184

MVS update

January 2002

In this issue

MVS Update

Published by
Xephon
27-35 London Road
Newbury
Berkshire RG14 1JL
England
Telephone: 01635 33598
From USA: 01144 1635 33598
E-mail: Jaimek@xephon.com

North American office
Xephon/QNA
PO Box 350100,
Westminster, CO 80035-0100
USA
Telephone: (303) 410 9344
Fax: (303) 438 0290

Contributions

Articles published in *MVS Update* are paid for at the rate of £170 ($260) per 1000 words and £100 ($160) per 100 lines of code for the first 200 lines of original material. The remaining code is paid for at the rate of £50 ($80) per 100 lines. In addition, there is a flat fee of £30 ($50) per article. To find out more about contributing an article, you can download a copy of our *Notes for Contributors* from www.xephon.com/nfc.

MVS Update on-line

Code from *MVS Update*, and complete issues in Acrobat PDF format, can be downloaded from our Web site at http://www.xephon.com/mvs; you will need to supply a word from the printed issue.

Editor
Jaime Kaminski

Disclaimer

Subscriptions and back-issues

A year's subscription to *MVS Update*, comprising twelve monthly issues, costs £340.00 in the UK; $505.00 in the USA and Canada; £346.00 in Europe; £352.00 in Australasia and Japan; and £350.00 elsewhere. In all cases the price includes postage. Individual issues, starting with the January 1992 issue, are available separately to subscribers for £29.00 ($43.50) each including postage.

Setting a PDSE alias

THE PROBLEM

There are some circumstances where an alias needs to be added to an existing member of a PDSE library. Unfortunately, the obvious solution of re-invoking the binder is not always possible because this requires the specification of all the options, which may not be available.

A SOLUTION

SETPDSEA solves this problem by loading the existing load module without binding it. The IEWBIND macro used for this purpose is described in the *DFSMS Program Management Manual*.

POSSIBLE MODIFICATIONS

In this particular application, the existing program name is used as an alias and the specified name used as the new program name. An easy change would be to add an alias rather than swapping the names – such a simple modification would involve only changing the assigned names.

DD STATEMENTS

The DD statements used are:

- SYSLMOD – the program library containing the program for which the alias is to be set.
- SYSPRINT – the report (log) file containing the list output from the binder. Depending on whether the binder was successful, the list output either contains error messages or the usual report.

The SETPDSEA code and a sample JCL are shown below.

SAMPLE PROGRAM

```
         TITLE 'Rename program name to alias and assign new name'
**
* FUNCTION:
* SETPDSEA renames the specified program and uses the original
* program name as an alias.
*
* INVOCATION:
* // EXEC SETPDSEA,PARM='aaaaaaaa pppppppp[R]'
*       aaaaaaaa = existing program name (= subsequent alias)
*       pppppppp = new program name (minimum 1 character)
*              R = (optional) switch = replace any existing alias
*
* RETURN CODE:
*   Ø = OK
*   8 = processing error
*  12 = parameter (length) error
*  16 = OPEN error
* 256+n = STOW error (n = STOW return code)
*         eg 26Ø = new name exists already
*            264 = old name does not exist
*
* DD STATEMENTS:
*  SYSLMOD  - program library that contains the program for
*             which the alias is to be set.
*  SYSPRINT - report (log) file
**
         PRINT NOGEN
SETPDSEA CSECT
SETPDSEA AMODE 31
SETPDSEA RMODE 24
         SPACE
         BAKR  R14,Ø         save registers + return address
         BASR  R12,Ø         set base register
         USING *,R12
         SPACE
         L     R2,Ø(R1)      pointer to parameter
         LH    R1,Ø(R2)      length of parameter data
         LA    RØ,2(R2)      address of parameter data
* parameter: mmmmmmmm aaaaaaaa[R]
         MVC   RC,=H'12'     preset RC: invalid parameter length
         CH    R1,=AL2(PMINLEN) test against minimum length
         JL    EXIT          too short
         CH    R1,=AL2(PMAXLEN) test against maximum length
         JH    EXIT          too long
* else parameter length OK
         LA    R2,PARM
         LA    R3,PMAXLEN
         ICM   R1,B'1ØØØ',=X'4Ø' padding byte
         MVCL  R2,RØ         move parameter to data area
```

```
* build variable-name entries
         LA    R1,VONAME     old name (becomes new alias)
         USING VDSECT,R1
         MVC   VLEN,=H'8'
         MVC   VNAME,PONAME
         LA    R1,VNNAME     new name
         USING VDSECT,R1
         MVC   VLEN,=H'8'
         MVC   VNAME,PNNAME
         SPACE
         MVC   RC,=H'16'     RC: open error
         MVC   RNAME,PNNAME
         OPEN  (SYSLMOD,(UPDAT))
         LTR   R15,R15
         JNZ   EXIT
         MVC   RC,=X'Ø1ØØ'   RC: STOW error
         STOW  SYSLMOD,PONAME,C
         STC   R15,RCRSC     append STOW RC
         CLOSE (SYSLMOD)
         CLI   RCRSC,4       test whether error occurred
         JH    EXIT          STOW error
         JL    STOWOK        STOW OK
* replace warning
         CLI   PFLAG,C'R'
         JNE   EXIT          replace not allowed
         SPACE
STOWOK   MVC   RC,=H'8'      RC: processing error
* start dialog
         IEWBIND FUNC=STARTD,                                          X
               VERSION=4,                                              X
               RETCODE=RTC,RSNCODE=RSC,                                X
               DIALOG=DTOKEN,                                          X
               FILES=FILELIST
         BAL   R9,TESTRC
         J     EXIT1         error return
* create workmod (without binding)
         IEWBIND FUNC=CREATEW,                                         X
               VERSION=4,                                              X
               RETCODE=RTC,RSNCODE=RSC,                                X
               DIALOG=DTOKEN,                                          X
               WORKMOD=WKTOKEN,                                        X
               INTENT=ACCESS
         BAL   R9,TESTRC
         J     EXIT2         error return
* include module
         IEWBIND FUNC=INCLUDE,                                         X
               VERSION=4,                                              X
               RETCODE=RTC,RSNCODE=RSC,                                X
               WORKMOD=WKTOKEN,                                        X
               INTYPE=NAME,                                            X
               ATTRIB=YES,                                             X
               DDNAME=LOADLIB,                                        X
               MEMBER=VNNAME
         BAL   R9,TESTRC
         J     EXIT2         error return
* add alias
         IEWBIND FUNC=ADDA,                                           X
               VERSION=4,                                             X
               RETCODE=RTC,RSNCODE=RSC,                               X
               WORKMOD=WKTOKEN,                                       X
               ANAME=VONAME
         BAL   R9,TESTRC
         J     EXIT2         error return
* save workmod
         IEWBIND FUNC=SAVEW,                                          X
               VERSION=4,                                             X
               RETCODE=RTC,RSNCODE=RSC,                               X
               WORKMOD=WKTOKEN,                                       X
               MODLIB=LOADLIB,                                        X
               SNAME=VNNAME,                                          X
               REPLACE=YES
         BAL   R9,TESTRC
         J     EXIT2         error return
         SPACE
         MVC   RC,=H'Ø'      reset return code
* delete workmod
EXIT2    IEWBIND FUNC=DELETEW,                                        X
               VERSION=4,                                             X
               RETCODE=RTC,RSNCODE=RSC,                               X
               WORKMOD=WKTOKEN,                                       X
               PROTECT=YES
* end dialog
EXIT1    IEWBIND FUNC=ENDD,                                           X
               VERSION=4,                                             X
               RETCODE=RTC,RSNCODE=RSC,                               X
               DIALOG=DTOKEN
         SPACE
EXIT     LH    R15,RC        set program return code
         PR    ,             program return
         SPACE 2
TESTRC   DS    ØH            test return code
* Display low-order 16-bits of RSC (reason code) as hexadecimal
* if non-zero
* Return: Ø(R9)              error
*          4(R9)             OK (no output message issued)
         LTR   R15,R15       return code
         BZ    4(R9)         OK, return
* else display reason code (hexadecimal)
N        EQU   4             field length
         UNPK  WK,RSC(N+1)   FaFbFcFdFeFgFhxx,abcdefghxx
         TR    WK,TRTAB-24Ø
         WTO   TEXT=MSG,ROUTCDE=(11)  Write To Programmer
         BR    R9             error return
TRTAB    DC    C'Ø123456789ABCDEF'
         TITLE 'Constants and work areas'
MSG      DC    H'12'
         DC    C'RSC:'
WK       DS    CL(N*2+1)
         SPACE 1
PARM     DS    ØC             EXEC parameter
PONAME   DS    CL8            old name
RNAME    DS    ØCL8
         DS    C              separator
PNNAME   DS    CL8            new name (moved to RNAME)
PFLAG    DS    CL1            switch (optional, R = replace)
PMAXLEN  EQU   *-PARM         parameter maximum length
         ORG   PNNAME
         SPACE
         DS    C              minimum alias
PMINLEN  EQU   *-PARM         parameter minimum length
         ORG   ,              reset origin counter
         SPACE
RC       DS    H              return code
RCRSC    EQU   *-1            STOW return code
RTC      DS    F              IEWBIND return code
RSC      DS    XL4            IEWBIND reason code
         SPACE
DTOKEN   DS    CL8            dialog token
WKTOKEN  DS    CL8            workarea token
         SPACE
FILELIST DC    F'1'           number of list entries
         DC    CL8'PRINT',F'8',A(PRINTX)
PRINTX   DC    CL8'SYSPRINT'
         SPACE
LOADLIB  DC    H'7',CL7'SYSLMOD'
         SPACE
VONAME   DS    H,CL8
VNNAME   DS    H,CL8
         SPACE
SYSLMOD  DCB  DDNAME=SYSLMOD,DSORG=PO,MACRF=(R,W)
         SPACE
VDSECT   DSECT
VLEN     DS    H
VNAME    DS    CL8
         SPACE
* Register equates
RØ       EQU   Ø
R1       EQU   1
R2       EQU   2
R3       EQU   3
R4       EQU   4
R5       EQU   5
R6       EQU   6
R7       EQU   7
```

```
R8       EQU   8
R9       EQU   9
R10      EQU   10
R11      EQU   11
R12      EQU   12
R13      EQU   13
R14      EQU   14
R15      EQU   15
         END
```

SAMPLE JCL

```
// EXEC PGM=SETPDSEA,
//  PARM='C00      C0A'
//*      |        |
//*      |        +— new name (1 to 8-characters)
//*      +——————— old name (8-characters, used as ALIAS)
//STEPLIB  DD DSN=LoadLibrary,DISP=SHR
//SYSLMOD  DD DSN=ProgramLibrary,DISP=SHR
//SYSPRINT DD SYSOUT=*
```

Systems Programmer (UK) © Xephon 2002

A REXX program to list GDG information

Have you ever needed to obtain information about Generation Data Groups (GDGs) quickly and easily? A few years ago, I was faced with the constant need to obtain the generation index for a GDG. I needed to look through LISTCAT output, searching for dates and generation numbers in order to find the index I needed: therefore, I wrote a REXX program to make the task easier. We were running OS/390 Version 2 Release 6 and DFSMS/MVS 1.4. The idea was to list, in one screen, all the information that I might need in order to pinpoint the dataset I wanted to use, and, just in case it might be useful, the VOLSERs that were associated with each GDG (back then I had not got any tape management software installed). In order to use this program, you just have to give it the GDG BASE name. It will create a sequential work file, and it will invoke BROWSE to view the results. If you prefer to run it in batch mode, it will write the lines to SYSTSPRT as if it were a listing. So, if you type:

```
     TSO GDG my.gdg.base.name.to.list
```

you may end up with something like this:

```
GDG NAME    ==> MY.GDG.BASE.NAME.TO.LIST

       Created ==> 2000/10/23 Limit(2) NOSCRATCH NOEMPTY

     -1 001      G0177V00   2001/07/12   H01778 H01862 H02481 H00652 H03064
        001                              H05652
      0 002      G0178V00   2001/07/13   H06191 H06192 H06193 H06194 H06197
        002                              H06212 H06213 H06214
```

You have the name in the first line (just in case you want to print it), and, in the third line, you have the creation date and limit. The SCRATCH/NOSCRATCH and EMPTY/NOEMPTY options have been added more recently. They were not needed for the initial objective.

