLLOWED BY THE NAME OF THE EXEC YOU WANT'
SAY '          CONVERTED TO REXX.  THE CONVERTED OUTPUT WILL HAVE THE'
SAY '          FILETYPE OF EXETOREX.'
SAY
SAY 'HOWTORUN: EXETOREX &EXECNAM  < *REPLACE >'
SAY
SAY ' &EXECNAM ENTER THE FILENAME OF THE EXEC OR EXEC2 PROGRAM TO'
SAY '          CONVERT TO REXX.  THE OUTPUT FILENAME WILL BE '
SAY '          "&EXECNAM EXETOREX A".'
SAY ' *REPLACE ENTER THIS KEYWORD IF A CONVERTED ',
                                            'VERSION ALREADY EXISTS.'
SAY '          IT WILL ERASE THE PRIOR CONVERSION AND START FRESH. '
SAY
SAY 'EXAMPLES: EXETOREX MINIREST'
SAY
SAY 'NOTES...  CAN''T HANDLE COMPOUND VARIABLES ',
                                          'PREFIXED WITH 2 & SIGNS.'
SAY '          CAN''T HANDLE QUOTES AS DATA.'
SAY '          NOTHING HOOKED TO HANDLE THE &RANGE NOR "&SUBRTN... OF"'
EXIT ØØØ
ERRØ1Ø:
SAY 'EXETOREX - INPUT EXEC NAMED ('EN') NOT FOUND. RUN CANCELLED.'
EXIT Ø1Ø
ERRØ2Ø:
SAY 'EXETOREX - INPUT EXEC NAMED ('EN') ALREADY CONVERTED, ',
                                                    'USE *REPLACE.'
EXIT Ø2Ø
ERRØ3Ø:
```

```
SAY 'EXETOREX - ERROR READING ('EN')''S FIRST RECORD, ',
                                                    'TELL TECH SUPPORT.'
EXIT Ø3Ø
```

EXETOREX SKELETON

```
/* AN MVI EXEC */
PARSE ARG ARGSTRING;DEBUG='';$X=FIND(TRANSLATE(ARGSTRING),'*DEBUG')
IF $X ¬= Ø THEN DO; ARGSTRING = DELWORD(ARGSTRING,$X,1); TRACE I
    DEBUG = '*DEBUG'; END
IF WORD(ARGSTRING,1) = '?' THEN SIGNAL DOC
BEGIN:
```

EXETOREX XEDIT

```
/* */
/* DO SOMETHING ABOUT THE "VTRACE    " REMOVAL REQUIREMENT. */
/* REMEMBER THAT &READ COMMANDS GO FLD FOR FLD,
                                          ADD A VZ TO END OF THEM */
ARG ARGSTRING; DEBUG = ''; $X = (FIND(ARGSTRING,'*DEBUG'))
IF $X ¬= Ø THEN DO; ARGSTRING = (DELWORD(ARGSTRING,$X,1)); TRACE I
    DEBUG = '*DEBUG'; END
FILESTCK = Ø; X = FIND(ARGSTRING,'*FILE')
IF X ¬= Ø THEN DO
    ARGSTRING = DELWORD(ARGSTRING,X,1)
    FILESTCK = 1
    END
IF (FIND(ARGSTRING,'?')) = Ø THEN SIGNAL BEGIN
DOC:
/*BEGTYPE
EDITNAME: EXETOREX

FUNCTION: INITIALLY TRANSLATE EXEC AND EXEC2 CODE TO REXX CODE.

BASICFMT: ENTER COMMAND WITHOUT ANY OPTIONS AFTER ENTERING EDIT MODE
          FOR THE EXEC FILE YOU WANT CONVERTED TO REXX.

NOTES...  CAN'T HANDLE COMPOUND VARIABLES.
          CAN'T HANDLE QUOTES AS DATA.
          NOTHING HOOKED TO HANDLE THE &RANGE NOR "&SUBRTN... OF"

ENDTYPE*/
'REXSAYIT EXETOREX XEDIT A /*BEGTYPE ENDTYPE*/' DEBUG
EXIT ØØØ
BEGIN:
'SET WRAP OFF'
/*
START WITH PARSE VALUE ARGSTRING WITH V1 V2 V3 ETC...
```

```
AND VINDEX = WORDS(ARGSTRING)
*/
'TRANSFER FN FT FM'
IF RC ¬= Ø THEN SIGNAL ERRØ1Ø /* ERROR READING FN/FT/FM OF INPUT FILE.*/
PULL FN FT FM
/* TRANSFER LINE / TO GET THE CURRENT LINE NUMBER */
/* EXTRACT /CURLINE/ / PUTS CURRENT LINE CONTENTS INTO CURLINE.3 */
'COMMAND MSGMODE OFF'
'COMMAND TOP'
'COMMAND FT EXETOREX'
'COMMAND FM A1'
'COMMAND C/ .&/ "."&/ * * '
'COMMAND TOP'
'COMMAND C/&CONTROL OFF    /TRACE(O)/ *  '
'COMMAND TOP'
'COMMAND C/&CONTROL OFF NOMSG    /TRACE(O)/ *  '
'COMMAND TOP'
'COMMAND TOP'
SKIPCNT = Ø
SKIPNEXT:
'COMMAND LOC /&SKIP /'
IF RC ¬= Ø THEN SIGNAL SKIPCHEK
'COMMAND EXTRACT /CURLINE/'
PARSE UPPER VAR CURLINE.3 SKIPTXT '&SKIP' SKIPAMT
IF SKIPAMT = '' THEN SKIPAMT = 1
IF DATATYPE(SKIPAMT) ¬= 'NUM' THEN SIGNAL SKIPNEXT
'COMMAND EXTRACT /LINE/'
SKIPBAS = LINE.1
SKIPCNT = SKIPCNT + 1
SKIPTAG = 'SKIP'RIGHT('ØØØ'||SKIPCNT,4)
SKIPLOC = SKIPAMT + SKIPBAS
'COMMAND REPLACE' SKIPTXT 'SIGNAL' SKIPTAG
SKIPTAGS.SKIPCNT = '*' SKIPTAG SKIPLOC
SIGNAL SKIPNEXT
SKIPCHEK:
IF SKIPCNT = Ø THEN SIGNAL SKIPEXIT
DO X = SKIPCNT BY -1 FOR SKIPCNT
   'COMMAND LOCATE :'WORD(SKIPTAGS.X,3)
   'COMMAND INPUT' WORD(SKIPTAGS.X,2)':'
   END
SKIPEXIT:
TYPELOGIC:
'COMMAND TOP'
TYPECNT = Ø
TYPENEXT:
TYPENUM = Ø
'COMMAND LOC /&BEGTYPE /'
IF RC ¬= Ø THEN SIGNAL TYPEEXIT
'COMMAND EXTRACT /CURLINE/'
PARSE UPPER VAR CURLINE.3 X '&BEGTYPE' TYPETXT
```

```
IF WORDS(X) > 2 THEN SIGNAL TYPENEXT
IF DATATYPE(STRIP(TYPETXT)) = 'NUM'
   THEN TYPELMT = STRIP(TYPETXT)
   ELSE IF WORDS(TYPETXT) = 1
   THEN TYPELMT = STRIP(TYPETXT)
   ELSE TYPELMT = ''
'COMMAND EXTRACT /LINE/'
TYPEBAS = LINE.1
TYPECNT = TYPECNT + 1
TYPESFX = RIGHT('ØØØ'||TYPECNT,4)
'COMMAND REPLACE' STRIP( X '/*BEG'TYPESFX TYPETXT,'L')
TYPESEQ = TYPEBAS
TYPELOOP:
TYPESEQ = TYPESEQ + 1
'COMMAND LOC :'TYPESEQ
IF RC ¬= Ø THEN SIGNAL TYPEZEND
'COMMAND EXTRACT /CURLINE/'
PARSE VAR CURLINE.3 TYPETXT
IF TYPELMT ¬= '' THEN SIGNAL TYPENLMT
IF TRANSLATE(LEFT(STRIP(TYPETXT),4)) ¬= '&END'
   THEN DO
    'COMMAND REPLACE ~'TYPETXT
    SIGNAL TYPELOOP
    END
IF TRANSLATE(LEFT(STRIP(TYPETXT),4)) = '&END'
   THEN DO
    'COMMAND REPLACE END'TYPESFX'*/'
    "COMMAND INPUT REXSAYIT" FN FT "*",
        "/*BEG"TYPESFX "END"TYPESFX"*/ DEBUG"
    SIGNAL TYPENEXT
    END
SIGNAL TYPENEXT
TYPENLMT:    /* THE BEGTYPE LIMIT LOGIC WAS INVOKED. */
TYPENUM = TYPENUM + 1
IF DATATYPE(TYPELMT) ¬= 'NUM'
   THEN IF WORD(TYPETXT,1) = TYPELMT THEN SIGNAL TYPEXLMT
'COMMAND REPLACE ~'TYPETXT
IF DATATYPE(TYPELMT) = 'NUM'
   THEN IF TYPENUM < TYPELMT
       THEN SIGNAL TYPELOOP
       ELSE NOP
   ELSE SIGNAL TYPELOOP
'COMMAND INPUT END'TYPESFX'*/'
"COMMAND INPUT REXSAYIT" FN FT "*",
    "/*BEG"TYPESFX "END"TYPESFX"*/ DEBUG"
SIGNAL TYPENEXT
TYPEXLMT:
'COMMAND REPLACE END'TYPESFX'*/'
"COMMAND INPUT REXSAYIT" FN FT "*",
    "/*BEG"TYPESFX "END"TYPESFX"*/ DEBUG"
```

```
SIGNAL TYPENEXT
TYPEZEND:
'COMMAND INPUT END'TYPESFX'*/'
"COMMAND INPUT REXSAYIT" FN FT "*",
    "/*BEG"TYPESFX "END"TYPESFX"*/ DEBUG"
TYPEEXIT:
STCKLOGIC:
'COMMAND TOP'
STCKNEXT:
STCKNUM = Ø
'COMMAND LOC /&BEGSTACK /'
IF RC ¬= Ø THEN SIGNAL STCKEXIT
'COMMAND EXTRACT /CURLINE/'
PARSE UPPER VAR CURLINE.3 X '&BEGSTACK' STCKTXT
IF WORDS(X) > 2 THEN SIGNAL STCKNEXT
IF DATATYPE(STRIP(STCKTXT)) = 'NUM'
   THEN STCKLMT = STRIP(STCKTXT)
   ELSE STCKLMT = ''
'COMMAND EXTRACT /LINE/'
STCKBAS = LINE.1
TYPECNT = TYPECNT + 1
STCKSFX = RIGHT('ØØØ'||TYPECNT,4)
'COMMAND REPLACE' STRIP( X '/*BEG'STCKSFX STCKTXT,'L')
STCKSEQ = STCKBAS
STCKLOOP:
STCKSEQ = STCKSEQ + 1
'COMMAND LOC :'STCKSEQ
IF RC ¬= Ø THEN SIGNAL STCKZEND
'COMMAND EXTRACT /CURLINE/'
PARSE VAR CURLINE.3 STCKTXT
IF STCKLMT ¬= '' THEN SIGNAL STCKNLMT
IF TRANSLATE(LEFT(STRIP(STCKTXT),4)) ¬= '&END'
   THEN DO
    'COMMAND REPLACE ~'STCKTXT
    SIGNAL STCKLOOP
    END
IF TRANSLATE(LEFT(STRIP(STCKTXT),4)) = '&END'
   THEN DO
    'COMMAND REPLACE END'STCKSFX'*/'
    "COMMAND INPUT REXSTKIT" FN FT "*",
        "/*BEG"STCKSFX "END"STCKSFX"*/ DEBUG"
    SIGNAL STCKNEXT
    END
SIGNAL STCKNEXT
STCKNLMT:    /* THE BEGSTCK LIMIT LOGIC WAS INVOKED. */
STCKNUM = STCKNUM + 1
'COMMAND REPLACE ~'STCKTXT
IF STCKNUM < STCKLMT THEN SIGNAL STCKLOOP
'COMMAND INPUT END'STCKSFX'*/'
"COMMAND INPUT REXSTKIT" FN FT "*",
```