From the fifth line onward, you will find the specific GDG information. In the first column, you will see the GDG index (what I was after, in the first place); in the second column, you will find the number relative to the limit – 1 will be the oldest generation, and it will be the first displayed. The third column has the generation creation date, and from the fourth column, until the end of the line, you will see the VOLSERs in use by that generation. They will be repeated in groups of five until all of the VOLSERs in use are listed.

```
/* REXX
                        ***********************
                        *      GDG 2.1.0      *
                        ***********************
-                                                                        */
parse upper arg gdg_name .
rk=0
zedsmsg=""
zedlmsg=""
if  gdg_name="" then
    do
        zedsmsg="No data specified"
        zedlmsg="You did not specify the name of the GDG to list"
        rk=28
   end
else
    do
        gdg_name="'"strip(gdg_name,,"'")"'"
        x=outtrap(listc.)
        address "TSO" "listcat ent("gdg_name") all"
        x=outtrap("OFF")
        if  rc=Ø then
            do
                if  subword(listc.1,1,2)="GDG BASE" then
                    do
                        call get_data
                    end
                else
                    do
                        zedsmsg="Invalid data"
                        zedlmsg=gdg_name"IS NOT A GDG BASE ENTRY"
                        rk=24
                    end
            end
        else
            do
                zedsmsg="Listcat Error"
                zedlmsg="Error ("rc") during LISTCAT of "gdg_name
                rk=rc
            end
    end
if  zedlmsg¬="" then
    do
        if  sysvar("SYSISPF")="ACTIVE" & sysvar("SYSENV")="FORE" then
            do
                address "ISPEXEC" "setmsg msg(isrzØØ1)"
            end
        else
            do
                say zedsmsg
                say zedlmsg
            end
    end
return rk
/* - - - - - - - - - - - - - */
get_data:
line.=""
g=2;
vv=Ø;
do  a=1 to listc.Ø
    if  limit="LIMIT" & pos("CREATION",listc.a)>Ø then
        do
            parse value listc.a with . crdate .
            crdate=word(translate(crdate,,"-"),2)
            crdate=jul_to_greg(space(translate(crdate,,".",Ø)))
        end
    if  pos("LIMIT",listc.a)>Ø then
        do
            parse var listc.a limit .
            parse var listc.a limit scratch empty .
            limit=word(translate(limit,,"-"),2)
            call get_gdg
        end
end
queue"    GDG Name    ==> "gdg_name
queue" "
queue"        Created ==> "crdate" Limit("limit")" scratch empty
queue" "
do  a=1 to queued()
    parse pull gdg.a
end
ln=a-1
do  a=3 to g
    ln=ln+1
    if  vrs=(vv-word(line.a,1))*-1 then
        do
            gdg.ln=left(" ",5)line.a
        end
    else
        do
            gdg.ln=format((vv-word(line.a,1))*-1,4)" "line.a
        end
    gdg.ln="   "gdg.ln
    vrs=(vv-word(line.a,1))*-1
end
if  sysvar("SYSISPF")="ACTIVE" & sysvar("SYSENV")="FORE" then
    do
        call browse_results
    end
else
    do   z=1 to ln
         say gdg.z
    end
return
/* - - - - - - - - - - - - - */
get_gdg:
do  b=a+1 to listc.Ø
    if  find(listc.b,"NONVSAM")>Ø then
        do
            vv=vv+1
            g=g+1
            p1=lastpos(".",listc.b)
            interpret "parse value listc.b with "p1+1" name ."
            call get_gdg_data
            line.g=right(vv,3,"Ø"),
                   left(" ",5)name,
                   left(" ",2)jul_to_greg(data),
                   left(" ",2)subword(vol,1,5)
            do  g1=6 by 5 while g1<=words(vol)
                g=g+1
                line.g=right(vv,3,Ø),
                       left(" ",29)subword(vol,g1,5)
            end
        end
end
return
/* - - - - - - - - - - - */
get_gdg_data:
do  c=b+1 to listc.Ø
    if  pos("CREATION",listc.c)>Ø then
        do
            parse value listc.c with "CREATION" data
            data=strip(translate(data,,"-"))
            do  d=c+1 to listc.Ø
                vol=""
                if  pos("VOLSER",listc.d)>Ø then
                    do
                        do  e=d while pos("VOLSER",listc.e)>Ø
                            parse value listc.e with "VOLSER" vlm .
                            vol=vol strip(translate(vlm,,"-"))
                        end
                        b=e-1
                        leave c
                    end
            end
        end
end
vol=space(translate(vol,,"*"))
return
/* - - - - - - - - - - - */
jul_to_greg:
parse arg j2g_data .
j2g_data=left(j2g_data,4)||right(j2g_data,3)
if  left(j2g_data,4)//4¬=Ø |,
        (left(j2g_data,4)//1ØØ=Ø & left(j2g_data,4)//4ØØ\=Ø) then
    do
        /* Normal Year */
        tab="Ø 31 59 9Ø 12Ø 151 181 212 243 273 3Ø4 334 365"
    end
else
    do
        /* Leap Year */
        tab="Ø 31 6Ø 91 121 152 182 213 244 274 3Ø5 335 366"
    end
do  mm=2
    if  right(j2g_data,3) <= word(tab,mm) then
        leave mm
end
dd=right(j2g_data,3)-word(tab,mm-1)
return left(j2g_data,4)"/"right(99+mm,2)"/"right(1ØØ+dd,2)
/* - - - - - - - - - - - */
browse_results:
br_dsn ="'"userid(),
    ||".D"date("J"),
    ||".T"space(translate(time(),,":"),Ø),
    ||".FWKØØØ'"
dd="O"time("S")
x=outtrap(info.,,"NOCONCAT")
"alloc f("dd") da("br_dsn") recfm(F B) lrecl(8Ø) space(1Ø 5)",
    "tracks new dsorg(PS)"
x=outtrap("OFF")
if  rc=Ø then
    do
        "execio "ln" diskw "dd" (finis stem gdg.)"
        if  rc=Ø then
            do
                address "ISPEXEC" "control errors return"
                address "ISPEXEC" "browse dataset("br_dsn")"
                if  rc¬=Ø then
                    do
                        zedsmsg="BROWSE error"
                        zedlmsg="Error ("rc") on "br_dsn" BROWSE"
                        rk=rc
                    end
            end
        else
            do
                zedsmsg="DISKW error"
                zedlmsg="Error ("rc") on "br_dsn" DISKW"
            end
        "free f("dd") delete"
    end
else
    do
        rk=rc
        zedsmsg="Alloc error ("rc")"
        do  i=1 to info.Ø
            zedlmsg=zedlmsg left(strip(info.i,"T"),79)
        end
    end
return
/* - - - - - - - - - - - */
```

Joao Bentes de Jesus
Systems Programmer
Mundial-Confiana, SA (Portugal)

Automated batch library updates

INTRODUCTION

This article is a development of the article published in Issue 169 of *MVS Update* from October 2000, called '*Automated and interactive library updates*'. This original article was about changing any specific strings in all members of a specific PO dataset by using ISPF panels interactively. This involved making changes to members of a single library.

Later, I thought that it would also be possible to do the same job in batch but on several libraries simultaneously, so as not to keep a TSO user busy and prevent them from doing other things. So, I developed a JCL (ZJCL), REXX program (ZREXX), and edit macro (ZMACRO) to do this. Since these sources are similar to the ones used in the previous article, you can consult that article when necessary. In addition, both the REXX and edit macro have extensive commentary lines.

HOW TO USE THE UTILITY

First of all, it is necessary to put the ZREXX and ZMACRO together in an individual user library. In this article, the name EXP.CTM.REXX is used for it. Then substitute this library name with the one you have chosen in all occurrences within the ZJCL and ZREXX members.

Note that if you choose a SYSPROC library you can eliminate 'ALTLIB' commands in the ZREXX source. Make any necessary changes on the SYSTSIN card of the JCL ZJCL and submit it. To check the result of the processing, go to spool and look at the SYSTSPRT output.

EXAMPLE

Assume that you want to apply the following two change commands to all members of the dataset EXP.CTM.TEST.

Change all ‘12345’ to ‘abcde’ between the columns 1 and 80:

```
C ‘12345' ‘abcde' CHARS 1 8Ø
```

Change all prefixes ‘SIS’ to ‘SYS1’ between the columns 1 and 80:

```
C 'SIS' 'SYS1' PREFIX 1 8Ø
```

So, what you have to do is enter the following lines for SYSTSIN card:

```
//SYSTSIN  DD   *
 ISPSTART +
 CMD(%ZREXX '12345' 'abcde' EXP.CTM.TEST CHARS  1 8Ø) +
 BATSCRW(132) BATSCRD(27) BREDIMAX(3) BDISPMAX(99999999)
 ISPSTART +
 CMD(%ZREXX 'SIS'   'SYS1'  EXP.CTM.TEST PREFIX 1 8Ø) +
 BATSCRW(132) BATSCRD(27) BREDIMAX(3) BDISPMAX(99999999)
```

On completion of the ZJCL, you would obtain output similar to this:

```
The following Change command will be issued for members of dataset:
EXP.CTM.TEST
=================================================================
C  '12345' 'ABCDE' CHARS 1 8Ø
Lrecl of the dataset is :  8Ø


MEMBER   CHG ERR DESCRIPTION-1     RC1 RC2 DESCRIPTION-2              COL2 MODE
======== === === ================= === === ========================== ==== ====
MEMBER1   15   Ø Member updated...   Ø   Ø Change macro command Rc=Ø.   8Ø OFF
MEM2     556   Ø Member updated...   Ø   Ø Change macro command Rc=Ø.   72 ON
MEM3       Ø   Ø Member not saved.   4   4 The string is not found...   8Ø OFF


NOTE: The COL2 field shows the Column2 value that was used in the
      Change command. It may be different from the value entered
      by the user. (Numbering mode situation.)
```

SOME NOTES ON THE UTILITY

As with the article published in Issue 169 of *MVS Update*, the use of edit macros is instrumental in updating libraries. However, this time we will use an edit macro in batch. To do so, we have to allocate all of the necessary ISPF libraries (ISPPLIB, ISPMLIB, ISPSLIB, ISPTLIB, and ISPPROF) to build a valid ISPF environment and we have to use the ISPSTART command to call the REXX program which has a link to that edit macro. Please have a look at the JCL ZJCL.

Note that if you submit the following job to run a REXX which uses an edit macro, you will experience some unexpected errors. This is because the job runs the REXX EXEC in TSO/E, not in an ISPF environment.

```
//TSOBATCH  EXEC PGM=IKJEFTØ1,DYNAMNBR=3Ø,REGION=4Ø96K
//SYSEXEC   DD   DSN=USERID.MYREXX.EXEC,DISP=SHR
//SYSTSPRT  DD   SYSOUT=A
//SYSTSIN   DD   *
  %REXX2
```

ZJCL

```
//INEXØØ41 JOB  MSGCLASS=X,MSGLEVEL=(1,1),CLASS=D
//*-----------------------------------------------------------------*/
//* AUTOMATED AND BATCH LIBRARY UPDATES                               */
//*                                                                   */
//* JCL               : ZJCL                                          */
//* REXX EXEC called  : ZREXX                                         */
//*                                                                   */
//*-----------------------------------------------------------------*/
//STEP1    EXEC PGM=IKJEFTØ1,DYNAMNBR=3Ø,REGION=35M
//SYSPROC  DD   DISP=SHR,DSN=EXP.CTM.REXX
//         DD   DISP=SHR,DSN=SIS.EXEC
//ISPPLIB  DD   DISP=SHR,DSN=ISP.SISPPENU
//ISPMLIB  DD   DISP=SHR,DSN=ISP.SISPMENU
//ISPSLIB  DD   DISP=SHR,DSN=ISP.SISPSENU
//ISPTLIB  DD   DISP=SHR,DSN=ISP.SISPTENU
//ISPPROF  DD   DISP=(NEW,DELETE,DELETE),DSN=&&PROF,UNIT=SYSDA,
// DCB=(ISP.SISPTENU),SPACE=(TRK,(1,1,1))
//SYSUDUMP DD   DUMMY
//ISPLOG   DD   SYSOUT=(,),DCB=(LRECL=125,BLKSIZE=129,RECFM=VA)
//SYSTSPRT DD   SYSOUT=A
//SYSTSIN  DD   *
ISPSTART +
CMD(%ZREXX 'abcde' '12345' EXP.CTM.TEST CHARS  1 8Ø) +
BATSCRW(132) BATSCRD(27) BREDIMAX(3) BDISPMAX(99999999)
ISPSTART +
CMD(%ZREXX 'SIS'   'SYS1'  EXP.CTM.TEST PREFIX 1 8Ø) +
BATSCRW(132) BATSCRD(27) BREDIMAX(3) BDISPMAX(99999999)
```