```
    "/*BEG"STCKSFX "END"STCKSFX"*/ DEBUG"
SIGNAL STCKNEXT
STCKZEND:
'COMMAND INPUT END'STCKSFX'*/'
"COMMAND INPUT REXSTKIT" FN FT "*",
    "/*BEG"STCKSFX "END"STCKSFX"*/ DEBUG"
STCKEXIT:
IFLOGIC:
'COMMAND TOP'
IFCC = 'EQ = NE ¬= LT < NG LE ¬> <= GT > GE NL >= ¬<'
IFNEXT:
'COMMAND LOC /&IF /'
IF RC ¬= Ø THEN SIGNAL IFEXIT
'COMMAND EXTRACT /CURLINE/'
PARSE UPPER VAR CURLINE.3 IFTEXT '&GOTO' X
IF LEFT(STRIP(IFTEXT),1) = '*' THEN SIGNAL IFNEXT
IF LEFT(STRIP(IFTEXT),1) = '~' THEN SIGNAL IFNEXT
IF X ¬= '' THEN SIGNAL IFNEXT
MAX = Ø; MAC = ''
DO X = 1 FOR WORDS(IFCC)
    Z = LASTPOS(' 'WORD(IFCC,X)' ',IFTEXT)
    IF MAX < Z THEN DO; MAX = Z; MAC = WORD(IFCC,X); END
    END
IF MAX = Ø THEN SIGNAL IFNEXT
X = WORDS(SUBSTR(IFTEXT,1,MAX)) /* CALCULATE THE CC'S POSITION -1 */
IF X < 2 THEN SIGNAL IFNEXT
/* INSERT THE THEN CLAUSE &IF A GT B COMMAND */
IF MAC ¬= '=' & SUBWORD(IFTEXT,X+3) ¬= '' &,
IF WORD(SUBWORD(IFTEXT,X+3),1) ¬= 'THEN'
   THEN IF MAC ¬= '=' & SUBWORD(IFTEXT,X+3) ¬= ''
       THEN IFNEW = SUBWORD(IFTEXT,1,X+2) 'THEN' SUBWORD(IFTEXT,X+3)
       ELSE IFNEW = SUBWORD(IFTEXT,1,X-1) 'THEN' SUBWORD(IFTEXT,X)
'COMMAND REPLACE' IFNEW
SIGNAL IFNEXT
IFEXIT:
'COMMAND TOP'
LOOPNEXT:
'COMMAND LOC /&LOOP /'
IF RC ¬= Ø THEN SIGNAL LOOPEXIT
'COMMAND EXTRACT /CURLINE/'
PARSE VAR CURLINE.3 FRONT '&LOOP' LINES TIMES
IF DATATYPE(LINES) = 'NUM'
   THEN DO
    'COMMAND REPLACE' FRONT 'DO' TIMES
    'COMMAND NEXT' LINES
    'INPUT END'  /* SIGNAL END OF DO LOOP */
    END
   ELSE DO
    'COMMAND REPLACE' FRONT 'DO' TIMES
    'COMMAND FIND' LINES
```

```
    'INPUT END'  /* SIGNAL END OF DO LOOP */
    END
SIGNAL LOOPNEXT
LOOPEXIT:
'COMMAND TOP'
'COMMAND C/&/V/ * * '
'COMMAND TOP'
'COMMAND C/VHT /HT / * * '
'COMMAND TOP'
'COMMAND C/VRT /RT / * * '
'COMMAND TOP'
'COMMAND C/VIF /IF / * * '
'COMMAND TOP'
'COMMAND C/VDATE /DATE(U) / * * '
'COMMAND TOP'
'COMMAND C/VTIME /TIME() / * * '
'COMMAND TOP'
'COMMAND C/VTYPE /SAY / * * '
'COMMAND TOP'
'COMMAND C/VPRINT /SAY / * * '
'COMMAND TOP'
'COMMAND C/ VLITERAL OF / VALUE( / * * '
'COMMAND TOP'
'COMMAND C/ VSUBSTR OF / SUBSTR( / * * '
'COMMAND TOP'
'COMMAND C/ VSUBSTR / SUBSTR( / * * '
'COMMAND TOP'
'COMMAND C/ VPIECE OF / SUBSTR( / * * '
'COMMAND TOP'
'COMMAND C/ VPIECE / SUBSTR( / * * '
'COMMAND TOP'
'COMMAND C/ VTRIM OF / STRIP( / * * '
'COMMAND TOP'
'COMMAND C/ VTRIM / STRIP( / * * '
'COMMAND TOP'
'COMMAND C/ VWORD OF / WORD( / * * '
'COMMAND TOP'
'COMMAND C/ VWORD / WORD( / * * '
'COMMAND TOP'
'COMMAND C/ VLOCATION OF / POS( / * * '
'COMMAND TOP'
'COMMAND C/ VLOCATION / POS( / * * '
'COMMAND TOP'
'COMMAND C/ VMULTIPLICATION OF / MULT( / * * '
'COMMAND TOP'
'COMMAND C/ VMULT OF / MULT( / * * '
'COMMAND TOP'
'COMMAND C/ VDIVISION OF / DIV( / * * '
'COMMAND TOP'
'COMMAND C/ VDIV OF / DIV( / * * '
```

```
'COMMAND TOP'
'COMMAND C/ VRANGE OF / RANGE( / * * '
'COMMAND TOP'
'COMMAND C/ VTRANSLATION OF / TRANSLATE( / * * '
'COMMAND TOP'
'COMMAND C/ VTRANS OF / TRANSLATE( / * * '
'COMMAND TOP'
'COMMAND C/ VLENGTH OF / LENGTH( / * * '
'COMMAND TOP'
'COMMAND C/ VLENGTH / LENGTH( / * * '
'COMMAND TOP'
'COMMAND C/ VDATATYPE OF / DATATYPE( / * * '
'COMMAND TOP'
'COMMAND C/ VDATATYPE / DATATYPE( / * * '
'COMMAND TOP'
'COMMAND C/ VCONCATENATION OF / CONCAT( / * * '
'COMMAND TOP'
'COMMAND C/ VCONCAT OF / CONCAT( / * * '
'COMMAND TOP'
'COMMAND C/ VCONCAT / CONCAT( / * * '
'COMMAND TOP'
'COMMAND C/ VLEFT OF / LEFT( / * * '
'COMMAND TOP'
'COMMAND C/ VRIGHT OF / RIGHT( / * * '
'COMMAND TOP'
'COMMAND C/VEXIT /EXIT / * * '
/*'COMMAND TOP'
'COMMAND C/-ERR/ERR/ * * ' */
'COMMAND TOP'
'COMMAND C/ VGOTO / THEN SIGNAL / * * '
'COMMAND TOP'
'COMMAND C/VGOTO /SIGNAL / * * '
'COMMAND TOP'
'COMMAND C/ VRETCODE / RC / * * '
'COMMAND TOP'
'COMMAND C/VRETCODE = /RC = / * * '
'COMMAND TOP'
'COMMAND C/ VRC / RC / * * '
'COMMAND TOP'
'COMMAND C/VREAD VARS /PULL / * * '
'COMMAND TOP'
'COMMAND C/VREAD ARGS /PULL V1 V2 V3 V4 V5 V6 V7 V8 ',
                                                'V9 V1Ø V11 V12 / * *'
'COMMAND TOP'
'COMMAND C/ EQ / = / * * '
'COMMAND TOP'
'COMMAND C/ NE / ¬= / * * '
'COMMAND TOP'
'COMMAND C/ GT / > / * * '
'COMMAND TOP'
```

```
'COMMAND C/ LT / < / * * '
'COMMAND TOP'
'COMMAND C/ NG / ¬> / * * '
'COMMAND TOP'
'COMMAND C/ LE / ¬> / * * '
'COMMAND TOP'
'COMMAND C/ GE / >= / * * '
'COMMAND TOP'
'COMMAND C/ NL / >= / * * '
'COMMAND TOP'
"COMMAND C/ =                          / = ''                    / * "
'COMMAND TOP'
'COMMAND C/VSTACK FIFO /QUEUE / * * '
'COMMAND TOP'
'COMMAND C/VSTACK LIFO /PUSH / * * '
'COMMAND TOP'
'COMMAND C/VSTACK /QUEUE / * * '   /* THE DEFAULT STACK MODE IS FIFO */
'COMMAND TOP'
'COMMAND C/PUSH HT /SET CMSTYPE HT / * '
'COMMAND TOP'
'COMMAND C/PUSH RT /SET CMSTYPE RT / * '
'COMMAND TOP'
'COMMAND C/ VCONTINUE / / * *'
'COMMAND TOP'
'COMMAND C/CLRSCRN/'VMFCLEAR'/ * * '
'COMMAND TOP'
'COMMAND C/SENTRIES/'SENTRIES'/ * * '
'COMMAND TOP'
'ALL /SIGNAL -/ | /SIGNAL  -/ | /SIGNAL   -/'
'CMSQ ERASE EXETOREX TEMPTAGS A'
'CMSQ ERASE EXETOREX TEMPSIGS A'
'COMMAND PUT * EXETOREX TEMPSIGS A'
'COMMAND ZONE 1 1'
'ALL /-/'
'COMMAND PUT * EXETOREX TEMPTAGS A'
'COMMAND ZONE 1 2'
'ALL / -/'
'COMMAND PUT * EXETOREX TEMPTAGS A'
'COMMAND ZONE 1 3'
'ALL /  -/'
'COMMAND PUT * EXETOREX TEMPTAGS A'
'COMMAND ZONE 1 4'
'ALL /   -/'
'COMMAND PUT * EXETOREX TEMPTAGS A'
'COMMAND ZONE 1 5'
'ALL /    -/'
'COMMAND PUT * EXETOREX TEMPTAGS A'
'COMMAND ZONE 1 *'
'ALL'
'COMMAND TOP'
```

```
'COMMAND C/VCALL -/CALL / *'
'COMMAND TOP'
'COMMAND C/VCALL  -/CALL / *'
'COMMAND TOP'
'COMMAND C/VCALL   -/CALL / *'
'COMMAND TOP'
'COMMAND C/SIGNAL -/SIGNAL / *'
'COMMAND TOP'
'COMMAND C/SIGNAL  -/SIGNAL / *'
'COMMAND TOP'
'COMMAND C/SIGNAL -/SIGNAL / *'
'COMMAND TOP'
'COMMAND C/SIGNAL  -/SIGNAL / *'
'COMMAND TOP'
'COMMAND C/SIGNAL   -/SIGNAL / *'
'COMMAND TOP'
'COMMAND C/SIGNAL    -/SIGNAL / *'
'COMMAND TOP'
'COMMAND C/SIGNAL     -/SIGNAL / *'
/* TAG RENAME DO LOOP LOGIC */
TAGSLIST = ''
'CMSQ STATE EXETOREX TEMPSIGS A'
IF RC = Ø THEN 'EXECIO * DISKR EXETOREX TEMPSIGS A 1 (STEM EXETOREX.'
IF DATATYPE(EXETOREX.Ø) = 'NUM'
   THEN DO SEQ = 1 FOR EXETOREX.Ø
    PARSE VAR EXETOREX.SEQ . ' SIGNAL' TAG .
    IF TAG = '' THEN ITERATE
    IF FIND(TAGSLIST,TAG) = Ø  /* CHECK IF TAG ALREADY CONVERTED */
       THEN DO
        TAGSLIST = TAGSLIST TAG
        'COMMAND TOP'
        /* GENERATE 'COMMAND C/-TAGNAME /TAGNAME: / *' */
        'COMMAND C/'TAG' /'RIGHT(TAG,LENGTH(TAG)-1)': / *'
        END
       ELSE ITERATE SEQ
    END
'CMSQ STATE EXETOREX TEMPTAGS'
IF RC = Ø THEN 'EXECIO * DISKR EXETOREX TEMPTAGS A 1 (STEM EXETOREX.'
IF DATATYPE(EXETOREX.Ø) = 'NUM'
   THEN DO SEQ = 1 FOR EXETOREX.Ø
    PARSE VAR EXETOREX.SEQ TAG .; TAG = STRIP(TAG)
    IF LEFT(TAG,1) ¬= '-' THEN ITERATE
    IF FIND(TAGSLIST,TAG) = Ø  /* CHECK IF TAG ALREADY CONVERTED */
       THEN DO
        TAGSLIST = TAGSLIST TAG
        'COMMAND TOP'
        /* GENERATE 'COMMAND C/-TAGNAME /TAGNAME: / *' */
        'COMMAND C/'TAG' /'RIGHT(TAG,LENGTH(TAG)-1)': / *'
        END
       ELSE ITERATE SEQ
```

```
    END
/* WHAT TO PUT IN QUOTES
EVERY LINE THAT BEGINS WITH A VTAG AND = SIGN, COVER RIGHT SIDE.
EVERY LINE THAT BEGINS WITH A (PUSH QUEUE PULL RETURN SAY) HANDLE.
EVERY LINE THAT BEGINS WITH "IF" HANDLE UP TO THE "THEN" FIELD.
DO NOT PUT RT HT AND RC'S IN QUOTES.
THINGS THAT FITS NEITHER, GET PUT WITHIN QUOTES, EXCEPT FOR VTAGS.
*/
LITLOGIC:
'COMMAND TOP'
XCMD = 'PUSH QUEUE RETURN SAY'
KCMD = "DEBUG PULL SIGNAL /*BEGTYPE ENDTYPE*/ TRACE(O) TRACE TRACE(I)"
KCMD = KCMD ""
LCMD = '| '' " = < > ¬ V'  /* LEFT SEARCH SKIP LIT LGC VALS */
RCMD = '('    /* RIGHT SEARCH SKIP LIT LGC VALUE */
LSKP = 'DEBUG + - ) , CALL DO END SIGNAL RETURN THEN EXIT RC',
       'SUBSTR( STRIP( WORD( POS( LENGTH( DATATYPE( VALUE(',
       'LEFT( RIGHT( FIND( SPACE( DET @DETACH EXECL @EXECLOAD'
ABBR = 'FI FIL FILE FILED FILEDE @FILEDEF X XE XED @XEDIT',
       'COPY @COPYFILE DEF @DEFINE T TY TYP @TYPE NAMEF @NAMEFIND ',
       'L LI LIS LIST LISTF LISTFI @LISTFILE ORD @ORDER TA @TAG',
       'FILEL @FILELIST GL GLO GLOB @GLOBAL ID IDENT @IDENTIFY',
       'SUB SUBMIT @XSUBMIT'
ABBR = ABBR,
       'IN INCL @INCLUDE SM @SMSG PU PUN @PUNCH Q QU QUE QUER @QUERY',
       'REL @RELEASE RELO @RELOCATE REF REFR @REFRESH R REN @RENAME',
       'STA STAR @START ST STO STOR @STORE SYN @SYNONYM',
       'TRAN TRANS @TRANSFER ATT ATTA ATTAC @ATTACH A ASS @ASSEMBLE'
CMDS = 'SUBSTR( ~1 STRIP( ~TRIM WORD( ~WORD',
       'POS( ~1 MULT( ~* DIV( ~/ RANGE( TRANSLATE( ~TRAN',
       'LENGTH( DATATYPE( ~1 CONCAT( ~|| LEFT( ~1 RIGHT( ~1 VALUE( ~1 '
EXEC = 'EX EXE EXEC'
LITSEQ = Ø
LITNEXT:
LITSEQ = LITSEQ + 1
'COMMAND LOC :'LITSEQ
IF RC ¬= Ø THEN SIGNAL LITEXIT
'COMMAND EXTRACT /CURLINE/'
IF RC ¬= Ø THEN SIGNAL LITEXIT
/* DROP THE EXEC COMMAND PREFIXES... */
IF FIND(EXEC,WORD(CURLINE.3,1)) ¬= Ø
   THEN CURLINE.3 = SUBWORD(CURLINE.3,2)
PARSE UPPER VAR CURLINE.3 1 LITTEXT,
                          1 XVAR XEQ XBDY,
                          1 'IF' XIF 'THEN' XTHEN
IF LITTEXT = '' THEN SIGNAL LITNEXT
IF RIGHT(XVAR,1) = ':' THEN SIGNAL LITNEXT
IF LEFT(STRIP(XVAR),1) = '*'
   THEN DO
    LITNEW = '/'LEFT(STRIP(LITTEXT),77)'*/'
```

```
    'COMMAND REPLACE' LITNEW
    SIGNAL LITNEXT
    END
IF LEFT(STRIP(XVAR),1) = '~' THEN SIGNAL LITNEXT
IF FIND(KCMD,XVAR) ¬= Ø THEN SIGNAL LITNEXT
LITNEW = ''
/* IF POS('(',LITTEXT) ¬= Ø THEN TRACE I*/
Z = Ø; DO X = 1 FOR WORDS(CMDS)
    Y = FIND(LITTEXT,WORD(CMDS,X))
    IF Y < 3 THEN ITERATE X
    IF WORD(CMDS,X+1) = '~1'
       THEN DO
        LFT = SUBWORD(LITTEXT,1,Y-1)
        MDL = WORD(LITTEXT,Y)
        TXT = SUBWORD(LITTEXT,Y+1)
        RGT = ''; DO Z = 1 FOR WORDS(TXT)
            IF WORD(LITTEXT,Y) = 'SUBSTR(' & WORD(TXT,Z) = '*' &,
               Z = 3
               THEN LEAVE
            IF Z ¬= 1 THEN RGT = RGT ','
            RGT = RGT WORD(TXT,Z)
            END
        LITTEXT = LFT MDL RGT ')' /* WON'T BE LIT'D DO TO ( PFX */
        END
       ELSE IF WORD(CMDS,X+1) = '~*'
       THEN LITTEXT = SUBWORD(LITTEXT,1,Y-1),
                      SUBWORD(LITTEXT,Y+1,1) '*',
                      SUBWORD(LITTEXT,Y+2)
       ELSE IF WORD(CMDS,X+1) = '~/'
       THEN LITTEXT = SUBWORD(LITTEXT,1,Y-1),
                      SUBWORD(LITTEXT,Y+1,1) '/',
                      SUBWORD(LITTEXT,Y+2)
       ELSE IF WORD(CMDS,X+1) = '~||'
       THEN DO
        LFT = SUBWORD(LITTEXT,1,Y-1)
        TXT = SUBWORD(LITTEXT,Y+1)
        RGT = ''; DO Z = 1 FOR WORDS(TXT)
            IF Z ¬= 1 THEN RGT = RGT '||'
            RGT = RGT WORD(TXT,Z)
            END
        LITTEXT = LFT RGT
        END
       ELSE IF WORD(CMDS,X+1) = '~TRAN'
       THEN LITTEXT = SUBWORD(LITTEXT,1,Y),
                      SUBWORD(LITTEXT,Y+1,1) ',',
                      SUBWORD(LITTEXT,Y+3) ',',
                      SUBWORD(LITTEXT,Y+2,1) ')'
       ELSE IF WORD(CMDS,X+1) = '~TRIM'
       THEN LITTEXT = SUBWORD(LITTEXT,1,Y),
                      SUBWORD(LITTEXT,Y+1) ",'T')"
```

```
      ELSE IF WORD(CMDS,X+1) = '~WORD'
      THEN DO
       N = WORDS(SUBWORD(LITTEST,Y+1))
       LITTEXT = SUBWORD(LITTEXT,1,Y),
                 SUBWORD(LITTEXT,Y+1,N-1) ',',
                 SUBWORD(LITTEXT,Y+N) ')'
       END
      ELSE DO
       LITTEXT = LITTEXT ')'
       END
    Z = 1    /* BE CAREFUL WITH THIS Z, IT'S CHECKED WAY DOWN! */
    PARSE       VAR LITTEXT   1 LITTEXT,
                              1 XVAR XEQ XBDY,
                              1 'IF' XIF 'THEN' XTHEN
    LEAVE X
    END
X = Ø; IF XVAR = 'IF' THEN DO X = 1 FOR WORDS(XIF)
    IF FIND(LCMD,LEFT(WORD(XIF,X),1)) ¬= Ø THEN ITERATE
    IF FIND(RCMD,RIGHT(WORD(XIF,X),1)) ¬= Ø THEN ITERATE
    IF FIND(LSKP,WORD(XIF,X)) ¬= Ø THEN ITERATE
    IF X > 3 & WORD(XIF,X) = 'IF'
      THEN Y = '&'
      ELSE Y = "\"WORD(XIF,X)"\"
    XIF = SUBWORD(XIF,1,X-1) Y SUBWORD(XIF,X+1)
    END
Y = Ø; IF XVAR = 'IF' THEN DO X = 1 FOR WORDS(XTHEN)
    IF X = 1 & FIND(KCMD,WORD(XTHEN,X)) ¬= Ø THEN LEAVE
    IF X = 1 & FIND(XCMD,WORD(XTHEN,X)) ¬= Ø THEN ITERATE
    IF X = 1 & WORD(XTHEN,2) = '=' THEN ITERATE
    IF FIND(LCMD,LEFT(WORD(XTHEN,X),1)) ¬= Ø THEN ITERATE
    IF FIND(RCMD,RIGHT(WORD(XTHEN,X),1)) ¬= Ø THEN ITERATE
    IF FIND(LSKP,WORD(XTHEN,X)) ¬= Ø THEN ITERATE
    XTHEN = SUBWORD(XTHEN,1,X-1) "\"WORD(XTHEN,X)"\" SUBWORD(XTHEN,X+1)
    Y = 1
    END
IF X ¬= Ø | Y ¬= Ø THEN LITNEW = 'IF' XIF 'THEN' XTHEN
IF LITNEW ¬= '' THEN SIGNAL LITREP
XTXT = LITTEXT
DO X = 1 FOR WORDS(XTXT)
    IF X = 1 & LEFT(WORD(XTXT,1),2) = '/*' THEN LEAVE
    IF X = 1 & RIGHT(WORD(XTXT,1),2) = '*/' THEN LEAVE
    IF X = 1 & FIND(XCMD,WORD(XTXT,X)) ¬= Ø THEN ITERATE
    Y = FIND(ABBR,WORD(XTXT,X))   /* CHECK FOR ABBREVIATIONS */
    IF X = 1 & Y ¬= Ø
      THEN DO
       PARSE VALUE SUBWORD(ABBR,Y) WITH . '@' Y .
       XTXT = Y SUBWORD(XTXT,2) /* USE FULLY QUAL COMMAND NOT ABBR. */
       END
    IF X = 1 & XEQ = '=' THEN ITERATE
    IF FIND(LCMD,LEFT(WORD(XTXT,X),1)) ¬= Ø THEN ITERATE
    IF FIND('SAY PUSH QUEUE',WORD(XTXT,1)) = Ø
```

```
       THEN DO
        IF FIND(RCMD,RIGHT(WORD(XTXT,X),1)) ¬= Ø THEN ITERATE
        IF FIND(LSKP,WORD(XTXT,X)) ¬= Ø THEN ITERATE
        END
    XTXT = SUBWORD(XTXT,1,X-1) "\"WORD(XTXT,X)"\" SUBWORD(XTXT,X+1)
    Z = 1
    END
IF Z = 1 THEN LITNEW = XTXT
IF LITNEW = '' THEN SIGNAL LITNEXT
LITREP:
'COMMAND REPLACE' LITNEW
SIGNAL LITNEXT
LITEXIT:
'COMMAND TOP'
"COMMAND C/\ \/ / * * "
'COMMAND TOP'
"COMMAND C/\/'/ * * "
'COMMAND TOP'
"COMMAND C/~// * * "
'COMMAND TOP'
'COMMAND GET EXETOREX SKELETON'
'COMMAND I VINDEX = WORDS(ARGSTRING)'
'COMMAND I PARSE VAR ARGSTRING V1 V2 V3 V4 V5 V6 V7 V8 V9 V1Ø,'
'COMMAND I       V11 V12 V13 V14 V15 V16'
'COMMAND BOTTOM'
'COMMAND INPUT /* ORIGINAL EXEC SOURCE FOR' FN FT FM 'FOLLOWS...'
'COMMAND BOTTOM'
'COMMAND GET' FN FT FM
'COMMAND INPUT END OF ORIGINAL SOURCE FOR' FN FT FM '*/'
'COMMAND LOC :1'
'COMMAND C/$FN/'FN'/ '
'COMMAND LOC :1'
'COMMAND C/$FT/'FT'/ '
'COMMAND LOC :1'
'COMMAND C/$FM/'FM'/ '
'COMMAND LOC :1'
'COMMAND C/$UID/'USERID()'/ '
EXIT:
'COMMAND TOP'
'CMSQ STATE EXETOREX TEMPSIGS'
IF RC = Ø THEN 'CMSQ ERASE EXETOREX TEMPSIGS'
'CMSQ STATE EXETOREX TEMPTAGS'
IF RC = Ø THEN 'CMSQ ERASE EXETOREX TEMPTAGS'
'DESBUF'
IF FILESTCK THEN PUSH 'FFILE'
EXIT ØØØ
ERRØ1Ø:
SAY 'EXETOREX - UNABLE TO GET FN FT FM OF HOST PROGRAM. '
EXIT Ø1Ø
```

An approach to file transfer between Unix and VM

In our organization we transfer files between PCs and the mainframe using IBM Personal Communications/3270 (ie PC/3270) Version 4.00 and the corresponding module IND$FILE, which is executed in a CMS virtual machine.

Recently we received a RISC machine and several PCs were connected to the RISC server via a LAN. The PCs are still locally connected to the mainframe via DFT cards in each of them through an IBM 3274 local terminal controller attached to the mainframe. Each server's (network) disk appears as a disk to every PC node of the LAN. Naturally the problem arose of how to exchange files *directly* between the server's disks and the mainframe using the PC nodes.

ASCII files in PC DOS consist of lines each ending with <CR><LF> (carriage return, line feed). When transferring files between the PC and the mainframe the module IND$FILE adds or removes <CR><LF> depending on the direction of the file transfer. But lines of ASCII files in Unix format end with just <LF>. However the module IND$FILE is not designed for this format. Therefore IND$FILE has to be modified in such a way that the module produced will correctly process ASCII files in Unix format.

Because I haven't the source of IND$FILE, I had to make the necessary changes in the executable code. Before that, I had to stop the flow of control in the particular case of transferring ASCII files. I did it using the PER facility.

I used the ZAP utility. I didn't change the size of the control sections, I placed most of the updates over the unexecutable parts of them, for example over "5664-281 COPYRIGHT IBM CORP 1983, 1984;..." in the first control section INDFTCMP. However the text "PROPERTY OF IBM, REFER TO..." was not overlaid.

The following code is a CMS file, named UNIX ZAP, and consists of control statements for the ZAP utility.