ZREXX

```
/*REXX*/
/*------------------------------------------------------------------*/
/* AUTOMATED AND BATCH LIBRARY UPDATES                                */
/*                                                                    */
/* REXX EXEC         : ZREXX                                          */
/* Edit macro called : ZMAC                                           */
/*                                                                    */
/* Important variables used in the REXX and macro:                   */
/* ==============================================                    */
/*                                                                   */
/*  Dsn   : Dataset of which members are to be updated. It will be   */
/*          referred as the "SOURCE dataset" in the commentary lines. */
/*                                                                   */
/*  From,To    : 'Isredit Change' command strings.                  */
/*  Col1, Col2 : 'Isredit Change' command boundaries.               */
/*  Qual  : Limiting keyword. It can be CHARS /PREFIX /SUFFIX /WORD . */
/*                                                                   */
/*  Mem   : Member to be updated by the edit macro.                  */
/*  Cnt   : Number of members in the Po dataset.                     */
/*  Rc1   : 'Ispexec Edit Dataset..' command Rc for a member.        */
/*  Msg1  : Short description for Rc1.                               */
/*  Rc2   : 'Isredit Change' command Rc.                             */
/*  Msg2  : Short description for Rc2.                               */
/*  Rc3   : 'Isredit Save' command Rc.                               */
/*  Chg   : Number of changes in a member.                           */
/*  Err   : Number of errors  in a member.                           */
/*  C2    : Real Column2 value to be used in the "Change" command.   */
/*          (If numbering mode is "ON" and the Column2 value that is  */
/*          entered by user is greater than Lrecl-8 value, then edit  */
/*          macro assumes Column2 as Lrecl-8. Because if the         */
/*          numbering mode is 'ON' for a member, then that member     */
/*          will have an 8-digit sequence number on the right-end    */
/*          side.)                                                   */
/*  Mod   : Numbering mode of a member.                              */
/*                                                                   */
/*-------------------------------------------------------------------*/
Arg From To Dsn Qual Col1 Col2
"Prof nopref"
"Ispexec Control Errors Return"   /*   Let EXEC process errors.       */

Say " "
Say " "
Say "The following Change command will be issued for members of the
dataset:"
Say Dsn
Say
"================================================================="
Say "C " From"" ""To"" Qual Col1 Col2
Say " "

/*-------------------------------------------------------------------*/
/* Allocate the library which has the edit macro.                    */
/*-------------------------------------------------------------------*/
"Altlib  Activate Application(Exec) Da(Exp.Ctm.Rexx)"

/*-------------------------------------------------------------------*/
/* Clean profile variables.                                          */
```

```
/*------------------------------------------------------------------*/
"Ispexec Verase (From,To,Qual,Col1,Col2)"
"Ispexec Verase (C2,Mod,Rc2,Rc3,Chg,Err,Msg2,L2,Mem)"

/*------------------------------------------------------------------*/
/* Put variables into the pool so that the edit macro can use them.   */
/*------------------------------------------------------------------*/
"Ispexec Vput (From,To,Qual,Col1,Col2) Profile"

/*------------------------------------------------------------------*/
/* Check if the SOURCE dataset is an existing PO dataset.           */
/*------------------------------------------------------------------*/
IF Sysdsn(Dsn) = 'OK' Then
Do
  X = Listdsi(Dsn)
  If X <> Ø Then
  Say 'Some Listdsi info not available. Function code =' X
  Else
   Do
    Dsorg   = Sysdsorg
    If Dsorg <> PO Then
     Do
      Say 'Dataset is not a PO dataset.'
      Exit 9Ø
     End
   End
End
Else Do
       Say 'Dataset does not exist.'
       Exit 91
      End

/*------------------------------------------------------------------*/
/* Check if the SOURCE dataset has any members?
*/
/*------------------------------------------------------------------*/
x = Outtrap('Var.')
"Listds" Dsn "Members"
x = Outtrap('Off')        /* Turns trapping OFF */
Cnt = Var.Ø-6

If Cnt = Ø Then
Do
  Say 'PO dataset has no members.'
  Exit 92
End

/*------------------------------------------------------------------*/
/* Check that if From or To string that are greater than the logical  */
/* record length of the SOURCE dataset.                             */
```

```
/*------------------------------------------------------------------*/
L2    = Syslrecl
"Ispexec Vput L2 Profile"

Say "Lrecl of the dataset is : " L2
Say " "
Say " "
Say " "
Say " "

If Length(From) > L2 Then
Do
  Say 'FROM string > Lrecl of the PO dataset.'
  Exit 93
End

If Length(To) > L2 Then
Do
  Say 'TO   string > Lrecl of the PO dataset.'
  Exit 94
End

/*------------------------------------------------------------------*/
/* Check that if Col1 or Col2 is greater then the Lrecl of the SOURCE */
/* dataset. If so, do not continue.                                 */
/*------------------------------------------------------------------*/
If (Col2 > L2) 3 (Col1 > L2) Then
Do
  Say 'Col1 or Col2 value cannot be bigger than Lrecl of the SOURCE data
set.'
  Say 'Col1= ' Col1 'Col2 = ' Col2
  Exit 95
End

/*------------------------------------------------------------------*/
/* Compress SOURCE dataset to prevent it from producing an S37 abend. */
/*------------------------------------------------------------------*/
Call Compress Dsn

/*------------------------------------------------------------------*/
/* Write the heading for the report.                                */
/*------------------------------------------------------------------*/
SAY "MEMBER   CHG  ERR DESCRIPTION-1     RC1 RC2 DESCRIPTION-2
COL2
MODE"
Say "======== ==== === ================ === ===
==========================
====
===="
```

```
/*-----------------------------------------------------------------------*/
/* Get members of the SOURCE dataset and execute the edit macro for   */
/* each member in a loop.                                              */
/*-----------------------------------------------------------------------*/
Do i = 7  To Var.Ø    /* Loop-Martapv */
  Mem = Strip(Var.i)
  "Ispexec Vput Mem            Profile"  /* Member_name into profile  */
  Call Editmac                           /* Update the current member */
  Rc1 = Result
  "Ispexec Vget (Chg,Err)      Profile"  /* Get change & error number */
  "Ispexec Vget (C2,Mod)       Profile"  /* Real Column2 and Num. mode*/
  "Ispexec Vget (Msg2,Rc2,Rc3) Profile"  /* Get "Change" and "Save"   */
                                         /*   command Return codes.   */
  Select
   When (Rc1 = Ø )           Then Msg1= 'Member updated...'
   When ((Rc1=4) & (Rc3>=4)) Then Msg1= 'No save.Abend S37'
   When (Rc1 = 4 )           Then Msg1= 'Member not saved.'
   When (Rc1 = 14)           Then Msg1= 'Member in use....'
   When (Rc1 = 2Ø)           Then Msg1= 'Severe error.....'
   Otherwise                 Nop
  End

/*-----------------------------------------------------------------------*/
/* Set the ISPF statistics for the SOURCE dataset members that are    */
/* updated by the edit macro. Changed members will have the "ZMAC"    */
/* string in their ID field.                                           */
/*-----------------------------------------------------------------------*/
If Rc3 = Ø Then
Do
  "Alloc Fi(Stats)  Da("Dsn")     Shr Reuse"
  "Ispexec Lminit   Dataid(Sta)   DDname(Stats) Enq(Shr)"
  "Ispexec Lmmstats Dataid("Sta") Member("Mem") User(ZMAC)"
  "Ispexec Lmfree   Dataid("Sta")"
End

/*-----------------------------------------------------------------------*/
/* Display the change result.                                          */
/*-----------------------------------------------------------------------*/
Mem2 = OVERLAY(Mem,'        ',1)
Say  Mem2 Format(Chg,4) Format(Err,3) Msg1 Format(Rc1,3) Format(Rc2,3),
  Msg2 Format(C2,4) Mod
END  /* End-of-Loop-Martapv */

Say " "
Say "NOTE: The COL2 field shows the Column2 value that was used in the"
Say "      Change command. It may be different from the value entered "
Say "      by the user. (Numbering mode situation.)                   "
Say " "
Say " "
```

```
"Ispexec Lmfree Dataid("Villar")"
"Altlib Deactivate Application(Exec)"
EXIT         /* End-of-the-REXX-ZREXX */

EDITMAC:
/*----------------------------------------------------------------------*/
/* Control if the member is still there.                                */
/*----------------------------------------------------------------------*/
Rs = Sysdsn(Dsn"("Mem")")
If Rs="OK" Then Nop
           Else Do
                  Say 'The specified member is not found in dataset.'
                  Return
                End

"Ispexec Edit Dataset('"Dsn"("Mem")') Macro(ZMAC)"
RETURN RC /* End-of-the-procedure-EDITMAC */

COMPRESS:
/*----------------------------------------------------------------------*/
/* Allocate files needed by Iebcopy.                                    */
/*----------------------------------------------------------------------*/
Arg Dsn
"Alloc Fi(Input)    Da("Dsn") Shr Reuse"
"Alloc Fi(Output)   Da("Dsn") Shr Reuse"
"Alloc Fi(Sysout)   Dummy Reuse"
"Alloc Fi(Sysprint) Dummy Reuse"
"Alloc Fi(Sysut1)   Unit(VIO) Space(1,1) Cyl New Delete Reuse"
"Alloc Fi(Sysut2)   Unit(VIO) Space(1,1) Cyl New Delete Reuse"
"Alloc Fi(Sysut3)   Unit(VIO) Space(1,1) Cyl New Delete Reuse"
"Alloc Fi(Sysut4)   Unit(VIO) Space(1,1) Cyl New Delete Reuse"

/*----------------------------------------------------------------------*/
/* Build the SYSIN control statements for the Iebcopy utility.          */
/*----------------------------------------------------------------------*/
"Alloc Fi(Sysin)    Unit(VIO) Space(1,Ø) Blksize(8Ø) Lrecl(8Ø),
Recfm(F B) Dsorg(PS) New Delete Reuse"

"Ispexec Lminit Dataid(Martapv) Ddname(Sysin) Enq(Exclu)"
Card = ' COPY OUTDD=OUTPUT,INDD=INPUT'
"Ispexec Lmopen  Dataid("Martapv") Option(Output)"
"Ispexec Lmput   Dataid("Martapv") Dataloc(Card) Datalen(8Ø)
Mode(Invar)"
"Ispexec Lmclose Dataid("Martapv")"
"Ispexec Lmfree  Dataid("Martapv")"
"Ispexec Select Pgm(IEBCOPY)"
Rc4 = Rc
If Rc4 <> Ø Then
Do
  Say 'Iebcopy Return Code =' Rc4
  Exit
End
/*------------------------------------------------------------------*/
/* Free all DDnames of Iebcopy.                                     */
/*------------------------------------------------------------------*/
"Free File(Input,Output,Sysin,Sysprint,Sysut1,Sysut2,Sysut3,Sysut4)"
RETURN                   /* End-of_procedure_COMPRESS */
```

ZMAC EDIT MACRO

```
"Isredit Macro"
/*------------------------------------------------------------------*/
/* AUTOMATED AND BATCH LIBRARY UPDATES                              */
/*                                                                  */
/* Edit macro        : ZMAC                                         */
/* Called from       : ZREXX                                        */
/* Purpose           : Change strings in a PO member.               */
/*------------------------------------------------------------------*/
Status = Msg('Off')

"Ispexec Control Errors Return"    /* Let EXEC process errors.       */
Chg = Ø; Err = Ø                   /* Change-count, Error-count = Ø   */

/*------------------------------------------------------------------*/
/* Get the Number Mode.                                             */
/* Edit macro has to take into account that if Number_Mode is 'ON'    */
/* then the number of editable characters is Lrecl-8. Because right-  */
/* end side will include "sequence numbers"  (NUMBER FIELDS).         */
/* So if the user enters a Column2 value that is bigger than the      */
/* value of Lrecl-8, It is modified by this edit macro so that we     */
/* will prevent errors.
*/
/*------------------------------------------------------------------*/
"Isredit (Mode) = Number"
Mod = Mode

/*------------------------------------------------------------------*/
/* Get variables from the Profile Variable Pool. These are the        */
/* strings which will be used on the 'ISPF Change' edit command.      */
/*------------------------------------------------------------------*/
"Ispexec Vget (From,To,Qual,Col1,Col2,L2,Mem) Profile"
C2 = Col2
If ((Mode = ON) & (Col2>=(L2-8)))  Then  C2 = L2-8
/*------------------------------------------------------------------*/
/* Build the change command.                                        */
/*------------------------------------------------------------------*/
If Length(From) = 1 Then
          "Isredit Change" "'"From"'" To "ALL" Qual Col1 C2
                    Else
          "Isredit Change" From To "ALL" Qual Col1 C2
          Rc2 = Rc

"Isredit (Chg,Err) = CHANGE_COUNTS"
Chg = Abs(Chg)
Err = Abs(Err)

If Rc2 = 4 Then Msg2='The string is not found...'
If Rc2 = 8 Then Msg2='Str2 is longer than Str1..'
If Rc2 =12 Then Msg2='Inconsistent parameters...'
If Rc2 =2Ø Then Msg2='Severe error.............'
If Rc2 = Ø Then
Do
  Msg2='Change macro command Rc=Ø.'

  /*-------------------------------------------------------------------*/
  /* If Error-Count is Ø, then we can save the current member in the  */
  /* SOURCE dataset.                                                   */
  /*                                                                   */
  /* NOTE: If there is at least one string that could not be changed  */
  /*       by the change command, the member is not saved.            */
  /*       However, if you want to save the member anyway, simply     */
  /*       remove the following "IF statement" and leave only the     */
  /*       "Save" command there.                                      */
  /*-------------------------------------------------------------------*/
  If Err= Ø Then
   Do
    "Isredit Save"
    Rc3 = Rc   /* Rc3 = Save Return Code */
   End
End

"Ispexec Vput (Chg,Err,C2,Mod,Msg2,Rc2,Rc3) Profile"

/*---------------------------------------------------------------------*/
/* No matter if it was saved or not, exit from the member to be able  */
/* to process the next member in the SOURCE dataset.                  */
/*---------------------------------------------------------------------*/
"Isredit Cancel"
Return
EXIT /* End-of-edit-macro-ZMAC */
```

Atalay Gul
MVS System Programmer
Gas Natural Informatica SA (Spain)

Data compression

We have a private company network with a central mainframe running OS/390. This has dozens of subhosts, some of which run OS/390, others run VSE/SP under VM, Windows NT, or Unix. Data processing is divided between the host and subhosts, so a huge quantity of data is constantly transferred between systems. During peak times a single file transfer can last more than an hour. That is why we decided to implement routines for data compression. The two most appropriate algorithms for compression are:

- *Huffman's coding*: the author David Huffman first presented this in 1953. The aim of Huffman coding is to provide shorter codes to bytes.
- *LZ77*: the authors Abraham Lempel and Jacob Ziv first presented LZ77 in 1977. It is a dictionary-based scheme for text compression based on the principle that when you find a match (a group of bytes that have already been seen in the input file), instead of writing those bytes you write the offset and the length of the repetition.

These algorithms are fast, easy to implement, and free of patents; that is why we decided to use them. This is also why many well known PC programs for compressing data, such as ARJ and ZIP, implement them.

The following compressing ratios were obtained from typical banking data:

65%-92% – LZ77

45%-85% – Huffman coding

75%-97% – amalgamation of both algorithms.

After comparing the compression ratio for different types of data we concluded that the best compression results were achieved by implementing the LZ77 algorithm first, and the Huffman code second. That is why we have used this order of implementation in the compression module.

The modules for compression and decompression are designed as subroutines because we have a CICS application for transferring data between the host and subhosts. Communication buffers are sent to compression subroutines before sending. These subroutines significantly decrease their size.

For testing purposes we developed two main programs that call the subroutines. These programs are COMPRESC for compressing and DECOMPRC for decompressing. They are used in the sample job in this article to demonstrate data compression. They can also be used if you want to perform compression for archive purposes.

The COMPRESC program has three input parameters with the following meanings:

- T – text data in input dataset
- B – binary data in input dataset
- input dataset/output dataset.

The first parameter is essential when you send or receive data between platforms with different code standards (EBCDIC or ASCII). If the first parameter has the value B, we treat the data as binary and just compress it. When a parameter has the value T, we translate from EBCDIC to ASCII and *vice versa* using suitable translation tables.