```
NAME IND$FILE INDFTCMP
*
```

```
REP 0004 41A64000
REP 0008 950AA000
REP 000C 4780F014
REP 0010 18A6
REP 0012 06A0
REP 0014 47F0C18A
REP 0018 5960F07C
REP 001C 4740F024
REP 0020 920D5100
REP 0024 47F0F00C
REP 0028 41A0000D
REP 002C 42A64000
REP 0030 18A6
REP 0032 47F0C18A
REP 0036 00000000000000000000
*
REP 0D6A 0A0A         CHANGE CRLF TO LFLF
*
NAME IND$FILE INDFTAMC
*
REP 0086 920AA000     FORMERLY 921AA000
REP 03D8 00000000     FORMERLY 00000001
*
NAME IND$FILE INDFTAMG
*
REP 01A2 47F0CCB8     FORMERLY 18A606A0
REP 029C 0700         FORMERLY 0660
REP 02B0 0700         FORMERLY 0650
*
NAME IND$FILE INDFTAMP
*
REP 0004 58F0CCC0
REP 0008 47F0F000
REP 000C 00020004
*
REP 02BE D200A0003068 FORMERLY D201A0003068
REP 0494 00000001     FORMERLY 00000002
*
NAME IND$FILE INDFTUTL
*
*   ---------------
*   ASCII TO EBCDIC
*   ---------------
*
REP 0371 4F
REP 03AD 5A
*
*   ---------------
*   EBCDIC TO ASCII
*   ---------------
```

```
*
REP Ø29F 21
REP Ø2AA 5D
END
```

To make the updates in the module IND$FILE, the ZAP utility is invoked by the command:

```
ZAP MODULE (INPUT UNIX .
```

The file-id of the module produced is also IND$FILE MODULE. This module processes Unix-format ASCII files as needed, but it doesn't do it properly in the case of PC DOS ASCII files. Therefore it is reasonable to rename it to IND$UNIX MODULE.

When the 3270 session is started from the PC via PC/3270, the file transfer between the PC and the mainframe is performed by switching to a DOS session and, after that, by issuing one of the following two commands:

```
SEND [d:]name.ext fname ftype fmode (ASCII CRLF other opts if any
RECEIVE [d:]name.ext fname ftype fmode (ASCII CRLF ...
```

The command to use depends on the direction of the transfer. These commands send to the CMS machine one of the following two orders:

```
IND$FILE PUT fname ftype fmode (ASCII CRLF other opts if any
IND$FILE GET fname ftype fmode (ASCII CRLF other opts if any
```

It is clear that they call the module IND$FILE.

But now there are two modules – IND$FILE and IND$UNIX. Because these two modules behave differently in the case of ASCII files, it was necessary to create an EXEC, named IND$FILE, which, besides standard options, accepts an additional one – UNIX. If it is present, it calls IND$UNIX MODULE. If this option is not present, the EXEC calls the module IND$FILE, which performs the file transfer between the PC and the mainframe in the usual way.

The code of this EXEC is shown below.

```
/* IND$FILE EXEC */
 TRACE;
 PARSE UPPER ARG FUNC FNAME FTYPE ARGUMENTS;
"EXECIO * CP (STEM MSGWNG. STRING Q SET";
 MSG = 'MSG OFF';
 WNG = 'WNG OFF';
 IF POS('MSG OFF',MSGWNG.1) = Ø THEN MSG = 'MSG ON';
```

```
 IF POS('WNG OFF',MSGWNG.1) = Ø THEN WNG = 'WNG ON';
"EXECIO Ø CP (SKIP STRING SET MSG OFF"; /* PLEASE, DON'T DISTURB */
"EXECIO Ø CP (SKIP STRING SET WNG OFF"; /* SAME AS ABOVE */
 TYPE = 'FILE';
 N = WORDPOS('UNIX',ARGUMENTS);
 IF N ¬= Ø THEN DO;
                 TYPE = 'UNIX';
                 ARGUMENTS = DELWORD(ARGUMENTS,N,1);
                END;
"IND$"||TYPE||" "||FUNC||" "||FNAME||" "||FTYPE||" "||ARGUMENTS;
 RETCODE = RC;
"EXECIO Ø CP (SKIP STRING SET "MSG; /* RESTORE MSG SETTING */
"EXECIO Ø CP (SKIP STRING SET "WNG; /* RESTORE WNG SETTING */
 EXIT RETCODE;
```

With IND$UNIX MODULE and IND$FILE EXEC, we can exchange ASCII files between Unix and VM from the DOS session of PC/3270 using the commands:

```
SEND P:name.ext fname ftype fmode (ASCII CRLF UNIX ...
RECEIVE P:name.ext fname ftype fmode (ASCII CRLF UNIX ...  ,
```

where P is one of the Unix server's disks (actually the HOME directory of the user). If we omit the option UNIX and choose one of the PC's native disks, we will use the standard PC DOS-VM mode of the file transfer.

There is another method to exchange ASCII files between the mainframe and the server's disks using PC nodes on the LAN. It consists of two steps: standard PC/3270 file transfer between the PC and the mainframe and, for example, use of FTP to move files from/ to the PC's disk and the network disk. However, it needs intermediate disk space (the PC's native disk) and is more time consuming.

It is also possible to connect the Unix server to a channel of the mainframe. This approach is much more powerful, because it offers many other opportunities. However it needs additional software and hardware and, consequently, is much more expensive.

Note: the changes made in IND$FILE MODULE to produce IND$UNIX don't affect the other functions of IND$FILE that are not associated with ASCII files.

Vladimir Ivanov Valov
Chief Expert in System Programming
Information Centre of Ministry of Finance (Bulgaria)

VM is dead – long live VM!

"VM is dead"– how many times have I heard that one? The first time was in 1981 when I was working as a computer operator for IBM. I was told by my shift leader: "That is the VM system in the corner of the machine room, but only developers use it and we will be taking it out soon".

That viewpoint was wrong then, and with the 'Web enabling' of VM it is even more erroneous now. What no-one reckoned with was the fierce loyalty of the VM user, or VM bigots as other users call us. I should admit at this stage that I am a confirmed VM user and have been for 16 years – there, I feel better for having admitted it!

The architecture of VM – single user 'virtual machines' is the equivalent of personal computing for the mainframe or, as IBM would have us believe, 'super server'. Three or four years ago, the rise of the ubiquitous PC and the hardware improvements on the mainframe meant that multiple guest systems could be run on the physically partitioned processor without VM – which looked like spelling the end of VM.

HARDWARE

However, three things have contributed to the phoenix like rise of VM from the ashes of 'downsizing'.

Firstly, there is reliability. Independent studies show the availability of a mainframe system is typically 99.7%. This has been achieved using redundant processors, architecture changes to reduce maintenance requirements, and CMOS technology to produce a massive reduction in running costs. All these factors have conspired to give the mainframe renewed business appeal.

Against this, the reliability of a PC-based system is typically 87%. If you have business-critical applications, this availability simply is not good enough. In addition, it has been estimated in the last three years that 50% of attempted client/server implementations have failed for either technical or cost reasons.

Secondly, there is the cost. Studies, particularly in the USA, have shown that computing costs based on the mainframe are typically $2,282 annually per seat. This is in comparison with PC-based computing that typically costs $6,446 annually per seat.

Thirdly, there is scalability. VM, in conjunction with the CMOS and ESA architectures, and resource sharing between processors, can now support huge user populations. Installations do not have to worry about how many users they can support concurrently, and can instead concentrate on the implementation of new applications.

SOFTWARE

Enough about mainframes. What is it specifically about VM that has given it a new lease of life in the last four years? It is actually belated recognition that VM itself fits neatly together with the newer technology allowing the mainframe to be extended into the new networks and applications, but without losing its underlying strengths.

The virtual machine concept provides overall security for users with programs running in the virtual machine, secure from unauthorized access unless the administrator of that virtual machine grants explicit access to another user. An extension of this is that under VM you can set up multiple copies of systems. This provides a full range of services from one platform; ie you can offer a production TCP/IP system, and make a test version available for acceptance testing at the same time. Application testing under VM is made safer using these multiple copies. The work of one user can be protected from the errors or changes of other users. For example, changes to the application judged likely to cause an abend can be tested in a separate virtual machine.

VM's personal computing approach offers services similar to those provided on LAN servers. This includes file and database managers, security and accounting, communication servers, and print file transmission. So the fact that PCs and Unix workstations are promoted as the future in computing is a little surprising – VM has been doing this for a long time (*plus ça change*). VM also supports local and remote clients. This can be another VM system, a TCP/IP network, or

LU6.2. This means that VM can act as a 'super server' and support client/server applications using an extensive range of communication protocols.

The advent of VM Web servers means that VM delivers on the latest 'hot button' in the computing industry. VM Web servers can access VM/CMS files, applications in REXX and CMS Pipelines, and even DB2. Web browsing is also possible from VM, with EnterWEB from Macro 4 and Enterprise View (née Charlotte) from Beyond providing supported products that extend the reach of the estimated 30,000,000 3270 terminals(!) from the traditional mainframe applications out into the World Wide Web or intranet.

The issue of security is a vital one for the Web-enabled user. It cannot be disputed that Unix systems and PC-based servers have a perceived security problem. This is not meant to be derogatory, but initially security was never a priority for these types of system. As they have started running 'business-critical' applications, then the search is on to secure these systems from unauthorized external access. This is not a problem to VM. It is licensed to US Department of Defence C2 standard, which means it has 'positive' security, ie you have to allow access explicitly to a resource. Using VM, a business can offer intranet/Internet access and still retain control and protect its business data.

The biggest drawback to VM's future was overcoming the perception that VM was a closed architecture. Two features of VM demonstrate that VM can provide a secure base to many different types of application.

The first of these is the Open Systems Adapter. This allows data to be transferred between TCP/IP, VTAM, and LAN networks supported by the Adapter. The following types of LAN protocol are supported: Ethernet, Token Ring, Fibre Distributed Data Interface (FDDI), and Asynchronous Transfer Mode (ATM).

The second is compliance with POSIX. The POSIX file system is supported by VM, which means that VM can 'host' Unix applications. This is part of Open Distributed Client/Server and allows a business to develop and run applications on any DCE platform.

How much more ‘open’ do you need? My personal view is that people are coming to accept the notion, long recognized in developing economies, that there is an appropriate technology depending on what you are trying to achieve. The PC has changed computing for ever – the centralized computing model where absolutely everything is run on a central processor no longer exists, but that is not what today’s mainframe is all about and VM exploits this situation.

AND FINALLY

The PC has given the computing user a taste of DIY computing, but what businesses need is a mainframe to underpin this freedom.

The PC gives the user a modern look and feel, local flexibility, and GUI front ends. However, if you want to run a client/server or Web-enabled application, the strengths of VM mean that you get secure, available, reliable, and, most importantly, scalable applications.

This is where VM becomes the ‘appropriate technology’. People have recognized that with the Open System support in VM you get a cost-effective ‘super server’ that meets the demands of the end user, but also satisfies the economic requirements of a business. That is why VM has a ‘rock solid’ future when four years ago it looked like it was finished as an operating system.

Now, did you keep that map of where you buried your mainframe?

Tony Sudworth
Product Marketing Consultant
Macro 4 (UK)

If you are running a Web server under VM then let us know how it is working and how users are responding to it.

We are also looking for performance hints and tips for VM/ESA. Let us know what makes VM work better at your site.

You can contact us on any of the addresses on page 2.

Electronic bulletin board - part 2

This month we continue the code for an electronic bulletin board system.