The DECOMPRC program has two parameters that describe input and output datasets. It does not require parameters that describe data types because the decompression modules recognize the type of compression from control information that is incorporated in the compressed data. This enables the phased implementation of modules in the communication programs. You have to implement decompression module in programs for receiving data first. They do not change received data until you implement the compression module on the sending side.

Modules can be implemented on different platforms without any change to the source code. The only change required is the modification of translation tables in line with your standards.

Before the modules were implemented in our communication programs we tested them on different types of data using COMPRESC and

DECOMPRC. This was done in the following way:

1 Compression and decompression of selected data on the same platform, then comparison of the source file and decompressed file.

2 File transfer of selected data to another platform and repeat the process from point 1.

3 Data compressed on one platform moved to another platform with standard file transfer binary, decompressed there, and the decompressed data with the source data on that platform.

In all cases the source data and decompressed data were identical.

Note that some PC editors or emulation file transfers add a character to the end of the file that has the hex value 1A in ASCII. C routines do not read that character when the dataset is opened as text (T value of the first parameter). That makes the decompressed data length shorter by 1 byte. If the data processed is binary (B value) or in EBCDIC this difference does not exist.

After implementing the modules on all our platforms our largest file transfer now lasts less then ten minutes.

COMPRESSION SUBROUTINE

```
/*********************************************************************
* CoLZHFc - Compression using LZ77 + Huffman algorithm
*
* - Input is the area for compressing, it is a maximum of 64KB long
* - There are three input parameters:
*   1  type of data (T-Text or B-Binary)
*   2  is area length
*   3  is the address of area for compressing
* - compressed data is returned in the same area as the data for
*   compressing
*
* - Subroutine returns following return codes:
*   rc = 0  compression successfully finished,
*           input BufSource area is overwritten by compressed data
*   rc = 2  effects of compression is insignificant
*   rc = 20 insufficient work storage
*********************************************************************/
#include <stdio.h>
```

```
 #include <string.h>
 #include <stdlib.h>
 #define  UNS unsigned
 #define  SH   short

 /*--- Declaration of local procedure --------------------------------*/
 void AddToIndex(UNS int, UNS int);
 void concatenate_bits (long, UNS char *, UNS int, UNS char *);
 UNS SH int ComLZc ( UNS SH int *, UNS char *);
 UNS SH int ComHFc ( UNS SH int *, UNS char *);

#ifdef DEBUGH
  void BitToChar (int, UNS char *, UNS char *);
  UNS char BufWork[256];
#endif

 /*--- Declaration of global structures and variables ----------------*/
 typedef struct keys
  {
     struct keys *pNext;
     UNS SH int Pos;
  } key;

 typedef struct
  {
     key *pFirstInDictionary;
     key *pLastInDictionary;
  } char_index;

 char_index *CharIndex;
 key    *pDictionary;
 UNS char  *BufComp;

 UNS SH int CoLZHFc (char TypeData,
                     UNS SH int *LenBufSource,
                     UNS char *BufSource)
 {
  UNS SH int i, rc1=0, rc2=0, LenBufS;

  struct
  {
     UNS SH int LenBufSource;
     UNS SH int LenBufCompress;
     char TypeCode;         /* Code of Blank 0x40-EBCIDIC, 0x20-ASCII */
     char TypeData;         /* 1 - Text,  0 - Binary */
     char TypeCompress;     /* 0x0f - LZ77, 0xF0 - HUFF, 0xFF - Both  */
     char CR;               /* \r - Carriage Return */
     char LF;               /* \n - Line Feed  */
     char Algoritam[4];     /* Identification of compress algorithm */
     char LRC;              /* LRC */
  } control = {Ø,Ø,' ',ØxØØ,ØxØØ,'\r','\n',"\xØØ\xØe\xØd\xff",ØxØØ};

  control.TypeData = (toupper(TypeData) == 'T') ? 1:Ø;
  if (*LenBufSource < 16) return(2);
 /*--- Allocation of the work areas --------------------------------*/
  LenBufS = *LenBufSource;
  BufComp = (UNS char *) calloc(LenBufS,sizeof(char));
  if (BufComp != NULL)
  {
 /*--- Copy control information and information about offsets and ---*/
 /*--- lengths of matches at the beginning of compressed area ------*/

     rc1 = ComLZc (&LenBufS,BufSource);
     if (rc1 == Ø)
         control.TypeCompress   = '\xfØ';

     rc2 = ComHFc (&LenBufS,BufSource);
     if (rc2 == Ø)
         control.TypeCompress   != '\xØf';

     if (LenBufS + sizeof(control) < *LenBufSource)
     {
         control.LenBufSource   = *LenBufSource;
         control.LenBufCompress = LenBufS;
         memcpy(&BufSource[LenBufS],&control,sizeof(control));
         LenBufS += sizeof(control);
         for(control.LRC = ØxØØ, i=Ø; i < LenBufS; i++)
             control.LRC ¬= BufSource[i];
         BufSource[LenBufS-1] = control.LRC;
         *LenBufSource = LenBufS;
     }
     free(BufComp);
  }
  else
     rc1 = 2Ø;
  return((rc1<<8)|rc2);
 }

 /*******************************************************************
 * ComLZc - Compression using the LZ77 algorithm
 *******************************************************************/
 UNS SH int ComLZc ( UNS SH int *LenBufSource, UNS char *BufSource)
 {
  UNS char   *pBufC, *pChT, *pChP, *pBufEnd;
  UNS SH int it, ip, i, j, m, rc, TotalLess;

  struct
  {
    UNS SH int LenBufS;
    UNS SH int NoPosLen;
  } control = {0,0};

  typedef struct
  {
     UNS SH int Curr_Pos;
     UNS SH int Pos;
     UNS SH int Len;
  } pos_len;

  key    *Dictionary, *pr;
  pos_len pl, *PosLen, *ppl;

  rc = 0;
  control.LenBufS  = *LenBufSource;
 /*--- Allocation of the work areas ---------------------------------*/
  Dictionary = (key *) calloc(control.LenBufS,sizeof(key));
  if (Dictionary == NULL) return(20);
  PosLen = (pos_len *) calloc(control.LenBufS,sizeof(pos_len));
  if (PosLen == NULL) return(20);
  CharIndex = (char_index *) calloc(256,sizeof(char_index));
  if (CharIndex == NULL) return(20);

  pBufEnd = &BufSource[control.LenBufS];
  ppl = PosLen;
  pDictionary = Dictionary;

  /*--- Find the match in the previous string -------------------*/
  /*--- and note its current position, offset and length --------*/

  TotalLess = it = control.NoPosLen = 0;
  do
  {
     pl.Len=1;
     m = BufSource[it];
     pr = CharIndex[m].pFirstInDictionary;
     if (pr == NULL)
        AddToIndex(m,it);
     else
     {
        pl.Pos=0;
        do
        {
            ip = pr->Pos;
            pChP = &BufSource[ip];
            pChT = &BufSource[it];
            if (BufSource[ip+10] == BufSource[it+10])
            {
               for(j=ip; *pChP++ == *pChT++ && pChT <= pBufEnd; j++);

               j -= ip;
```

```
               if (j > pl.Len)
               {
                 pl.Curr_Pos = it;
                 pl.Pos = ip;
                 pl.Len = j;
               }
            }
            pr = pr->pNext;
        } while(pr != NULL);

        if (pl.Len > 9)
        {
           ppl->Curr_Pos = pl.Curr_Pos;
           ppl->Pos = pl.Pos;
           ppl->Len = pl.Len;
           ppl++;
           control.NoPosLen++;
           TotalLess += pl.Len - 9;
        }

        for(i=it, j=it + pl.Len; i < j ; i++)
        {
           m = BufSource[i];
           AddToIndex(m,i);
        }
     }
     it = it + pl.Len;
 } while(it < control.LenBufS);

#ifdef DEBUGL
 printf("\n ComLZc lens %i nopl %i",
          control.LenBufS,control.NoPosLen);
#endif
 if (control.NoPosLen > 0 && TotalLess > 10)
 {
    pBufC = BufComp;
    memcpy(pBufC,&control,sizeof(control));
    pBufC += sizeof(control);

    i = control.NoPosLen * sizeof(pos_len);
    memcpy(pBufC,PosLen,i);
    pBufC += i;

    /*--- Copy parts that are different in compressed area ---------*/
    for(i=0,ppl=PosLen,m=0; i<control.NoPosLen; i++,ppl++)
    {
#ifdef DEBUGL
 printf("\n %2i.ComLZc curr_pos %5i pos %5i len %5i",
         i,ppl->Curr_Pos,ppl->Pos,ppl->Len);
#endif
```

```
      j=ppl->Curr_Pos - m;
      memcpy(pBufC,&BufSource[m],j);
      pBufC += j;
      m = ppl->Curr_Pos + ppl->Len;
    }

    if (m < control.LenBufS)
    {
      j = control.LenBufS - m;
      memcpy(pBufC,&BufSource[m],j);
      pBufC += j;
    }
/*--- Copy compressed area into input-output area ----------------*/
    m = (UNS SH int) (pBufC - BufComp);
    if (m >= control.LenBufS)
        rc=2;
    else
    {
       *LenBufSource = m;
       memcpy(BufSource,BufComp,m);
    }
}
else rc = 2;
/*--- Free working areas -------------------------------------------*/
 free(Dictionary);
 free(PosLen);
 free(CharIndex);

 return(rc);
}

/*****************************************************************/
void AddToIndex(UNS int m, UNS int it)
{
  if (CharIndex[m].pFirstInDictionary == NULL)
     CharIndex[m].pFirstInDictionary = pDictionary;
  else
     CharIndex[m].pLastInDictionary->pNext = pDictionary;
  CharIndex[m].pLastInDictionary = pDictionary;
  pDictionary->Pos = it;
  pDictionary++;
}

/******************************************************************
* ComHFc - Compression using the Huffman algorithm
*****************************************************************/
UNS SH int ComHFc( UNS SH int *LenBufSource, UNS char BufSource[])
{
 UNS char *pBufS;
```

```
  struct
  {
    UNS SH int LenBufS;
    UNS SH int NoHuffCodes;
  } control = {Ø,Ø};

  struct velocity
  {  UNS char     Ch;
     UNS SH int   NoCh;
  } Velocity[256];

  struct velocity pom;

  typedef struct Code
  {
     UNS SH int   NoBitHofCode;
     UNS char     HofCode[32];
  };

  struct Code  *IndexCode[256];
  struct Code  *pCode, *pCodeRoot, *pp;

  long NoBit=Ø, NoBitSource, il;
  SH int  rc, i, j, k;
  UNS SH int  m, kk;

  UNS char BitMaps[8]={Øx8Ø,Øx4Ø,Øx2Ø,Øx1Ø,ØxØ8,ØxØ4,ØxØ2,ØxØ1};

#ifdef DEBUGH
  printf("\n ComHFc >> Len buf source = %i",*LenBufSource);
#endif

  rc=Ø;
  control.LenBufS = *LenBufSource;
  if (control.LenBufS < 16) return(2) ;
  /*--------- Allocation of working areas ----------------------------*/
  memset(BufComp,Ø,control.LenBufS);
  memset(Velocity,Ø,sizeof(Velocity));
  memset(IndexCode,Ø,sizeof(IndexCode));

  /*--------- Scan area for compressing and count frequency of chrs --*/
  for(m=Ø; m < control.LenBufS; m++)
      Velocity[BufSource[m]].NoCh++;

  for(control.NoHuffCodes=Ø, m= Ø; m <= 255; m++)
    if (Velocity[m].NoCh)
    {
         Velocity[control.NoHuffCodes].Ch = (UNS char) m;
         Velocity[control.NoHuffCodes].NoCh=Velocity[m].NoCh;
         control.NoHuffCodes++;
    }
  control.NoHuffCodes--;

  if (control.LenBufS > control.NoHuffCodes * 2 + 15)
  {
   kk = sizeof(control);
   memcpy(BufComp,&control,kk);

  if (control.NoHuffCodes == Ø)
  {
       BufComp[kk] = Velocity[Ø].Ch;
       NoBit = 8Ø;
  }
  else
  {
    k = control.NoHuffCodes+1;
    pCodeRoot = (struct Code *) calloc(k,sizeof(struct Code));
    if (pCodeRoot == NULL) return(2Ø);

    for(i=Ø, pCode=pCodeRoot; i <= control.NoHuffCodes; i++, pCode++)
         IndexCode[(UNS SH int) Velocity[i].Ch] = pCode;

  /*----------- Generate optimal code - Huffman's code -------------*/
  /*-- Sort character set by frequency of apperance-shell sort ------*/
    for(k=control.NoHuffCodes/2;k > Ø; k /= 2)
       for(i=k; i <= control.NoHuffCodes; i++)
          for(j=i-k;
              j >= Ø && Velocity[j].NoCh < Velocity[j+k].NoCh;
              j -= k)
          {
              pom = Velocity[j];
              Velocity[j] = Velocity[j+k];
              Velocity[j+k] = pom;
          }

  /*-------- Preparation for generating binary Huffman's code -------*/
    kk += control.NoHuffCodes*2 - 1;
    for(k=control.NoHuffCodes; k >= 1; k--)
    {
       pom = Velocity[k];
       for (i=k-1; i >= Ø && Velocity[i].NoCh < pom.NoCh;
            i--)
            Velocity[i+1] = Velocity[i];
       Velocity[i+1]=pom;
       Velocity[k-1].NoCh += Velocity[k].NoCh;
       BufComp[kk--] = Velocity[k-1].Ch;
       BufComp[kk--] = Velocity[k].Ch;
    }
    kk++;
    for(i=1; i <= control.NoHuffCodes; i++, kk += 2)
    {
       pp=IndexCode[(UNS SH int) BufComp[kk+1]];
       pCode=IndexCode[(UNS SH int) BufComp[kk]];
       j=pp->NoBitHofCode;
       k=j/8;
       j = j - k*8;
       memcpy(pCode->HofCode,pp->HofCode,k+1);
       pCode->HofCode[k] != BitMaps[j];
       pCode->NoBitHofCode = ++(pp->NoBitHofCode);
    }

#ifdef DEBUGH
  /*----------------- Print of the Huffman's code -----------------*/
    printf("\n ComHFc Compressed length %i",(NoBit+7)/8);
    printf("\n ComHFc Print of Huffman codes %i",control.NoHuffCodes);
    for(i=Ø; i <= control.NoHuffCodes; i++)
    {
       pCode = IndexCode[(UNS SH int) Velocity[i].Ch];
       BitToChar(pCode->NoBitHofCode,pCode->HofCode,BufWork);
       printf("\n %i. c=%c %Ø2x lcode=%i code=%s",
         i,Velocity[i].Ch,Velocity[i].Ch,pCode->NoBitHofCode,BufWork);
    }
#endif

  /*-------------------- Encipher input file ----------------------*/
    NoBitSource= control.LenBufS * 8l;
    for(il=Ø, NoBit = kk*8l ,pBufS = BufSource;
        il < control.LenBufS && NoBit < NoBitSource;
        il++, pBufS++)
      {
#ifdef DEBUGH
       printf("\r\n ComHFc >==> %i >%c< >%Ø2x<",NoBit,*pBufS,*pBufS);
#endif
       pCode = IndexCode[(UNS SH int) *pBufS];
       i = pCode->NoBitHofCode;
       concatenate_bits(NoBit,BufComp,i,pCode->HofCode);
       NoBit += i;
      }
  if (NoBit >= NoBitSource)
       rc = 2;
  free(pCodeRoot);
  }

  if (rc==Ø)
  {
      m = NoBit/8;
      if (NoBit > (m * 8)) m++;
      *LenBufSource = m;
      memcpy(BufSource,BufComp,m);
```

```
   }
  }
  else rc = 2;
#ifdef DEBUGH
       printf("\n ComHFc >>> %i %li rc=%i",i,NoBit,rc);
#endif
   return(rc);
  }

  /*---------- Procedure adds bit string2 to bit string1 -----------*/
  void concatenate_bits (long    NoBit1, UNS char *str1,
                        UNS int NoBit2, UNS char *str2)
  {
   UNS char *ps1, *ps2, field;
   UNS long i, j, RestBit;

   i=NoBit1/8;
   ps1 = (UNS char *) &str1[i];
   ps2 = (UNS char *) str2;

   RestBit = NoBit1 - i*8;
   i = 8 * sizeof(UNS char);
   j=(NoBit2+RestBit)/i+1;
   if (NoBit2+RestBit > j*i) j++;
   for(i=1; i <= j; i++,ps1++,ps2++)
   {
     field = *ps2;
     field >>= RestBit;
     *ps1 != field;
     field = *ps2;
     field <<= (8-RestBit);
     *(ps1+1) = field;
   }
#ifdef DEBUGH
   /*-------------------- Print compressed file --------------------*/
         BitToChar(NoBit2,str2,BufWork);
         printf("\n HofCode %i %s",NoBit2,BufWork);
         i=(NoBit1+NoBit2)/8-1;
         j=NoBit1+NoBit2 - i*8;
         BitToChar(j,&str1[i],BufWork);
         printf("\n BufComp %i %s",NoBit1,BufWork);
#endif
  }

#ifdef DEBUGH
  /*****************************************************************
  * Subroutine for conversion bit string to character string of Ø and 1
  * - Parameters:
  *    - Number of bits
  *    - Bit string address
```

```
 *    - Character string address
 ******************************************************************/
 void BitToChar (int NoBits, UNS char *BitS, UNS char *CharS)
 {
  unsigned char BitMaps[8] = {Øx8Ø,Øx4Ø,Øx2Ø,Øx1Ø,ØxØ8,ØxØ4,ØxØ2,ØxØ1};
  int i,j;
     for(i=Ø; i <= NoBits; i += 8, BitS++)
        for(j=Ø; j <= 7 && i + j < NoBits; j++, CharS++)
           if (*BitS & BitMaps[j])
              *CharS = '1';
           else *CharS = 'Ø';
     *CharS = '\Ø';
 }
#endif
```