```
  if dy = 'Sunday'    then do; a=1; b=2; c=3; d=4; e=5; f=6; g=7; end
  if dy = 'Monday'    then do; a=Ø; b=1; c=2; d=3; e=4; f=5; g=6; end
  if dy = 'Tuesday'   then do; a=Ø; b=Ø; c=1; d=2; e=3; f=4; g=5; end
  if dy = 'Wednesday' then do; a=Ø; b=Ø; c=Ø; d=1; e=2; f=3; g=4; end
  if dy = 'Thursday'  then do; a=Ø; b=Ø; c=Ø; d=Ø; e=1; f=2; g=3; end
  if dy = 'Friday'    then do; a=Ø; b=Ø; c=Ø; d=Ø; e=Ø; f=1; g=2; end
  if dy = 'Saturday'  then do; a=Ø; b=Ø; c=Ø; d=Ø; e=Ø; f=Ø; g=1; end
  limit = max.mm
  dline = 12
do 6
  if a =  Ø | a = 99 then a.fld = '    '
            else if a < 1Ø then a.fld = '   ' || a
                           else a.fld = '  '  || a
  if b =  Ø | b = 99 then b.fld = '    '
            else if b < 1Ø then b.fld = '   ' || b
                           else b.fld = '  '  || b
  if c =  Ø | c = 99 then c.fld = '    '
            else if c < 1Ø then c.fld = '   ' || c
                           else c.fld = '  '  || c
  if d =  Ø | d = 99 then d.fld = '    '
            else if d < 1Ø then d.fld = '   ' || d
                           else d.fld = '  '  || d
  if e =  Ø | e = 99 then e.fld = '    '
            else if e < 1Ø then e.fld = '   ' || e
                           else e.fld = '  '  || e
  if f =  Ø | f = 99 then f.fld = '    '
            else if f < 1Ø then f.fld = '   ' || f
                           else f.fld = '  '  || f
  if g =  Ø | g = 99 then g.fld = '    '
            else if g < 1Ø then g.fld = '   ' || g
                           else g.fld = '  '  || g
  a.att = lo
  b.att = lo
  c.att = lo
  d.att = lo
  e.att = lo
  f.att = lo
  g.att = lo
  dte = mm || '/' || dd || '/' || yy
  if dte = DATE(USA)
     then do
          if a = dd then a.att = hi
          if b = dd then b.att = hi
```

```
            if c = dd then c.att = hi
            if d = dd then d.att = hi
            if e = dd then e.att = hi
            if f = dd then f.att = hi
            if g = dd then g.att = hi
            end
  a.xxx = a.att || a.fld
  b.xxx = b.att || b.fld
  c.xxx = c.att || c.fld
  d.xxx = d.att || d.fld
  e.xxx = e.att || e.fld
  f.xxx = f.att || f.fld
  g.xxx = g.att || g.fld
  yyy  = lo || ' |'
  xxx  = '     ' ||,
  a.xxx || b.xxx || c.xxx || d.xxx || e.xxx || f.xxx || g.xxx || ' '
  zzz  = lo || '      |'
  xxx  = CENTER(xxx,46)
  xxx  = '       ' yyy || xxx || zzz
  xxx  = CENTER(xxx,8Ø)
  dline = dline + 1
  'COMMAND SET RESERVED' dline 'BLUE NONE N' xxx
  a = g + 1
  b = a + 1
  c = b + 1
  d = c + 1
  e = d + 1
  f = e + 1
  g = f + 1
  if a > limit then a = 99
  if b > limit then b = 99
  if c > limit then c = 99
  if d > limit then d = 99
  if e > limit then e = 99
  if f > limit then f = 99
  if g > limit then g = 99
end
  dte = mm || '/' || dd || '/' || yy
  if dte = DATE(USA)
     then do
          jul  = DATE(J)
          jda  = SUBSTR(jul,3,3)
          xxx  = '               Day of Year : ' || jda
          end
     else xxx  = '                                    '
  xxx  = CENTER(xxx,46)
  xxx  = '|' || xxx || '|'
  xxx  = CENTER(xxx,8Ø)
  'COMMAND SET RESERVED +19 BLUE NONE N' xxx
  xxx  = '|                                              |'
  xxx  = CENTER(xxx,8Ø)
```

```
  'COMMAND SET RESERVED +2Ø BLUE NONE N' xxx
  xxx  = '+-------------------------------------------+'
  xxx  = CENTER(xxx,8Ø)
  'COMMAND SET RESERVED +21 BLUE NONE N' xxx
  'COMMAND SET RESERVED +22 BLUE NONE N '
return
CALCJUL:
  call SETMO
  jda = SUBSTR(pcal,3,2)
  pmm = SUBSTR(pcal,1,2)
  idx = Ø1
  if pmm > Ø1 then
  do until pmm = idx
     jda = jda + max.idx
     idx = idx + 1
     if LENGTH(idx) = 1 then idx = Ø || idx
  end
CHKJUL:
  if LENGTH(jda) = 1 then jda = ØØ || jda
  if LENGTH(jda) = 2 then jda = Ø || jda
  jul = SUBSTR(pcal,5,2) || jda
RETURN
SETMO:
  max. = Ø
  max.Ø1 = 31
  max.Ø2 = 28
  max.Ø3 = 31
  max.Ø4 = 3Ø
  max.Ø5 = 31
  max.Ø6 = 3Ø
  max.Ø7 = 31
  max.Ø8 = 31
  max.Ø9 = 3Ø
  max.1Ø = 31
  max.11 = 3Ø
  max.12 = 31
  idx = ØØ
  do until yy < idx
     if yy = idx then max.Ø2 = 29
     idx = idx + Ø4
     if LENGTH(idx) = 1 then idx = Ø || idx
  end
return
CALCDAY:
  days = yy * 365
  idx = ØØ
  do until yy < idx
     if yy > idx then days = days + 1
     idx = idx + Ø4
     if LENGTH(idx) = 1 then idx = Ø || idx
```

```
  end
  days = days + jda
  dayn = days / 7
  dayn = TRUNC(dayn)
  dayn = dayn * 7
  dayn = ( days - dayn ) - 1
  day  = '????????'
  if dayn = -1 then dy = 'Saturday'
  if dayn =  Ø then dy = 'Sunday'
  if dayn =  1 then dy = 'Monday'
  if dayn =  2 then dy = 'Tuesday'
  if dayn =  3 then dy = 'Wednesday'
  if dayn =  4 then dy = 'Thursday'
  if dayn =  5 then dy = 'Friday'
  if dayn =  6 then dy = 'Saturday'
return
CALCMTH:
  if mm = Ø1 then mth = 'January'
  if mm = Ø2 then mth = 'February'
  if mm = Ø3 then mth = 'March'
  if mm = Ø4 then mth = 'April'
  if mm = Ø5 then mth = 'May'
  if mm = Ø6 then mth = 'June'
  if mm = Ø7 then mth = 'July'
  if mm = Ø8 then mth = 'August'
  if mm = Ø9 then mth = 'September'
  if mm = 1Ø then mth = 'October'
  if mm = 11 then mth = 'November'
  if mm = 12 then mth = 'December'
return
```

EBBCHECK EXEC

```
/*  System     : EBBNEWS                                                 */
/*  EXEC name  : EBBCHECK                                                */
/*  Invoked by : EBBNEWS EXEC                                            */
/*  Function   : This EXEC displays a message on the menu if there       */
/*               are no new items.                                       */
  parse arg okay ctl .
  if okay <> '$OKAY$'
     then do
          say 'This EXEC can only be invoked through the EBBNEWS EXEC'
          exit 99
          end
  'GLOBALV SELECT EBBNEWS GET naddr nmode'
  'SET CMSTYPE HT'
  'ACC' naddr nmode
  saverc = rc
  'SET CMSTYPE RT'
```

```
if saverc then exit rc
'EXECIO * DISKR $EBBNEWS $CONTROL A (STEM' ctlrec.
'FINIS $EBBNEWS $CONTROL A'
if ctlrec.Ø = Ø then exit
count = Ø
i = Ø
do ctlrec.Ø
   i = i + 1
   board = SUBSTR(ctlrec.i,1,8)
   bdate = SUBSTR(ctlrec.i,1Ø,8)
   btime = SUBSTR(ctlrec.i,19,5)
   brest = SUBSTR(ctlrec.i,25,55)         /* ' ' or 'Screened'      */
   parse upper var brest brest
   if WORD(brest,1) = 'SCREENED'
      then iterate
   bcomp = bdate || btime
   'EXECIO * DISKR' board 'EBBNEWS' nmode '(STEM' item.
   j = 1
   do until j > item.Ø
      if j > 2
         then do
              idate = SUBSTR(item.j,1Ø1,8)
              itime = SUBSTR(item.j,75,5)
              icomp = idate || itime
              if icomp > bcomp
                 then do
                      count = count + 1
                      end
              end
      j = j + 1
   end
end
if count = Ø
   then do
        queue 'MSG There are no new items',
              'on any of the "unscreened" bulletin boards'
        exit
        end
if count = 1
   then do
   queue 'MSG There is 1 new bulletin item.  Press PF8 to view.'
        exit
        end
if count > 1
   then do
queue 'MSG There are' count 'new bulletin items.  Press PF8 to view.'
        exit
        end

'VMFCLEAR'
exit
```

EBBCREEN XEDIT

```
/*  System     : EBBNEWS                                                 */
/*  Macro name : EBBCREEN                                                */
/*  Invoked by : EBBMENU XEDIT macro                                     */
/*  Function   : This macro is invoked from the main menu of the         */
/*               PROFS Bulletin Board.  If the cursor is placed          */
/*               on a line containing a board, then the "SCREEN"         */
/*               funtion is toggled when PF11 is pressed.                */
  if SUBSTR(USERID(),1,7) = 'EBBNEWS'
     then do
          'COMMAND MSG The "Screen" facility is not available.'
          exit
          end
  'COMMAND SET CTLCHAR ! ESCAPE'
  'COMMAND SET CTLCHAR % PROTECT HIGH'
  'COMMAND SET CTLCHAR @ PROTECT NOHIGH'
  'COMMAND EXTRACT /CURSOR'
  line = CURSOR.1
  address CMS 'EXECIO * DISKR $EBBNEWS $CONTROL A (STEM' ctlrec.
  address CMS 'FINIS $EBBNEWS $CONTROL A'
  if line < 5
     then do
          signal HELP
          exit
          end
  if line > 1Ø
     then do
          signal HELP
          exit
          end
  if line = 5
     then do
          stat = WORD(ctlrec.1,4)
          if stat = ''
             then do
                  zzz = '...................Screened'
                  address CMS 'MAKEBUF'
                  queue 'LOCATE/STEVE'
                  queue 'SET ZONE 3Ø 8Ø'
                  QUEUE 'CHANGE/        /SCREENED/1 1'
                  queue 'FILE'
                  address CMS 'XEDIT $EBBNEWS $CONTROL A'
                  address CMS 'DROPBUF'
                  end
             else do
                  zzz = '   '
                  address CMS 'MAKEBUF'
                  queue 'LOCATE/STEVE'
                  queue 'SET ZONE 3Ø 8Ø'
```