COMPRESSION PROGRAM

```
 /******************************************************************
     A program for the compression of sequential datasets. COMPRESC
     reads records from the input file and writes them in a 64KB area.
     Then it calls the compression subroutine and provides the area
     length and area address.
     Input parameters:
     - Type of data in input dataset:
        T - Text data
        B - Binary data
     - input dataset in the following format:
        DD:IN - DD name of the host input dataset
        input.txt - name of the PC input dataset
     - output dataset in the following format:
        DD:OUT - DD name of the host output dataset
        output.txt - name of the PC output dataset
     Output dataset consists of blocks of compressed data with the
     following format: length of block (2 by), compressed block.
  ******************************************************************/
 #include <stdio.h>
 #include <string.h>
 #include <stdlib.h>
 #include <signal.h>
 #define  UNS unsigned
 #define  SH   short
 #define  BUF_SIZE 6553Ø

 /******************************************************************/
 main(int brparm, char *parm[])
 {
   FILE        *fIn, *fOut;
   char        *In, *Out, *BufIn, TypeData, *Format;
   long  int   Num_By, Num_Comp_By = 1l, Num_Source_By = Øl;
```

```
UNS SH int  i, rc;
UNS SH int  BufLen;
UNS SH int  Num_Block=Ø;
UNS char    neof = ØxØ1;

struct
{
   char TypeCode;        /* Code of Blank Øx4Ø-EBCIDIC, Øx2Ø-ASCII */
   char TypeData;        /* 1 - Text,  Ø - Binary */
} control;

UNS SH CoLZHFc(char, UNS SH int *, char *);

signal(SIGTERM,SIG_DFL);
signal(SIGABRT,SIG_DFL);

TypeData = toupper(*parm[1]);
In  = parm[2];
Out = parm[3];
if (TypeData != 'T' && TypeData != 'B')
   { printf("\n COMP Type of data must be (T - Text or B - Binary)");
     exit(12);
   }
Format = (TypeData == 'T') ? "r":"rb";
if ((fIn = fopen(In,Format)) == NULL)
   { printf("\n COMP Open error In dataset %s",In);
     exit(13);
   }
if ((fOut = fopen(Out,"wb")) == NULL)
   { printf("\n COMP Open error Out dataset %s",Out);
     exit(14);
   }

BufIn = (char*) calloc(BUF_SIZE+5,sizeof(char));
if (BufIn == NULL) return(2);

BufLen = fread(BufIn,sizeof(char),BUF_SIZE,fIn);
if (i = ferror(fIn))
{
   printf("\n COMP reading error");
   exit(14);
}
control.TypeCode = ' ';
control.TypeData = (toupper(TypeData) == 'T') ? 1:Ø;
fwrite(&control,sizeof(control),1,fOut);

while(BufLen > Ø)
{
 printf("\n COMP block %2li. length %i by",++Num_Block,BufLen);
```

```
    Num_Source_By += BufLen;

    Num_By = BufLen;
    rc = CoLZHFc(TypeData,&BufLen,BufIn);
    printf(" - gives %i by  ratio is %2.2f %% RC=%i",
            BufLen,100.00f - (BufLen * 100.00f) / Num_By,rc);

    Num_Comp_By += BufLen + 2;
    fwrite((char *)&BufLen,sizeof(char),2,fOut);
    fwrite(BufIn,sizeof(char),BufLen,fOut);
    BufLen = fread(BufIn,sizeof(char),BUF_SIZE,fIn);
   }   /* while */

   printf("\n COMP ### Total %i by Compressed, result is %i by"
     " ratio is %7.2f %%",Num_Source_By, Num_Comp_By,
      100.00 - (Num_Comp_By * 100.00f /Num_Source_By));
   return(0);
 }
```

DECOMPRESSION SUBROUTINE

```
/*********************************************************************
 * DeLZHFc - Decompress data compressed with LZ77 + Huffman algorithm
 *
 * - Input is the area for decompressing
 * - There are two input parameters:
 *   1  is area length
 *   2  is the address of area for decompressing
 * - Compressed data is returned in the same area as the data for
 *   compressing
 *
 * - Subroutine returns following return codes:
 *   rc = 0  decompression successfully finished,
 *           input BufSource area is overwritten by decompressed data
 *   rc = 4  data is not compressed, input BufSource area is not
 *           overwritten by the decompressed data
 *   rc = 12 control data are damaged, input BufSource is not
 *           overwritten by decompressed data
 *   rc = 16 decompression is unsuccessful, probably input data is
 *           damaged
 *********************************************************************/
 #include <stdio.h>
 #include <string.h>
 #include <stdlib.h>
 #define  UNS unsigned
 #define  SH   short

 /*--- Declaration of procedures ------------------------------------*/
 void RotateI(UNS SH int *,UNS long int);
 void TranslD(UNS char *, UNS int, char);
 void ModTrT (char, char, char);
 UNS SH int DecLZc (UNS char, UNS SH int *, UNS char *);
 UNS SH int DecHFc (UNS char, UNS SH int *, UNS char *);
 /*--- Declaration of global structures and variables ---------------*/
 UNS char  *BufDecompr;
 struct code
 {
    struct code *NextCode[2];
    UNS    char  Ch;
 };
#ifdef DEBUGH
 void prcode(struct code *, char *, int);
 char BufWork[256];
#endif

/*------------------------------------------------------------------*/
 UNS SH int DeLZHFc (UNS SH int *LenBufSource, UNS char *BufSource)
 {
  UNS SH int   i, lc, rc1, rc2, LenBufS;
  UNS char     LRC;

  struct
  {
     UNS SH int LenBufDecompr;
     UNS SH int LenBufCompress;
     char TypeCode;         /* Code of Blank Øx4Ø-EBCIDIC, Øx2Ø-ASCII */
     char TypeData;         /* 1 - Text,  Ø - Binary */
     char TypeCompress;     /* ØxØf - LZ77, ØxFØ - HUFF, ØxFF - Both  */
     char CR;               /* \r - Carriage Return */
     char LF;               /* \n - Line Feed  */
     char Algoritam[4];     /* Identification of compress algorithm   */
     char LRC;              /* LRC */
  } control;

 /*------- Check control information -------------------------------*/
  LenBufS = *LenBufSource;
  lc = sizeof(control);
  if (LenBufS <= lc) return(4);

  memcpy((char *)&control,&BufSource[LenBufS-lc],lc);
  if (control.TypeCode != ' ')
     RotateI(&control.LenBufDecompr,2);
  if (control.LenBufCompress != (LenBufS - sizeof(control))) return(4);

  for(LRC = ØxØØ, i=Ø; i < LenBufS; i++)
      LRC ¬= BufSource[i];
  if (LRC != ØxØØ) return(4);
  if (strncmp(control.Algoritam,"\xØØ\xØe\xØd\xff",4) != Ø)
     return(4);
  BufDecompr=(UNS char *)calloc(control.LenBufDecompr,sizeof(char));
  if (BufDecompr == NULL) return(20);
  LenBufS -= lc;

  if ((control.TypeCompress & 0x0f) == 0x0f)
     rc1 = DecHFc (control.TypeCode, &LenBufS, BufSource);

  if ((control.TypeCompress & 0xf0) == 0xf0)
     rc2 = DecLZc (control.TypeCode, &LenBufS, BufSource);

  if (control.LenBufDecompr != LenBufS) return(12);
  if (rc1 == 0 !! rc2 == 0)
     *LenBufSource = LenBufS;

  if (control.TypeCode != ' ' && control.TypeData == 0x01)
  {
     ModTrT(control.CR,control.LF,control.TypeCode);
     TranslD(BufSource,*LenBufSource,control.TypeCode);
  }
  free(BufDecompr);

  return((rc1<<8)|rc2);
}
 /*******************************************************************
 * DecLZc - Decompress data compressed with the LZ77 algorithm
 *******************************************************************/
 UNS SH int DecLZc (UNS char TypeCode,
                    UNS SH int *LenBufSource,
                    UNS char *BufSource)
 {
  UNS char    *pBufDecompr, *ppBufDecompr;
  UNS char    *pBufSource, *pBufEndSource, *pBufPosLen;
  UNS SH int  i, j, l, rc;

  struct
  {
    UNS SH int LenBufDecompr;
    UNS SH int NoPosLen;
  } control;

  typedef struct
  {
     UNS SH int Curr_Pos;
     UNS SH int Pos;
     UNS SH int Len;
  } pos_len;
  pos_len *pl;

 /*------- Check control information -------------------------------*/
```

```
  pBufSource = BufSource;
  memcpy((char *) &control,pBufSource,sizeof(control));
  pBufSource += sizeof(control);

  if (TypeCode != ' ')
     RotateI(&control.LenBufDecompr,2);

 /*------ Decompress the area --------------------------------------*/
  pBufPosLen = pBufSource;
  pBufSource += control.NoPosLen * sizeof(pos_len);
  if (TypeCode != ' ')
     RotateI((UNS SH int*)pBufPosLen,control.NoPosLen*3);
#ifdef DEBUGL
 printf("\n DecLZc LenBufDecompr %i NoPosLen %i",
         control.LenBufDecompr,control.NoPosLen);
#endif
 /*--- Decompression ------------------------------------------------*/
  pBufEndSource = BufSource + *LenBufSource;
  memset(BufDecompr,Ø,control.LenBufDecompr);
  pBufDecompr = BufDecompr;
  for(i=Ø,l=Ø; i<control.NoPosLen && pBufSource <= pBufEndSource; i++)
  {
     pl = (pos_len *) pBufPosLen;
     pBufPosLen += sizeof(pos_len);
#ifdef DEBUGL
 printf("\n DecLZc cp %i p %i l %i",pl->Curr_Pos,pl->Pos,pl->Len);
#endif
     j=pl->Curr_Pos - 1;
     if (j <= control.LenBufDecompr)
     {
        memcpy(pBufDecompr,pBufSource,j);
        pBufDecompr += j;
        pBufSource += j;
        control.LenBufDecompr -= j;
        if (pl->Len <= control.LenBufDecompr)
        {
           ppBufDecompr = &BufDecompr[pl->Pos];
           /* if buffers not overlap we can use memcpy */
           if (pBufDecompr - ppBufDecompr > pl->Len)
           {
              memcpy(pBufDecompr,ppBufDecompr,pl->Len);
              pBufDecompr += pl->Len;
           }
           else
           {
              j = pl->Len;
              do {
                *pBufDecompr++ = *ppBufDecompr++;
              } while (--j);
           }
         l = pl->Curr_Pos + pl->Len;
         control.LenBufDecompr -= pl->Len;
      }
      else break;
   }
   else break;
  }
/*--------------- Add different string to the end ------------------*/
  if (i != control.NoPosLen)
     rc = 12;
  else
  {
     j=pBufEndSource - pBufSource;
     if (control.LenBufDecompr != j)
        rc = 12;
     else
     {
        memcpy(pBufDecompr,pBufSource,j);
        pBufDecompr += j;
        control.LenBufDecompr -= j;
        if (control.LenBufDecompr != Ø)
           rc = 16;
        else
        {
           rc = Ø;
           l = (UNS SH int) (pBufDecompr - BufDecompr);
           *LenBufSource = l;
           memcpy(BufSource,BufDecompr,l);
        }
     }
  }

 return(rc);
}

/*********************************************************************
* DecHFc - Decompress data compressed with the Huffman algorithm
*********************************************************************/
UNS SH int DecHFc (UNS char TypeCode,
                   UNS SH int *LenBufSource,
                   UNS char *BufSource)
{
 UNS char    *pBufDecompr, *pBufEndDecompr;
 UNS char    *pBufSource, *pBufEndSource;

 struct
 {
   UNS SH int LenBufDecompr;
   UNS SH int NoHuffCodes;
 } control;
```

```
   struct code *ppCode, *pCode, *pCodeRoot = NULL;
   struct code **pHuffCode;

   UNS SH int  k, l, Bit;
   UNS char    BitMaps[8] = { 0x80,0x40,0x20,0x10,0x08,0x04,0x02,0x01 };
   int  j,rc;

  /*------- Check control information -------------------------------*/
   pBufSource = BufSource;
   memcpy((char *) &control,pBufSource,sizeof(control));
   pBufSource += sizeof(control);
   if (TypeCode != ' ')
      RotateI(&control.LenBufDecompr,2);
#ifdef DEBUGH
  printf("\n DecHFc LenBufdecompr %i NoHuffCodes %i",
           control.LenBufDecompr,control.NoHuffCodes);
#endif

   pBufEndSource = BufSource + *LenBufSource;

   if (control.LenBufDecompr > 0 &&
       control.NoHuffCodes >= 0 && control.NoHuffCodes <= 255)
   {
     memset(BufDecompr,0,control.LenBufDecompr);
     pHuffCode = (struct code **) calloc(256,sizeof(struct code *));
     if (pHuffCode == NULL) return(20);
     k = (control.NoHuffCodes + 1) * 2;
     pCodeRoot = (struct code *) calloc(k,sizeof(struct code));
     if (pCodeRoot == NULL) return(20);

     pBufDecompr = BufDecompr;
     pBufEndDecompr = BufDecompr + control.LenBufDecompr;
  /*----------------- Generate Huffan's code -------------------------*/
     pCode = pCodeRoot+1;
     pHuffCode[(UNS SH int) *(pBufSource+1)] = pCodeRoot;
     for(k=1; k <= control.NoHuffCodes; k++)
     {
        ppCode = pHuffCode[(UNS SH int) *(pBufSource+1)];
        if (ppCode == NULL) return(32);
        for(j=0; j <= 1; j++, pCode++, pBufSource++)
        {
           pHuffCode[(UNS SH int) *pBufSource] = pCode;
           ppCode->NextCode[1-j] = pCode;
           pCode->Ch = *pBufSource;
        }
     }

#ifdef DEBUGH
  printf("\n DecHFc Print Huff code");
```

```
 prcode(pCodeRoot, BufWork, Ø);
#endif
 /*------------------------ Decompression -------------------------*/
    k=Ø;
    do
    {
      Bit = (*pBufSource & BitMaps[k]) != ØxØØ ? 1: Ø;
      for(pCode=pCodeRoot;
          pCode->NextCode[Bit] != NULL && pBufSource < pBufEndSource;
          pCode = pCode->NextCode[Bit],k++)
      {
         if (k > 7)
         {
            k = Ø;
            pBufSource++;
         }
         Bit = (*pBufSource & BitMaps[k]) != ØxØØ ? 1: Ø;
      }
      *pBufDecompr++=pCode->Ch;
    } while( pBufDecompr < pBufEndDecompr &&
             pBufSource < pBufEndSource);
    free(pCodeRoot);
    free(pHuffCode);

    if (pBufDecompr != pBufEndDecompr)
       rc = 16;
    else
    {
       rc = Ø;
       l = (UNS SH int) (pBufDecompr - BufDecompr);
       *LenBufSource = l;
       memcpy(BufSource,BufDecompr,l);
    }
 }
 else rc=12;

 return(rc);
 }

/*---------- Procedure for rotate byte in integer fields -----------*/
void RotateI(UNS SH int *IntArray, UNS long int NoInt)
{
  UNS SH int i;
  UNS char *ch, c;
  for(i=Ø, ch = (char *) IntArray; i<NoInt; i++, ch += 2)
  {
     c = *ch;
     *ch = *(ch+1);
     *(ch+1) = c;
  }
```

```
}

#ifdef DEBUGH
 /*---------------- Procedure for code printing ---------------------*/
 void prcode(struct code *p, char codeb[],int k)
 {
   if (p->NextCode[0] == NULL && p->NextCode[1] == NULL)
   {
       codeb[k] = '\0';
       printf("\n %c %x %i -> %s ", p->Ch, p->Ch, k, codeb);
   }
   else {
         codeb[k] = '0';
         prcode(p->NextCode[0],codeb,k+1);
         codeb[k] = '1';
         prcode(p->NextCode[1],codeb,k+1);
        }
 };
#endif
```

DECOMPRESSION PROGRAM

```
 /*********************************************************************
  Program for decompression of sequential datasets. DECOMPRC reads
  records from the input file till the end of the compressed block.
  is reached. Then it calls the decompression subroutine and provides
  the area length and area address.
  Input parms:
  - input dataset in the following format:
     DD:IN - DDname of the host input dataset
     input.txt - name of the PC input dataset
  - output dataset in the following format:
     DD:OUT - DDname of the host output dataset
     output.txt - name of the PC output dataset
  *********************************************************************/
 #include <stdio.h>
 #include <string.h>
 #include <stdlib.h>
 #include <signal.h>
 #define  UNS unsigned
 #define  SH  short
 #define  BUF_SIZE 65535

 extern void RotateI(UNS SH int *,UNS SH int);
 /********************************************************************/
 main(int brparm,char *parm[])
 {
   FILE        *fIn, *fOut;
   char        *In, *Out, *BufIn, *Format;
   UNS int        rc = 0;
   UNS SH int  Num_Block=0, BufLen;
   long int    NumBy = 0, Len=0, l=0;

   struct
   {
      char TypeCode;       /* Code of Blank 0x40-EBCIDIC, 0x20-ASCII */
      char TypeData;       /* 1 - Text,  0 - Binary */
   } control;

   UNS SH DeLZHFc(UNS SH int *, char *);

#ifdef DEBUG
/*--- Check if Translation tables are OK ----------------------------*/
void Check_TrT(void);
     Check_TrT();
#endif

   signal(SIGTERM,SIG_DFL);
   signal(SIGABRT,SIG_DFL);

   In = parm[1];
   Out = parm[2];
   if ((fIn = fopen(In,"rb")) == NULL)
      { printf("\n COMP Open error In dataset %s",In);
        exit(12);
      }
   fread(&control,sizeof(control),1,fIn);

   Format = (control.TypeData == 0x01) ? "w":"wb";
   if ((fOut = fopen(Out,Format)) == NULL)
      { printf("\n COMP Open error Out dataset %s",Out);
        exit(13);
      }

   BufIn = (char*) calloc(BUF_SIZE+5,sizeof(char));
   if (BufIn == NULL) return(2);

   fread(&BufLen,sizeof(char),2,fIn);
   if (control.TypeCode != ' ')
       RotateI(&BufLen,1);
   while (!feof(fIn) && BufLen > 0 && rc <=4)
   {
    fread(BufIn,sizeof(char),BufLen,fIn);

    printf("\n DEC  block %2li. length %i by",++Num_Block,BufLen);
    rc = DeLZHFc(&BufLen, BufIn);
    printf(" - decompressed to %i by RC=%i", BufLen,rc);
    NumBy += BufLen;
/*-- Print output buffer -------------------------------------------*/
    Out = BufIn;
    Len = BufLen;
    do
    {
       l = fwrite(Out,sizeof(char),Len,fOut);
       if (Len > l)
          fwrite("\n",sizeof(char),1,fOut);
       if (l==Ø) l = 1;
       Len -=l;
       Out += l;
    } while(Len > Ø);

    fread(&BufLen,sizeof(char),2,fIn);
    if (control.TypeCode ! = ' ')
        RotateI(&BufLen,1);
 } /* while */
 printf("\n ### TOTAL DECOMPRESSED TO %i by",NumBy);
 if (rc > 4)
   printf("\n >>> DECOMPESSION ERROR <<<");
   return(Ø);
 }
```

DATA TRANSLATION SUBROUTINE

```
#include <stdio.h>
#define  UNS unsigned
#define  SH  short
/**********************************************************************
  The translate tables in this subroutine specify translation from
  EBCDIC to ASCII and vice versa according to our national standard.
  You have to check if they are compatible with your standards
  and customize them if they are not. When you make changes to the
  translation tables you have to keep them inverse. To check them use
  the procedure Check_TrT.
 **********************************************************************/

UNS char TrT_Ebcdic_Ascii[256] = {
/*----  TRANSLATION TABLE EBCDIC ==> ASCII ---------------------------*/
/*----  Ø   1   2   3   4   5   6   7   8   9   A   B   C   D   E   F */
/*Ø*/"\xØØ\xØ1\xØ2\xØ3\xDF\xØ9\xBA\xAA\xAE\xC3\xC4\xØB\xØC\xØD\xØE\xØF"
/*1*/"\x1Ø\x11\x12\x13\xAF\xB9\xØ8\xBB\x18\x19\xCA\xC5\x1C\x1D\x1E\x1F"
/*2*/"\xBØ\xB1\xB2\xB3\xB4\xØA\x17\x1B\xBC\xBF\xCØ\xC1\xC2\xØ5\xØ6\xØ7"
/*3*/"\xC8\xC9\x16\xCB\xCC\xCD\xCE\xØ4\xD9\xDA\xDB\xDC\x14\x15\xFE\x1A"
/*4*/"\x2Ø\xFF\x83\x84\xEE\xAØ\xC7\x9F\x87\x86\x5B\x2E\x3C\x28\x2B\x21"
/*5*/"\x26\x82\xA9\x89\x85\xA1\x8C\x96\x92\xE1\x5D\x24\x2A\x29\x3B\x5E"
/*6*/"\x2D\x2F\xB6\x8E\xF1\xB5\xC6\xAC\x8Ø\x8F\x7C\x2C\x25\x5F\x3E\x3F"
/*7*/"\xF3\x9Ø\xA8\xD3\xDE\xD6\xD7\x95\x91\x6Ø\x3A\x23\x4Ø\x27\x3D\x22"
/*8*/"\xF4\x61\x62\x63\x64\x65\x66\x67\x68\x69\x98\xE5\xDØ\xEC\xFD\xAD"
/*9*/"\xF8\x6A\x6B\x6C\x6D\x6E\x6F\x70\x71\x72\x88\xE4\xE7\xF7\xF2\xCF"
/*A*/"\xA5\x7E\x73\x74\x75\x76\x77\x78\x79\x7A\x97\xD5\xD1\xED\xFC\xB8"
/*B*/"\xFA\xA4\xBE\xDD\xBD\xF5\xA7\xAB\xA6\x8D\x9D\xE3\xE6\xF9\xEF\x9E"
/*C*/"\x7B\x41\x42\x43\x44\x45\x46\x47\x48\x49\xF0\x93\x94\xEA\xA2\x8B"
/*D*/"\x7D\x4A\x4B\x4C\x4D\x4E\x4F\x50\x51\x52\xB7\xFB\x81\x9C\xA3\xD8"
/*E*/"\x5C\xF6\x53\x54\x55\x56\x57\x58\x59\x5A\xD4\xE2\x99\xE8\xE0\x8A"
/*F*/"\x30\x31\x32\x33\x34\x35\x36\x37\x38\x39\xD2\xEB\x9A\x9B\xE9\x7F"
};

UNS char TrT_Ascii_Ebcdic[256] = {
/*----  TRANSLATION TABLE ASCII ==> EBCDIC --------------------------*/
/*----  0   1   2   3   4   5   6   7   8   9   A   B   C   D   E   F */
/*0*/"\x00\x01\x02\x03\x37\x2D\x2E\x2F\x16\x05\x25\x0B\x0C\x0D\x0E\x0F"
/*1*/"\x10\x11\x12\x13\x3C\x3D\x32\x26\x18\x19\x3F\x27\x1C\x1D\x1E\x1F"
/*2*/"\x40\x4F\x7F\x7B\x5B\x6C\x50\x7D\x4D\x5D\x5C\x4E\x6B\x60\x4B\x61"
/*3*/"\xF0\xF1\xF2\xF3\xF4\xF5\xF6\xF7\xF8\xF9\x7A\x5E\x4C\x7E\x6E\x6F"
/*4*/"\x7C\xC1\xC2\xC3\xC4\xC5\xC6\xC7\xC8\xC9\xD1\xD2\xD3\xD4\xD5\xD6"
/*5*/"\xD7\xD8\xD9\xE2\xE3\xE4\xE5\xE6\xE7\xE8\xE9\x4A\xE0\x5A\x5F\x6D"
/*6*/"\x79\x81\x82\x83\x84\x85\x86\x87\x88\x89\x91\x92\x93\x94\x95\x96"
/*7*/"\x97\x98\x99\xA2\xA3\xA4\xA5\xA6\xA7\xA8\xA9\xC0\x6A\xD0\xA1\xFF"
/*8*/"\x68\xDC\x51\x42\x43\x54\x49\x48\x9A\x53\xEF\xCF\x56\xB9\x63\x69"
/*9*/"\x71\x78\x58\xCB\xCC\x77\x57\xAA\x8A\xEC\xFC\xFD\xDD\xBA\xBF\x47"
/*A*/"\x45\x55\xCE\xDE\xB1\xA0\xB8\xB6\x72\x52\x07\xB7\x67\x8F\x08\x14"
/*B*/"\x20\x21\x22\x23\x24\x65\x62\xDA\xAF\x15\x06\x17\x28\xB4\xB2\x29"
/*C*/"\x2A\x2B\x2C\x09\x0A\x1B\x66\x46\x30\x31\x1A\x33\x34\x35\x36\x9F"
/*D*/"\x8C\xAC\xFA\x73\xEA\xAB\x75\x76\xDF\x38\x39\x3A\x3B\xB3\x74\x04"
/*E*/"\xEE\x59\xEB\xBB\x9B\x8B\xBC\x9C\xED\xFE\xCD\xFB\x8D\xAD\x44\xBE"
/*F*/"\xCA\x64\x9E\x70\x80\xB5\xE1\x9D\x90\xBD\xB0\xDB\xAE\x8E\x3E\x41"
};

void Translate(UNS char *TargetStr, UNS char *SourceStr,
               UNS char *TrT, UNS int NoCh)
{
  UNS int i;
  for(i=0; i<NoCh; i++,SourceStr++)
      *TargetStr++ = TrT[*SourceStr];
}

void TranslD(UNS char *Str, UNS int LenStr, char TypeCode)
{
  void Translate(UNS char *,UNS char *,UNS char *, UNS int);

  if (TypeCode == 0x20)  /* Source is ASCII code */
     Translate(Str,Str,TrT_Ascii_Ebcdic,LenStr);
  else
  if (TypeCode == 0x40)  /* Source is EBCDIC code */
     Translate(Str,Str,TrT_Ebcdic_Ascii,LenStr);
}

void ModTrT(char CR, char LF, char TypeCode)
```

```
{
  struct
  {
     char CR;              /* \r - Carriage Return */
     char LF;              /* \n - Line Feed       */
  } local = {'\r','\n'};

  if (TypeCode == 0x20)    /* Source is ASCII code */
  {
     TrT_Ascii_Ebcdic[CR] = local.CR;
     TrT_Ascii_Ebcdic[LF] = local.LF;
  }
  else
  if (TypeCode == 0x40)    /* Source is EBCDIC code */
  {
     TrT_Ebcdic_Ascii[CR] = local.CR;
     TrT_Ebcdic_Ascii[LF] = local.LF;
  }
}

#ifdef DEBUG
/*--- Check if Translation tables are OK ----------------------------*/
void Check_TrT(void)
{
   int i, j;
   UNS char c1,c2, c3;
   for(i=j=0; i<256; i++)
   {
      c1=i;
         Translate(&c2,&c1,TrT_Ebcdic_Ascii,1);
         Translate(&c3,&c2,TrT_Ascii_Ebcdic,1);
      if (c1 != c3)
      {
         j++;
         printf("%i TRT c=%c %x %x %x \n",i,c1,c1,c2,c3);
      }
   }
   if (j==0)
        printf("\nTranslation table is OK!");
   else
        printf("\nTranslation table is not inverse!");
}
#endif
```