```
                QUEUE 'CHANGE/SCREENED/        /1 1'
                queue 'FILE'
                address CMS 'XEDIT $EBBNEWS $CONTROL A'
                address CMS 'DROPBUF'
                address CMS 'EXEC EBBCHECK $OKAY$'
                end
          xxx = '  PF1'
          yyy = 'From The Desk of... the Company President'
          'COMMAND SET RESERVED  5 WHITE NONE HIGH' xxx '!@'yyy zzz
          end
   if line = 6
      then do
          stat = WORD(ctlrec.2,4)
          if stat = ''
             then do
                zzz = '.........................Screened'
                address CMS 'MAKEBUF'
                queue 'LOCATE/SAFETY'
                queue 'SET ZONE 3Ø 8Ø'
                queue 'CHANGE/        /SCREENED/1 1'
                queue 'FILE'
                address CMS 'XEDIT $EBBNEWS $CONTROL A'
                address CMS 'DROPBUF'
                end
             else do
                zzz = '   '
                address CMS 'MAKEBUF'
                queue 'LOCATE/SAFETY'
                queue 'SET ZONE 3Ø 8Ø'
                queue 'CHANGE/SCREENED/        /1 1'
                queue 'FILE'
                address CMS 'XEDIT $EBBNEWS $CONTROL A'
                address CMS 'DROPBUF'
                address CMS 'EXEC EBBCHECK $OKAY$'
                end
          xxx = '  PF2'
          yyy = 'REPORTS and Safety News'
          'COMMAND SET RESERVED  6 WHITE NONE HIGH' xxx '!@'yyy zzz
          end
   if line = 7
      then do
          stat = WORD(ctlrec.3,4)
          if stat = ''
             then do
                zzz = '.........................Screened'
                address CMS 'MAKEBUF'
                queue 'LOCATE/VISITOR'
                queue 'SET ZONE 3Ø 8Ø'
                queue 'CHANGE/        /SCREENED/1 1'
                queue 'FILE'
```

```
                address CMS 'XEDIT $EBBNEWS $CONTROL A'
                address CMS 'DROPBUF'
                end
           else do
                zzz = '   '
                address CMS 'MAKEBUF'
                queue 'LOCATE/VISITOR'
                queue 'SET ZONE 3Ø 8Ø'
                queue 'CHANGE/SCREENED/        /1 1'
                queue 'FILE'
                address CMS 'XEDIT $EBBNEWS $CONTROL A'
                address CMS 'DROPBUF'
                address CMS 'EXEC EBBCHECK $OKAY$'
                end
        xxx = '  PF3'
        yyy = 'Vistors and Plant Tours'
        'COMMAND SET RESERVED  7 WHITE NONE HIGH' xxx '!@'yyy zzz
        end
if line = 8
   then do
        stat = WORD(ctlrec.4,4)
        if stat = ''
           then do
                zzz = '.................Screened'
                address CMS 'MAKEBUF'
                queue 'LOCATE/VACATION'
                queue 'SET ZONE 3Ø 8Ø'
                queue 'CHANGE/        /SCREENED/1 1'
                queue 'FILE'
                address CMS 'XEDIT $EBBNEWS $CONTROL A'
                address CMS 'DROPBUF'
                end
           else do
                zzz = '   '
                address CMS 'MAKEBUF'
                queue 'LOCATE/VACATION'
                queue 'SET ZONE 3Ø 8Ø'
                queue 'CHANGE/SCREENED/        /1 1'
                queue 'FILE'
                address CMS 'XEDIT $EBBNEWS $CONTROL A'
                address CMS 'DROPBUF'
                address CMS 'EXEC EBBCHECK $OKAY$'
                end
        xxx = '  PF4'
        yyy = 'Vacations and OUT-OF-TOWN Notices'
        'COMMAND SET RESERVED  8 WHITE NONE HIGH' xxx '!@'yyy zzz
        end
if line = 9
   then do
        stat = WORD(ctlrec.5,4)
```

```
         if stat = ''
            then do
                 zzz = '.........................Screened'
                 address CMS 'MAKEBUF'
                 queue 'LOCATE/FYI'
                 queue 'SET ZONE 3Ø 8Ø'
                 queue 'CHANGE/        /SCREENED/1 1'
                 queue 'FILE'
                 address CMS 'XEDIT $EBBNEWS $CONTROL A'
                 address CMS 'DROPBUF'
                 end
            else do
                 zzz = '   '
                 address CMS 'MAKEBUF'
                 queue 'LOCATE/FYI'
                 queue 'SET ZONE 3Ø 8Ø'
                 queue 'CHANGE/SCREENED/        /1 1'
                 queue 'FILE'
                 address CMS 'XEDIT $EBBNEWS $CONTROL A'
                 address CMS 'DROPBUF'
                 address CMS 'EXEC EBBCHECK $OKAY$'
                 end
         xxx = '  PF5'
         yyy = 'FYI - General Information'
         'COMMAND SET RESERVED  9 WHITE NONE HIGH' xxx '!@'yyy zzz
         end
   if line = 1Ø
      then do
         stat = WORD(ctlrec.6,4)
         if stat = ''
            then do
                 zzz = '................................Screened'
                 address CMS 'MAKEBUF'
                 queue 'LOCATE/IDEAS'
                 queue 'SET ZONE 3Ø 8Ø'
                 queue 'CHANGE/        /SCREENED/1 1'
                 queue 'FILE'
                 address CMS 'XEDIT $EBBNEWS $CONTROL A'
                 address CMS 'DROPBUF'
                 end
            else do
                 zzz = '   '
                 address CMS 'MAKEBUF'
                 queue 'LOCATE/IDEAS'
                 queue 'SET ZONE 3Ø 8Ø'
                 queue 'CHANGE/SCREENED/        /1 1'
                 queue 'FILE'
                 address CMS 'XEDIT $EBBNEWS $CONTROL A'
                 address CMS 'DROPBUF'
                 address CMS 'EXEC EBBCHECK $OKAY$'
```

```
                end
          xxx = '  PF6'
          yyy = 'Task Force Groups'
          'COMMAND SET RESERVED 1Ø WHITE NONE HIGH' xxx '!@'yyy zzz
          end
  exit
HELP:
  'EXEC EBBHELP EBBCREEN'
  exit
```

EBBCURR EXEC

```
/*  System     : EBBNEWS                                              */
/*  EXEC name  : EBBCURR                                              */
/*  Invoked by : EBBNEWS XEDIT macro                                  */
/*  Function   : This EXEC displays all current items from all the    */
/*               bulletin boards that have not previously been seen   */
/*               by the terminal user.                                */
  parse arg okay ctl .
  if okay <> '$OKAY$'
     then do
          say 'This EXEC can only be invoked through the EBBNEWS EXEC'
          exit 99
          end
  'GLOBALV SELECT EBBNEWS GET naddr nmode'
  'SET CMSTYPE HT'
  'ACC' naddr nmode
  saverc = rc
  'SET CMSTYPE RT'
  if saverc then exit rc
  'SET CMSTYPE HT'
  'STATE $EBBNEWS $CONTROL A'
  state_rc = rc
  'SET CMSTYPE RT'
  if state_rc ¬= Ø
     then do
          record = 'PRES     ØØØØØØØØ ØØ:ØØ'
          'EXECIO 1 DISKW $EBBNEWS $CONTROL A Ø F 8Ø (STRING' record
          record = 'SAFETY   ØØØØØØØØ ØØ:ØØ'
          'EXECIO 1 DISKW $EBBNEWS $CONTROL A (STRING' record
          record = 'VISITOR  ØØØØØØØØ ØØ:ØØ'
          'EXECIO 1 DISKW $EBBNEWS $CONTROL A (STRING' record
          record = 'VACATION ØØØØØØØØ ØØ:ØØ'
          'EXECIO 1 DISKW $EBBNEWS $CONTROL A (STRING' record
          record = 'FYI      ØØØØØØØØ ØØ:ØØ'
          'EXECIO 1 DISKW $EBBNEWS $CONTROL A (STRING' record
          record = 'IDEAS    ØØØØØØØØ ØØ:ØØ'
          'EXECIO 1 DISKW $EBBNEWS $CONTROL A (STRING' record
          'FINIS $EBBNEWS $CONTROL A'
          end
```

```
'SET CMSTYPE HT'
'ERASE $EBBNEWS $CURRENT A'
'SET CMSTYPE RT'
if ctl = 'ALL'
   then title = 'ALL PROFS Bulletins... '
   else title = 'New PROFS Bulletins... '
'EXECIO 1 DISKW $EBBNEWS $CURRENT A Ø F 12Ø (STRING' title
blanks = '                          '
'EXECIO 1 DISKW $EBBNEWS $CURRENT A (STRING' blanks
'EXECIO * DISKR $EBBNEWS $CONTROL A (STEM' ctlrec.
'FINIS $EBBNEWS $CONTROL A'
count = Ø
i = Ø
do ctlrec.Ø
   i = i + 1
   board = SUBSTR(ctlrec.i,1,8)
   bdate = SUBSTR(ctlrec.i,1Ø,8)
   btime = SUBSTR(ctlrec.i,19,5)
   brest = SUBSTR(ctlrec.i,25,55)         /* ' ' or 'Screened'      */
   parse upper var brest brest
   if WORD(brest,1) = 'SCREENED'
      then iterate
   bcomp = bdate || btime
   if ctl = 'ALL'
      then bcomp = 'ØØØØØØØØØØ:ØØ'
      else bcomp = bdate || btime
   gotsome = y
   'EXECIO * DISKR' board 'EBBNEWS' nmode '(stem' item.
   j = 1
   do until j > item.Ø
      if j = 1
         then do
              heading = blanks SUBSTR(item.j,1,5Ø)
              heading = SUBSTR(heading,1,111) || board
              end
      if j > 2
         then do
              idate = SUBSTR(item.j,1Ø1,8)
              itime = SUBSTR(item.j,75,5)
              if j = item.Ø
                 then do
                      newrec = board idate itime brest
         'EXECIO 1 DISKW $EBBNEWS $CONTROL A' i 'F 8Ø (STRING' newrec
                      end
              icomp = idate || itime
              if icomp > bcomp
                 then do
                      currec = item.j board
                      if gotsome = y
                         then do
                 'EXECIO 1 DISKW $EBBNEWS $CURRENT A (STRING' heading
```

```
                  'EXECIO 1 DISKW $EBBNEWS $CURRENT A (STRING' blanks
                              gotsome = n
                              end
                  'EXECIO 1 DISKW $EBBNEWS $CURRENT A (STRING' currec
                       count = count + 1
                       end
               end
         j = j + 1
      end
      'EXECIO 1 DISKW $EBBNEWS $CURRENT A (STRING' blanks
   end
   'FINIS $EBBNEWS $CURRENT A'
   if count = Ø
      then do
           'ERASE $EBBNEWS $CURRENT A'
           queue 'MSG There are no new items',
                 'on any of the "unscreened" bulletin boards'
           exit
           end
   'VMFCLEAR'
   'XEDIT $EBBNEWS $CURRENT A (PROFILE EBBCURR'
   'ERASE $EBBNEWS $CURRENT A'
   'VMFCLEAR'
   exit
```

EBBCURR XEDIT

```
/*  SYSTEM     : EBBNEWS                                                 */
/*  MACRO NAME : EBBCURR                                                 */
/*  INVOKED BY : EBBCURR EXEC                                            */
/*  FUNCTION   : This macro formats a PROFS bulletin board for           */
/*               all current bulletins not yet seen by the user.         */
'COMMAND SET AUTOSAVE OFF'
'COMMAND SET MSGMODE OFF'
'COMMAND SET SCOPE ALL'
'COMMAND SET CASE M I'
'COMMAND SET CMDLINE OFF'
'COMMAND SET CURLINE ON 5'
'COMMAND SET MSGLINE ON 2'
'COMMAND SET PREFIX OFF'
'COMMAND SET SCALE OFF'
'COMMAND SET WRAP ON'
'COMMAND SET STAY ON'
'COMMAND SET SHADOW OFF'
'COMMAND SET VERIFY 1 8Ø'
'COMMAND SET COLOR  *        BLUE   NONE    HIGH'
'COMMAND SET COLOR  CURLINE  GREEN  NONE  NOHIGH'
'COMMAND SET COLOR  FILEAREA GREEN  NONE  NOHIGH'
'COMMAND SET COLOR  MSGLINE  RED    NONE    HIGH'
'COMMAND EXTRACT /FNAME'
```

```
'COMMAND SET LINEND OFF'
'COMMAND SET ENTER EBBVIEW'
'COMMAND SET PFØ1 EBBGOTO'
'COMMAND SET PFØ2 EMSG Invalid PF key pressed.  Try again.'
'COMMAND SET PFØ3 EMSG Invalid PF key pressed.  Try again.'
'COMMAND SET PFØ4 EBBVIEW'
'COMMAND SET PFØ5 EBBCAL'
'COMMAND SET PFØ6 EBBPRINT'
'COMMAND SET PFØ7 UP 1Ø'
'COMMAND SET PFØ8 DOWN 1Ø'
'COMMAND SET PFØ9 EXEC EBBHELP EBBCURR'
'COMMAND SET PF1Ø FORWARD'
'COMMAND SET PF11 BACKWARD'
'COMMAND SET PF12 QQUIT'
'COMMAND SET LINEND ON #'
'COMMAND SET CTLCHAR ! ESCAPE'
'COMMAND SET CTLCHAR % PROTECT HIGH'
'COMMAND SET CTLCHAR @ PROTECT NOHIGH'
xxx   = DATE(W) '      '
yyy   = 'Electronic News - PROFS Bulletin Board'
zzz   = '     ' DATE(USA)
'COMMAND SET RESERVED  1 N  !@' xxx ' !%' yyy ' !@' zzz
'COMMAND SET RESERVED  2 BLUE NONE N  '
xxx  = '  Move the cursor next to an item and press a PF key',
       '(or ENTER to view) '
'COMMAND SET RESERVED  3 BLUE NONE N' xxx
'COMMAND SET RESERVED  4 BLUE NONE N  '
'COMMAND :1'
'COMMAND EXTRACT /CURLINE'
xxx = CURLINE.3
xxx = SUBSTR(xxx,1,5Ø)
'COMMAND :1'
'COMMAND DELETE 2'
'COMMAND SET RESERVED  5 WHITE NONE HIGH' xxx
xxx = COPIES('-',79)
'COMMAND SET RESERVED  6 BLUE NONE N' xxx
xxx = COPIES('-',79)
'COMMAND SET RESERVED -3 BLUE NONE N' xxx
xxx  = ' PF1= Board     2=           3=      ',
       '   4= View        5= Calendar   6= Print '
'COMMAND SET RESERVED -2 BLUE NONE HIGH' xxx
xxx  = ' PF7= Up 1/2    8= Down 1/2  9= Help ',
       '  1Ø= Next Page  11= Previous  12= Return'
'COMMAND SET RESERVED -1 BLUE NONE HIGH' xxx
'COMMAND SET MSGMODE ON'
'COMMAND CURSOR SCREEN 1 1'
'COMMAND TOP'
exit
```

EBBDEL XEDIT

```
/*  System     : EBBNEWS                                                 */
/*  Macro name : EBBDEL                                                  */
/*  Invoked by : EBBBULL XEDIT macro                                     */
/*  Function   : This macro deletes detail information on an item        */
/*               listed in the PROFS Bulletin Board.                     */
  parse arg board pass curs1 curs2 desc
  'COMMAND EXTRACT /CURSOR/LSCREEN/SIZE/LINE'
  cursscrn = CURSOR.1
  cursfile = CURSOR.3
  screenl  = LSCREEN.1
  filel    = SIZE.1
  saveline = LINE.1
  if pass = 'PASS2'
     then signal PASS2
  if cursscrn > screenl  then signal MSG
  if cursscrn < 2        then signal MSG
  if cursfile > filel    then signal MSG
  if cursfile < 1        then signal MSG
  'CP SET SMSG ON'
  'SMSG EBBNEWS ::: '
  if rc = 45
     then do
     'EMSG EBBNEWS is not logged on... check with computer operations'
          'CP SET SMSG OFF'
          exit
          end
     else
     if rc = 57
        then do
     'EMSG EBBNEWS is not receiving... check with computer operations'
             'CP SET SMSG OFF'
             exit
             end
     else
     if rc > Ø
        then do
     'EMSG EBBNEWS is not available... NOTIFY OPERATIONS IMMEDIATELY'
             'CP SET SMSG OFF'
             exit
             end
  'CP SET SMSG OFF'
  'COMMAND EMSG Press PF3 again to delete the item, else press PF12'
  msgtext = 'Invalid key pressed.  Press PF3 again to delete, else PF12'
  'COMMAND SET LINEND OFF'
  'COMMAND SET ENTER EMSG' msgtext
  'COMMAND SET PFØ1 EMSG' msgtext
  'COMMAND SET PFØ2 EMSG' msgtext
  'COMMAND SET PFØ3 EBBDEL' board 'PASS2' cursscrn cursfile desc
  'COMMAND SET PFØ4 EMSG' msgtext
  'COMMAND SET PFØ5 EMSG' msgtext
```

```
  'COMMAND SET PFØ6 EMSG' msgtext
  'COMMAND SET PFØ7 EMSG' msgtext
  'COMMAND SET PFØ8 EMSG' msgtext
  'COMMAND SET PFØ9 EXEC EBBHELP#SET PF12 QQUIT'
  'COMMAND SET PF1Ø EMSG' msgtext
  'COMMAND SET PF11 EMSG' msgtext
  'COMMAND SET PF12 QQUIT'
  'COMMAND SET LINEND ON #'
  exit
PASS2:
  cursscrn = curs1
  cursfile = curs2
  'COMMAND SET LINEND OFF'
  'COMMAND SET ENTER EBBVIEW'
  'COMMAND SET PFØ1 EXEC EBBADD $OKAY$' board desc
  'COMMAND SET PFØ2 EBBREPLX' board desc
  'COMMAND SET PFØ3 EBBDEL' board 'PASS1'
  'COMMAND SET PFØ4 EBBVIEW'
  'COMMAND SET PFØ5 EMSG Invalid PF key pressed.  Try again'
  'COMMAND SET PFØ6 EBBPRINT'
  'COMMAND SET PFØ7 TOP#SORT * D 66 67 69 7Ø 72 73 75 79#CUR S 1 1'
  'COMMAND SET PFØ8 TOP#SORT * A 61 63#CUR S 1 1'
  'COMMAND SET PFØ9 EXEC EBBHELP#SET PF12 QQUIT'
  'COMMAND SET PF1Ø FORWARD'
  'COMMAND SET PF11 BACKWARD'
  'COMMAND SET PF12 QQUIT'
  'COMMAND SET LINEND ON #'
  'GLOBALV SELECT EBBNEWS GET naddr nmode'
CHECK:
  if cursscrn > screenl  then signal MSG
  if cursscrn < 2        then signal MSG
  if cursfile > filel    then signal MSG
  if cursfile < 1        then signal MSG
GETFN:
  'COMMAND :' cursfile
  'COMMAND EXTRACT /CURLINE'
  'COMMAND :' saveline
  strg = CURLINE.3
  if strg = ' ' then exit
  if SUBSTR(strg,1,5) = '-----' then exit
  init = SUBSTR(strg,61,3)
  'DESBUF'
  'EXECIO * DISKR EBBUSER LIST' nmode,
                        '(FINIS ZONE 1Ø 12 LOCATE /'init'/'
  if rc > Ø
     then do
     'COMMAND EMSG You are not authorized to delete Bulletin items'
          exit
          end
  if QUEUED() <> 2
     then do
```

```
          'COMMAND EMSG Error reading user list.  Call systems'
          exit
          end
  parse pull recnum; parse pull record
  parse var record ruser rinit rusr1 rusr2 rusr3 .
  usr = USERID()
  if usr = ruser | usr = rusr1 | usr = rusr2 | usr = rusr3
     then nop
     else do
          "COMMAND EMSG You are not authorized to delete",
                   ruser || "'s bulletins"
          exit
          end
  fn   = SUBSTR(strg,81,8)
  ft   = SUBSTR(strg,90,8)
  type = SUBSTR(strg,99,1)
EXECUTE:
  'EXEC EBBSEND $OKAY$ DELETE' board '...' fn ft
  if rc = 0
     then do
          'COMMAND :' cursfile
          'COMMAND CLOCATE :1'
          msg = LEFT('------- deleted ------',80,' ')
          'COMMAND CREPLACE' msg
          'COMMAND :' saveline
          end
  'COMMAND CURSOR SCREEN' cursscrn '1'
  exit
MSG:
  'COMMAND EMSG Place the cursor next to the item to delete',
              'and press PF3'
  exit
```

EBBDELX.XEDIT

```
/*  System     : EBBNEWS                                              */
/*  Macro name : EBBDELX                                              */
/*  Invoked by : EBBRECV EXEC                                         */
/*  Function   : This XEDIT macro validates the delete (compares the  */
/*               requestor initials to the file) and erases the line  */
/*               and the file if okay.                                */
  parse upper arg uinit fn ft
  fn  = SUBSTR(fn,1,8)
  ft  = SUBSTR(ft,1,8)
  srch = '/' || fn || ' ' || ft || '/'
  'COMMAND LOCATE' srch
  if rc > 1 then signal DONE
/*  'COMMAND EXTRACT /CURLINE'      */
/*  record = CURLINE.3              */
/*  rinit = SUBSTR(record,61,3)     */
```

```
/*  if rinit <> uinit then exit 88  */
  'COMMAND DELETE 1'
  if ft <> '........'
     then do
          'STATE' fn ft 'A'
          if rc = 0
             then 'ERASE' fn ft 'A'
          end
DONE:
  'COMMAND FILE'
  exit
```

EBBEDIT XEDIT

```
/*  System     : EBBNEWS                                              */
/*  Macro name : EBBEDIT                                              */
/*  Invoked by : EBBADD EXEC or EBBREPL EXEC                          */
/*  Function   : This macro displays the screen for replacing items   */
/*               on the PROFS Bulletin Board.                         */
  parse pull func board old_fn old_ft rest
  rest = STRIP(rest,'L')
  len = LENGTH(rest)
  dlm = POS(':::',rest)
  dlm = dlm - 1
  old_bull = SUBSTR(rest,1,dlm)
  old_bull = SUBSTR(old_bull,1,50)
  dlm = dlm + 4
  len = len - dlm
  len = len + 1
  desc = SUBSTR(rest,dlm,len)
  desc = STRIP(desc)
  call INIT
  call SETUP
  call DISPLAY
READ_TAGS:
  'DESBUF'
  'COMMAND READ ALL TAG'
  no_lines = QUEUED()
LOOP:
  pfkey = 0
  do no_lines
     parse pull strg
     parse var strg action .
     if action = 'ETK' then pfkey = 0
     if action = 'PFK' then pfkey = WORD(strg,2)
     if action = 'RES'
        then do
             row = WORD(strg,2)
             col = WORD(strg,3)
             cnt = WORDS(strg)
```

```
            if cnt > 3
              then loc = WORDINDEX(strg,4)
              else loc = Ø
            if row = '5'
               then do
                    if col = '1Ø'
               then if loc = Ø then,
            subj = '_____________________________________________________'
                                 else subj = SUBSTR(strg,loc,5Ø)
               else if loc = Ø then flash = 'N'
                                 else flash = SUBSTR(strg,loc,1)
                    end
            if row = '7'
               then if loc = Ø then cron = '____________'
                                 else cron = SUBSTR(strg,loc,12)
            if row = '9'
               then do
                    if loc = Ø then prof = '________'
                                 else prof = SUBSTR(strg,loc,8)
                    end
            if row = '11'
               then do
                    if col = '6Ø'
                       then if loc = Ø then fn = '________'
                                       else fn = SUBSTR(strg,loc,8)
                    if col = '71'
                       then if loc = Ø then ft = '________'
                                       else ft = SUBSTR(strg,loc,8)
                    end
            end
  end
  if pfkey =  Ø then signal ENTER_KEY
  if pfkey =  1 then signal EXECUTE
  if pfkey =  2 then signal ADD_LINE
  if pfkey =  3 then signal DEL_LINE
  if pfkey =  4 then signal VIEW
  if pfkey =  5 then signal CALENDAR
  if pfkey =  6 then signal FULLSCREEN
  if pfkey =  7 then signal REFRESH
  if pfkey =  8 then signal BADPF
  if pfkey =  9 then signal HELP
  if pfkey = 1Ø then signal FORWARD
  if pfkey = 11 then signal BACKWARD
  if pfkey = 12 then signal EXIT
  signal BADPF
ENTER_KEY:
  call DISPLAY
  signal READ_TAGS
EXECUTE:
  if SUBSTR(subj,1,8) = '________' | SUBSTR(subj,1,8) = '        '
     then do
```