JCL FOR COMPILING ON OS/390

```
//useridL     JOB MSGCLASS=X,MSGLEVEL=(1,0),NOTIFY=&SYSUID,CLASS=A
//*
//COMPRESC PROC M=,D=
//COMPRESC EXEC EDCC1,CPARM='OBJ,SOURCE,OFFSET,LIST',
//         CPARM2=&D,INFILE=userid.USER.C(&M)
```

```
//COMPILE.SYSLIN DD DSN=userid.USER.OBJ(&M),DISP=SHR
//    PEND
//*
//COMPRESL PROC M=,D=
//COMPRESL EXEC EDCCL1,CPARM='OBJ,SOURCE,OFFSET,LIST',
//         CPARM2=&D,INFILE=userid.USER.C(&M)
//LKED.SYSLIB   DD DSN=CEE.SCEELKED,DISP=SHR DISP=SHR
//              DD DSN=userid.USER.OBJ,DISP=SHR
//LKED.SYSLMOD  DD DSN=userid.USER.LOAD(&M),DISP=SHR
//    PEND
//*
//COMHFC   EXEC COMPRESC,M=COLZHFC,D='DEFINE(NODEBUGL,NODEBUGH)'
//COMPRESC EXEC COMPRESL,M=COMPRESC,D='DEFINE(NODEBUG)'
//*
//DECLZC   EXEC COMPRESC,M=TRANSLD,D='DEFINE(NODEBUG)'
//DECHFC   EXEC COMPRESC,M=DELZHFC,D='DEFINE(NODEBUGL,NODEBUGH)'
//DECOMPRC EXEC COMPRESL,M=DECOMPRC,D='DEFINE(NODEBUG)'
//
```

JCL FOR OS/390 TESTING

```
//useridC JOB MSGCLASS=X,NOTIFY=&SYSUID,MSGLEVEL=(1,1),CLASS=C
//*********************************************************
//*        DELETING WORK DATASET
//*********************************************************
//DEL      EXEC  PGM=IDCAMS
//SYSPRINT DD DUMMY
//SYSIN    DD *
   DELETE         userid.C.LIST
   DELETE         userid.D.LIST
   SET MAXCC=Ø
/*
//*********************************************************
//*        COMPRESSING TEST DATA
//*********************************************************
//COMPRESS EXEC  PGM=COMPRESC,PARM='"T" "DD:IN" "DD:OUT"'
//STEPLIB  DD DSN=userid.USER.LOAD,DISP=SHR
//SYSPRINT DD SYSOUT=X
//IN       DD DSN=userid.TEST,DISP=SHR
//OUT      DD DSN=userid.C.LIST,DISP=(NEW,CATLG,DELETE),
//         UNIT=SYSDA,DCB=(RECFM=U,BLKSIZE=Ø),
//         SPACE=(CYL,(5ØØ,5ØØ),RLSE)
//*
//*********************************************************
//*        DECOMPRESSING COMPRESSED DATA
//*********************************************************
//DECOMPR  EXEC  PGM=DECOMPRC,PARM='"DD:IN" "DD:OUT"'
//STEPLIB  DD DSN=userid.USER.LOAD,DISP=SHR
//SYSPRINT DD SYSOUT=X
```

```
//IN       DD DSN=userid.C.LIST,DISP=SHR
//OUT      DD DSN=userid.D.LIST,DISP=(NEW,CATLG,DELETE),
//         UNIT=SYSDA,DCB=userid.TEST,
//         SPACE=(CYL,(500,500),RLSE)
/*
//*************************************************************
//*        COMPARISON OF TEST AND DECOMPRESSED DATA
//*************************************************************
//COMPARE  EXEC  PGM=IEBCOMPR
//SYSPRINT DD SYSOUT=X
//SYSUT1   DD DSN=userid.TEST,DISP=SHR
//SYSUT2   DD DSN=userid.D.LIST,DISP=SHR
//SYSIN    DD *
   COMPARE TYPORG=PS
/*
//
```

BAT PROCEDURE FOR PC TESTING

```
del test.co#
del test.de#

compresc T test.txt test.co#
decomprc   test.co# test.de#
comp       test.txt test.de#
```

Emina Spasic and Dragan Nikolic
Systems Programmers
Postal Savings Bank (Yugoslavia)

Batch FTP between an MVS client and NT server

INTRODUCTION

For such a widely used facility, FTP seems to have been neglected when it comes to documentation, particulary within MVS. During my work on a recent project I had the feeling that I had to somewhat reinvent the wheel. Much of my time was spent testing several possible solutions, because I could not find a clear example within the documentation. Most of the examples were also mainly to do with the on-line use of FTP, hints and tips for batch implementation were hard to find.

So where is the first place to look when you need information on FTP (MVS)? No not in the FTP users guide, because there isn't one (well I couldn't find one). I found most of my material in the *OS/390 TCP/IP OE: User's Guide*, Chapter 2, *Transferring Data using File Transfer Protocol (FTP) in an OE Environment*.

The following article documents my recent experiences in the use of FTP in a batch environment, which I hope will be of use to others in a similar situation. The project on which I worked needed a quick solution, but one that could be automated, hence FTP for the transfer, REXX for the logic, and JES2 for the automation.

PROJECT BACKGROUND

With the ever-falling price of hardware it is not suprising to find that often the answer to a problem is 'we can use a server for this and a server for that'. All too often little or no thought is given to the origin of the data. Recently we were faced with such a problem, where a server solution was developed and as an after thought, it was considered that a connection to the mainframe data would be 'nice', and we would like it 'yesterday'. That is were I came in, with a backgroud in quick solutions, experience of REXX and JCL, and I could spell FTP.

The project was relatively straight forward. A file of changes on the server needed to be transferred one-to-one to the mainframe and a change confirmation file on the mainframe needed to be transfered one-to-many to the server.

Later the project was expanded to include the transfer to the server of complete and incremental update information. Both of these were previously distributed on disk.

With the dramatic increase in data volumes, various modifications to the data handling had to be made. The techniques I employed and the modifications made are documented below.

I will start with the simplest solution first because the common parts are easier to see and understand before the solutions become somewhat more complex.

SOLUTION 1

The file on the server, ayymmtt.ant (where yymmdd is the date), needs to be transfered to the next generation of the generation dataset TESTENV.FTPANTRG.A00001.BESTR on the mainframe. The controlling JCL supplies the DD statement, the date, the username, and password for the FTP connection, and calls the REXX procedure AISGET.

The REXX procedure AISGET in turn interogates the DD statement for the name of the output file, it then converts the date into a filename which is expected to be received from the server platform. The FTP commands are queued on a stack and FTP is then called, using the stack as input. FTP returns a error code which is then interpreted by the REXX procedure to ascertain the severity of the error.

The FTP return code is a five-digit number; the first two digits represent the action undertaken (FTP subcommand), and the last three represent the FTP reply code. To allow quick decyphering of the codes by our data processing team, the codes and their meanings are written to the output list each time the REXX routine is called.

AISGET JCL

```
//AISGET JOB ,'USERID',
//           MSGLEVEL=(1,1),MSGCLASS=X,
//           CLASS=A,REGION=4M,
//           NOTIFY=&SYSUID,
//           USER=,GROUP=,PASSWORD=
//*
```

```
//* LIB: USERID.JCL.CNTL(AISGET)
//* DOC: TRANSFER OF FILE AJJMMDD.ANT FROM AIS-SERVER
//*
//*****************************************************************
//* CALL:
//* AISGET <FILEDATE JJMMTT> <USERID> <PASSWORD>
//*****************************************************************
//STEP1    EXEC TSOBATCH
//* DSN CONTAINING REXX PROCEDURES
//SYSPROC  DD DSN=SYNPU1.SYST.CLIST,DISP=SHR
//         DD DSN=SYNPU1.V2R6.CLIST,DISP=SHR
//OUTPUT   DD SYSOUT=*
//* GENERATION DATASET TO RECEIVE TRANSFER FILE
//ANTRAG   DD DSN=TESTENV.FTPANTRG.AØØØØ1.BESTR(+1),
//             DISP=(NEW,CATLG,DELETE),
//             LRECL=512
//SYSTSIN  DD  *
%AISGET Ø1Ø831 USERIDØ1 PASSWORDØ1
//*
```