```
         call DISPLAY
      'COMMAND EMSG You must supply a subject for the Bulletin Board'
         signal READ_TAGS
         end
      else do
   'COMMAND execio 1 DISKW $EBBNEWS $JUNK$ A 1 F 8Ø (FINIS STRING' subj
         if rc <> Ø
            then do
                 call DISPLAY
            'COMMAND EMSG ERROR writing the subject to your A disk'
                 signal READ_TAGS
                 end
         end
   if cron <> '____________' & cron <> '            '
      then do
         cron8 = SUBSTR(cron,1,8)
         cron4 = SUBSTR(cron,9,4)
         'PROFS RETRIEVE' cron8 cron4 'DISK'
         cronfn = 'D' || SUBSTR(cron,3,3) || SUBSTR(cron,9,4)
         'SET CMSTYPE HT'
         'STATE' cronfn 'SCRIPT A'
         saverc = rc
         'SET CMSTYPE RT'
         if saverc <> Ø
            then do
                 call DISPLAY
      'COMMAND EMSG' cron 'cannot be retrieved from the PROFS database.'
                 signal READ_TAGS
                 end
        'COPYFILE' cronfn 'SCRIPT A $EBBNEWS SCRIPT A (REPLACE'
        'ERASE' cronfn 'SCRIPT A'
        'COPYFILE $EBBNEWS $JUNK$ A $EBBNEWS $SUBJ$ A (REPLACE'
        'EXEC EBBSEND $OKAY$' func board 'PROFS' old_fn old_ft flash
        if rc = Ø
            then do
                 if func = 'REPLACE' then signal exit
                 call INIT
                 call DISPLAY
                 'COMMAND EMSG The Bulletin Board has been updated'
                 end
         signal READ_TAGS
         end
   if prof <> '________' & prof <> '        '
      then do
         xfn = STRIP(prof,,'_')
         'SET CMSTYPE HT'
         'STATE' xfn 'SCRIPT A'
         saverc = rc
         'SET CMSTYPE RT'
         if saverc <> Ø
            then do
```

```
                call DISPLAY
'COMMAND EMSG' xfn 'SCRIPT is not on your A disk.  No action taken.'
                signal READ_TAGS
                end
        'COPYFILE' xfn 'SCRIPT A $EBBNEWS SCRIPT A (REPLACE'
        'COPYFILE $EBBNEWS $JUNK$ A $EBBNEWS $SUBJ$ A (REPLACE'
        'EXEC EBBSEND $OKAY$' func board 'PROFS' old_fn old_ft flash
        if rc = 0
            then do
                 if func = 'REPLACE' then signal exit
                 call INIT
                 call DISPLAY
                 'COMMAND EMSG The Bulletin Board has been updated'
                 end
        signal READ_TAGS
        end
if fn <> '________' & fn <> '        '
   then do
        xfn = STRIP(fn,,'_')
        xft = STRIP(ft,,'_')
        'SET CMSTYPE HT'
        'STATE' xfn xft 'A'
        saverc = rc
        'SET CMSTYPE RT'
        if saverc <> 0
           then do
                call DISPLAY
'COMMAND EMSG' xfn xft 'is not on your A disk.  No action taken.'
                signal READ_TAGS
                end
        'COPYFILE' xfn xft 'A $EBBNEWS SCRIPT A (REPLACE'
        'COPYFILE $EBBNEWS $JUNK$ A $EBBNEWS $SUBJ$ A (REPLACE'
   'EXEC EBBSEND $OKAY$' func board 'non-PROFS' old_fn old_ft flash
        if rc = 0
            then do
                 if func = 'REPLACE' then signal exit
                 call INIT
                 call DISPLAY
                 'COMMAND EMSG The Bulletin Board has been updated'
                 end
        signal READ_TAGS
        end
'EXTRACT /SIZE'
'TOP'
string = ''
if SIZE.1 <> 0
   then do SIZE.1
        'DOWN 1'
        'EXTRACT /CURLINE'
        string = string CURLINE.3
        end
```

```
  word_count    = WORDS(string)
  if word_count > Ø
     then do
          'ERASE $EBBNEWS SCRIPT A'
          'SAVE $EBBNEWS SCRIPT A'
          'COPYFILE $EBBNEWS $JUNK$ A $EBBNEWS $SUBJ$ A (REPLACE'
     'EXEC EBBSEND $OKAY$' func board 'non-PROFS' old_fn old_ft flash
          if rc = Ø
              then do
                   if func = 'REPLACE' then signal exit
                   call INIT
                   call DISPLAY
                   'COMMAND EMSG The Bulletin Board has been updated'
                   end
          signal READ_TAGS
          end
/* Drop through when just adding a subject line to a board           */
  'ERASE $EBBNEWS SCRIPT A'
  'COPYFILE $EBBNEWS $JUNK$ A $EBBNEWS $SUBJ$ A (REPLACE'
  'EXEC EBBSEND $OKAY$' func board 'non-PROFS' old_fn old_ft flash
  if rc = Ø
      then do
           if func = 'REPLACE' then signal exit
           call INIT
           call DISPLAY
           'COMMAND EMSG The Bulletin Board has been updated'
           end
  signal READ_TAGS
ADD_LINE:
  'EXTRACT /CURSOR/LSCREEN/SIZE/LINE'
  cursscrn = CURSOR.1
  cursfile = CURSOR.3
  screenl  = LSCREEN.1
  filel    = SIZE.1
  saveline = LINE.1
  if cursscrn > screenl  then signal ADD_1
  if cursscrn < 2        then signal ADD_1
  if cursfile > filel    then signal ADD_1
  if cursfile < 1        then signal ADD_1
  'COMMAND :' cursfile
ADD_1:
  'ADD 1'
  'COMMAND :' saveline
  call DISPLAY
  'CURSOR SCREEN' CURSOR.1 CURSOR.2
  signal READ_TAGS
DEL_LINE:
  'EXTRACT /CURSOR/LSCREEN/SIZE/LINE'
  cursscrn = CURSOR.1
  cursfile = CURSOR.3
  screenl  = LSCREEN.1
```

```
  filel    = SIZE.1
  saveline = LINE.1
  if cursscrn > screenl  then signal NO_DEL
  if cursscrn < 2        then signal NO_DEL
  if cursfile > filel    then signal NO_DEL
  if cursfile < 1        then signal NO_DEL
  'COMMAND :' cursfile
  'DELETE 1'
  'COMMAND :' saveline
  call DISPLAY
  'CURSOR SCREEN' CURSOR.1 CURSOR.2
  signal READ_TAGS
NO_DEL:
  'EMSG Move the cursor to the detail line to delete',
           'and press PF3 again'
  call DISPLAY
  'CURSOR SCREEN' CURSOR.1 CURSOR.2
  signal READ_TAGS
VIEW:
  if cron <> '____________' & cron <> '            '
     then do
          cron8 = SUBSTR(cron,1,8)
          cron4 = SUBSTR(cron,9,4)
          'PROFS RETRIEVE' cron8 cron4 'DISK'
          cronfn = 'D' <> SUBSTR(cron,3,3) <> SUBSTR(cron,9,4)
          'SET CMSTYPE HT'
          'STATE' cronfn 'SCRIPT A'
          saverc = rc
          'SET CMSTYPE RT'
          if saverc <> Ø
             then do
                  call DISPLAY
     'COMMAND EMSG' cron 'cannot be retrieved from the PROFS database.'
                  signal READ_TAGS
                  end
          'PROFS MEMO' cronfn 'SCRIPT A'
          'ERASE' cronfn 'SCRIPT A'
          call DISPLAY
          signal READ_TAGS
          end
  if prof <> '________' & prof <> '        '
     then do
          xfn = STRIP(prof,,'_')
          'SET CMSTYPE HT'
          'STATE' xfn 'SCRIPT A'
          saverc = rc
          'SET CMSTYPE RT'
          if saverc <> Ø
             then do
                  call DISPLAY
  'COMMAND EMSG' xfn 'SCRIPT is not on your A disk.  No action taken.'
```

```
                signal READ_TAGS
                end
          'PROFS MEMO' xfn 'SCRIPT A'
          call DISPLAY
          signal READ_TAGS
          end
  if fn <> '________' & fn <> '        '
     then do
          xfn = STRIP(fn,,'_')
          xft = STRIP(ft,,'_')
          'SET CMSTYPE HT'
          'STATE' xfn xft 'A'
          saverc = rc
          'SET CMSTYPE RT'
          if saverc <> Ø
             then do
                  call DISPLAY
  'COMMAND EMSG' xfn xft 'is not on your A disk.  No action taken.'
                  signal READ_TAGS
                  end
          'PROFS OFSMOSCR' xfn xft 'A'
          call DISPLAY
          signal READ_TAGS
          end
  'EXTRACT /SIZE'
  'TOP'
  if SIZE.1 <> Ø
     then do SIZE.1
          'DOWN 1'
          'EXTRACT /CURLINE'
          string = string CURLINE.3
          end
  word_count    = WORDS(string)
  if word_count > Ø
     then do
          call DISPLAY
          signal READ_TAGS
          end
  call DISPLAY
  'COMMAND EMSG No reference file specified.  No action taken.'
  signal READ_TAGS
CALENDAR:                                  /* display calendar       */
  'EBBCAL'
  call DISPLAY
  signal READ_TAGS
FULLSCREEN:                                /* toggle fullscreen mode */
  if fullscreen = 'NO'
     then fullscreen = 'YES'
     else fullscreen = 'NO'
  call DISPLAY
  signal READ_TAGS
```

```
REFRESH:                                    /* clear all input        */
  call INIT
  call DISPLAY
  signal READ_TAGS
HELP:
  'EXEC EBBHELP EBBEDIT'
  call DISPLAY
  signal READ_TAGS
FORWARD:                                    /* scroll forward         */
  'EXTRACT /CURSOR/LSCREEN/SIZE/LINE'
  cursscrn = CURSOR.1
  cursfile = CURSOR.3
  screenl  = LSCREEN.1
  filel    = SIZE.1
  saveline = LINE.1
  if fullscreen = 'YES'
     then 'DOWN 16'
     else 'DOWN 6'
  call DISPLAY
  'CURSOR SCREEN' CURSOR.1 CURSOR.2
  if cursfile > filel | if cursfile < 1
     then 'CURSOR FILE' filel '1'
  signal READ_TAGS
BACKWARD:
  'EXTRACT /CURSOR/LSCREEN/SIZE/LINE'
  cursscrn = CURSOR.1
  cursfile = CURSOR.3
  screenl  = LSCREEN.1
  filel    = SIZE.1
  saveline = LINE.1
  if fullscreen = 'YES'
     then 'UP 16'
     else 'UP 6'
  call DISPLAY
  'CURSOR SCREEN' CURSOR.1 CURSOR.2
  if cursfile > filel | if cursfile < 1
     then 'CURSOR FILE 1 1'
  signal READ_TAGS
BADPF:
  call DISPLAY
  'COMMAND EMSG Invalid key pressed.  No action taken.'
  signal READ_TAGS
```

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

P C Shumway
Systems Analyst (USA)

Ascent Solutions has announced Version 2.1 of PKZIP, the compression and decompression software. Version 2.1 gives users the ability to compress a number of selected CMS files, both mini-disk and SFS, into a single ZIP archive. It also allows users to update the contents of an existing ZIP archive, by adding new files, updating existing files, or deleting files from the archive. The product can also translate between EBCDIC and ASCII, so it is compatible with PC and other versions of PKZIP. It can also convert record-oriented data (from VM systems) into stream-oriented data for PC and Unix machines, and *vice versa*.

For further information contact:
Ascent Solutions, 9009 Springboro Pike, Miamisburg, OH 45342, USA.
Tel: (937) 847 2374.

* * *

Sterling Software has announced Release 1.2 of VM:Webserver, its World Wide Web server for VM/ESA sites.

The new release adds functions that improve CGI (Common Gateway Interface) script processing, which simplify Web authoring and increase security. CGI scripts can run in dynamically created virtual machines, or dynamic workers, rather than in the VM:webserver virtual machine. This reduces demand on the primary server and eliminates the need for multiple Web servers on a single port, thereby simplifying TCP/IP configuration and network changes.

Security improvements result from the use of dynamic workers. A CGI script that runs on a dynamic worker assumes the security profile from the VM user identification of the CGI script owner, not that of the VM:Webserver virtual machine. This ensures that untrusted CGI scripts do not compromise system security.

SSIs (Server Side Includes) allow Web publishers to include common Web page elements, eg current date and time or headers and footers, on all Web pages by creating a single source document instead of coding them for each occurrence.

For further information contact:
Sterling Software, 1800 Alexander Bell Drive, Reston, VA 22091, USA.
Tel: (703) 264 8000.
Sterling Software International, 1 Longwalk Road, Stockley Park, Uxbridge, Middlesex, UB11 1DB, UK.
Tel: (0181) 867 8000.

* * *

Xephon is holding a conference in London on 12-13 May called *VM Update '97*.

For further information contact Xephon at any of the addresses shown on page 2.