AISGET.EXEC

```
/* REXX ************************************************************ */
/*                          ** AISGET  **                            */
/*                                                                   */
/*  Connection to TCP/IP server "ais-ablage"  (AIS-FTP)              */
/*  change directory to root/ant/anthost                             */
/*  transfer file ayymmdd.ant from AIS-FTP server to host            */
/*  delete source file ayymmdd.ant from AIS-FTP server               */
/*                                                                   */
/* ***************************************************************** */
arg filedate aisuser aispass .      /* filedate = YYMMDD             */
                                    /* userID and password from JCL   */
trace o                             /* trace conrol (on=r / off=o     */
rc = LISTDSI(antrag "FILE")         /* get DSN info from DD statement */
dsantrag = SYSDSNAME                /* SYSDSNAME = JCL dataset name   */
dsantrag = "'"dsantrag"'"           /* in quotes to avoid prefix      */
antragnt = "a"filedate".ant"        /* NT name = ayymmdd.ant          */
getantrg = "get "antragnt           /* build FTP "get" statement      */
getantrg = getantrg dsantrag        /*                                */
getantrg = getantrg " (REPLACE"     /*                                */
                          /* "get ayymmdd.ant hostdsname (REPLACE"    */
delantrg = "delete "antragnt        /* build FTP "delete" statement   */
                                    /* "delete ayymmdd.ant"           */
/* ***************************************************************** */
/* build complete FTP commands together in the stack for the "get"   */

   "newstack"
   queue aisuser aispass            /* userid password                */
   queue "cd ant/anthost"           /* change directory to source     */
   queue "ls"             /* list the directory for job listing       */
   queue getantrg                   /* get file from server           */
   queue "quit"                     /* exit FTP                       */
   queue ""

/* call FTP using stack as input */
   "ftp ais-ablage    (exit "       /* FTP call to server             */

   ftpcode = rc                     /* FTP returns 5 digit returncode */
   ftpcode = right(ftpcode,5,'Ø')
   ftpscmd = substr(ftpcode,1,2)    /* subcommand = 1st 2 digits      */
   ftprply = substr(ftpcode,3,3)    /* reply code = last 3 digits     */
   call ftpscmds                    /* display list of subcommands    */
   call ftpcodes                    /* display list of reply codes    */
                          /* display subcommand and reply code        */
   say "****************************"
   say "FTP SUBCOMMAND = " ftpscmd
   say "FTP REPLY CODE = " ftprply
   say "****************************"

/* interpret severity of reply code and pass errorcode of Ø, 4 or 8   */
/* back to JCL                                                        */
   ftpretcd = ''
   select
          when ftpscmd = "ØØ"  then ftpretcd = Ø
          when ftprply = "11Ø" then ftpretcd = Ø
          when ftprply = "12Ø" then ftpretcd = Ø
          when ftprply = "125" then ftpretcd = Ø
          when ftprply = "15Ø" then ftpretcd = Ø
          when ftprply = "2ØØ" then ftpretcd = Ø
          when ftprply = "211" then ftpretcd = Ø
          when ftprply = "212" then ftpretcd = Ø
          when ftprply = "213" then ftpretcd = Ø
          when ftprply = "214" then ftpretcd = Ø
          when ftprply = "215" then ftpretcd = Ø
          when ftprply = "22Ø" then ftpretcd = Ø
          when ftprply = "221" then ftpretcd = Ø
          when ftprply = "226" then ftpretcd = Ø
          when ftprply = "23Ø" then ftpretcd = Ø
          when ftprply = "25Ø" then ftpretcd = Ø
          when ftprply = "257" then ftpretcd = Ø
          when ftprply = "331" then ftpretcd = Ø
          when ftprply = "332" then ftpretcd = Ø
          when ftprply = "55Ø" then ftpretcd = 4
   otherwise ftpretcd = 8
   end
   if ftpretcd > 4 then
   do
     "delstack"           /* delete unused commands from stack        */
     exit ftpretcd                  /* exit when FTP call in error    */
   end
/* ****************************************************************** */
   "delstack"             /* delete unused commands from stack        */
/* ****************************************************************** */
/* build complete FTP commands together in the stack for the "delete" */
   "newstack"
   queue aisuser aispass              /* user  pass                   */
   queue "cd ant/anthost"             /* change directory             */
   queue "ls"                         /* list                         */
   queue delantrg                     /* delete antrag on NT          */
   queue "quit"
   queue ""

/* call FTP using stack as input */
   "ftp ais-ablage    (exit "         /* FTP to "ais-ablage"  host    */

   ftpcode = rc                     /* FTP returns 5 digit returncode */
   ftpcode = right(ftpcode,5,'Ø')
   ftpscmd = substr(ftpcode,1,2)    /* subcommand = 1st 2 digits      */
   ftprply = substr(ftpcode,3,3)    /* reply code = last 3 digits     */
   call ftpscmds                    /* display list of subcommands    */
   call ftpcodes                    /* display list of reply codes    */
                          /* display subcommand and reply code        */
   say "****************************"
   say "FTP SUBCOMMAND = " ftpscmd
   say "FTP REPLY CODE = " ftprply
   say "****************************"

return ftpretcd                     /* return to JCL with returncode  */

/* ****************************************************************** */
ftpcodes:

/* table of FTP reply codes to be displayed in JOB output list        */

say "+----------------------------------------------------------------+"
say "| Code| Description                                              |"
say "+-----+----------------------------------------------------------+"
say "| ØØØ | FTP subcommand contains an incorrect parameter           |"
say "|     |                                                          |"
say "|     | NB:  A reply code of ØØØ is returned from the FTP client |"
say "|     |      when it detects an incorrect parameter. The FTP     |"
say "|     |      client in this case does not send the command to the|"
say "|     |      FTP server.                                         |"
say "+-----+----------------------------------------------------------+"
say "| 11Ø | Restart marker reply                                     |"
say "+-----+----------------------------------------------------------+"
say "| 12Ø | Service ready in nnn minutes                             |"
say "+-----+----------------------------------------------------------+"
say "| 125 | Data connection already open; transfer starting          |"
say "+——+——————————————————————————————+"
say "| 15Ø | File status okay; about to open data connection         |"
say "+-----+----------------------------------------------------------+"
say "| 2ØØ | Command okay                                             |"
say "+-----+----------------------------------------------------------+"
say "| 2Ø2 | Command not implemented; not used on this host           |"
say "+-----+----------------------------------------------------------+"
say "| 2Ø8 | Unable to delete data set because expiration date has not|"
say "|     | passed                                                   |"
say "+-----+----------------------------------------------------------+"
say "| 211 | System status, or system help reply                      |"
say "+-----+----------------------------------------------------------+"
say "| 212 | Directory status                                         |"
say "+-----+----------------------------------------------------------+"
say "| 213 | File status                                              |"
say "+-----+----------------------------------------------------------+"
say "| 214 | Help message                                             |"
say "+-----+----------------------------------------------------------+"
say "| 215 | MVS is the operating system of this server               |"
say "+-----+----------------------------------------------------------+"
say "| 22Ø | Service ready for new user                               |"
say "+-----+----------------------------------------------------------+"
say "| 221 | QUIT command received                                    |"
say "+-----+----------------------------------------------------------+"
say "| 226 | Closing data connection; requested file action successful|"
say "+-----+----------------------------------------------------------+"
say "| 23Ø | User logged on; proceed                                  |"
say "+-----+----------------------------------------------------------+"
say "| 25Ø | Requested file action OK, completed                      |"
say "+-----+----------------------------------------------------------+"
say "| 257 | PATH NAME created                                        |"
say "+-----+----------------------------------------------------------+"
say "| 331 | Send password please                                     |"
say "+-----+----------------------------------------------------------+"
say "| 332 | Supply minidisk password using account                   |"
say "+-----+----------------------------------------------------------+"
say "| 421 | Service not available                                    |"
say "+-----+----------------------------------------------------------+"
say "| 425 | Cannot open data connection                              |"
say "+-----+----------------------------------------------------------+"
say "| 426 | Connection closed; transfer ended abnormally             |"
say "+-----+----------------------------------------------------------+"
say "| 45Ø | Requested file action not taken; file busy               |"
say "+-----+----------------------------------------------------------+"
say "| 451 | Requested action abended; local error in processing      |"
say "+-----+----------------------------------------------------------+"
say "| 452 | Requested action not taken; insufficient storage space in|"
say "|     | system                                                   |"
say "+-----+----------------------------------------------------------+"
say "| 5ØØ | Syntax error; command unrecognized                       |"
say "+-----+----------------------------------------------------------+"
say "| 5Ø1 | Syntax error in parameters or arguments                   |"
say "+-----+-----------------------------------------------------------+"
say "| 5Ø2 | Command not implemented                                   |"
say "+-----+-----------------------------------------------------------+"
say "| 5Ø3 | Bad sequence of commands                                  |"
say "+-----+-----------------------------------------------------------+"
say "| 5Ø4 | Command not implemented for that parameter                |"
say "+-----+-----------------------------------------------------------+"
say "| 53Ø | Not logged on                                             |"
say "+-----+-----------------------------------------------------------+"
say "| 532 | Need account for storing files                            |"
say "+-----+-----------------------------------------------------------+"
say "| 55Ø | Requested action not taken; file not found or no access  |"
say "+-----+-----------------------------------------------------------+"
say "| 551 | Requested action abended; page type unknown               |"
say "+-----+-----------------------------------------------------------+"
say "| 552 | Requested file action ended abnormally; exceeded storage |"
say "|     | allocation                                                |"
say "+-----+-----------------------------------------------------------+"
say "| 553 | Requested action not taken; file name not allowed        |"
say "+-----+-----------------------------------------------------------+"
say "| 554 | Transfer aborted; unsupported SQL statement               |"
say "+-----+-----------------------------------------------------------+"
return

/* *************************************************************** */
ftpscmds:

/* table of FTP subcommand codes to be displayed in JOB output list   */

say "+-------------------------------------------+"
say "| Code Number | Subcommand                  |"
say "+-------------+-----------------------------+"
say "| 1           | AMBIGUOUS                   |"
say "+-------------+-----------------------------+"
say "| 2           | ?                           |"
say "+-------------+-----------------------------+"
say "| 3           | ACCOUNT                     |"
say "+-------------+-----------------------------+"
say "| 4           | APPEND                      |"
say "+-------------+-----------------------------+"
say "| 5           | ASCII                       |"
say "+-------------+-----------------------------+"
say "| 6           | BINARY                      |"
say "+-------------+-----------------------------+"
say "| 7           | CD                          |"
say "+-------------+-----------------------------+"
say "| 8           | CLOSE                       |"
say "+-------------+-----------------------------+"
say "| 9           | TSO                         |"
say "+--------------+--------------------------------+"
say "| 10           | OPEN                           |"
say "+--------------+--------------------------------+"
say "| 11           | DEBUG                          |"
say "+--------------+--------------------------------+"
say "| 12           | DELIMIT                        |"
say "+--------------+--------------------------------+"
say "| 13           | DELETE                         |"
say "+--------------+--------------------------------+"
say "| 14           | DIR                            |"
say "+--------------+--------------------------------+"
say "| 15           | EBCDIC                         |"
say "+--------------+--------------------------------+"
say "| 16           | GET                            |"
say "+--------------+--------------------------------+"
say "| 17           | HELP                           |"
say "+--------------+--------------------------------+"
say "| 18           | LOCSTAT                        |"
say "+--------------+--------------------------------+"
say "| 19           | USER                           |"
say "+--------------+--------------------------------+"
say "| 20           | LS                             |"
say "+--------------+--------------------------------+"
say "| 21           | MDELETE                        |"
say "+--------------+--------------------------------+"
say "| 22           | MGET                           |"
say "+--------------+--------------------------------+"
say "| 23           | MODE                           |"
say "+--------------+--------------------------------+"
say "| 24           | MPUT                           |"
say "+--------------+--------------------------------+"
say "| 25           | NOOP                           |"
say "+--------------+--------------------------------+"
say "| 26           | PASS                           |"
say "+--------------+--------------------------------+"
say "| 27           | PUT                            |"
say "+--------------+--------------------------------+"
say "| 28           | PWD                            |"
say "+--------------+--------------------------------+"
say "| 29           | QUIT                           |"
say "+--------------+--------------------------------+"
say "| 30           | QUOTE                          |"
say "+--------------+--------------------------------+"
say "| 31           | RENAME                         |"
say "+--------------+--------------------------------+"
say "| 32           | SENDPORT                       |"
say "+--------------+--------------------------------+"
say "| 33           | SENDSITE                       |"
say "+--------------+--------------------------------+"  "
say "| 34           | SITE                           |"
say "+-------------+-------------------------------+"
say "| 35          | STATUS                        |"
say "+-------------+-------------------------------+"
say "| 36          | STRUCT                        |"
say "+-------------+-------------------------------+"
say "| 37          | SUNIQUE                       |"
say "+-------------+-------------------------------+"
say "| 38          | SYSTEM                        |"
say "+-------------+-------------------------------+"
say "| 39          | TRACE                         |"
say "+-------------+-------------------------------+"
say "| 40          | TYPE                          |"
say "+-------------+-------------------------------+"
say "| 41          | LCD                           |"
say "+-------------+-------------------------------+"
say "| 42          | LOCSITE                       |"
say "+-------------+-------------------------------+"
say "| 43          | LPWD                          |"
say "+-------------+-------------------------------+"
say "| 44          | MKDIR                         |"
say "+-------------+-------------------------------+"
say "| 45          | LMKDIR                        |"
say "+-------------+-------------------------------+"
say "| 46          | EUCKANJI                      |"
say "+-------------+-------------------------------+"
say "| 47          | IBMKANJI                      |"
say "+-------------+-------------------------------+"
say "| 48          | JIS78KJ                       |"
say "+-------------+-------------------------------+"
say "| 49          | JIS83KJ                       |"
say "+-------------+-------------------------------+"
say "| 50          | SJISKANJI                     |"
say "+-------------+-------------------------------+"
say "| 51          | CDUP                          |"
say "+-------------+-------------------------------+"
say "| 52          | RMDIR                         |"
say "+-------------+-------------------------------+"
say "| 53          | HANGEUL                       |"
say "+-------------+-------------------------------+"
say "| 54          | KSC5601                       |"
say "+-------------+-------------------------------+"
say "| 55          | TCHINESE                      |"
say "+-------------+-------------------------------+"
say "| 56          | RESTART                       |"
say "+-------------+-------------------------------+"
say "| 99          | UNKNOWN                       |"
say "+-------------+-------------------------------+"
return
```

SOLUTION 2

Here I will concentrate on the differences to AISGET. The main difference is that the routine now sends files to the server. These files, produced per agent/PC user, are obtained by extracting information from a single created on the mainframe.

The first step is to sort the dataset on the agent/PC user key, position 7 of length 6 characters.

This is particularly important because the routine simply sends the files as and when they are completed. If any agent/PC user has data in more than one location in the input file, then these will be treated as files for separate users. The result being that the last file and only the last file will exist, previous files being simply overwritten.

The name of the files for one run of the routine are always the same (rmyymmdd.rml and rmyymmdd.eti) but their location varies in accordance with the userid of the owning agent/PC user. The location is 'root/xxxxxxx/rmlhost', where xxxxxxx is the 6 digit agent/PC user key padded with 0 to make it 7 characters long.

The routine reads the whole input into internal storage with one command, and then accesses it via an array structure. This was chosen because the input file was not too big and it was easier to code. For larger data volumes it is probably not the best method.

To make the transfer easier to control, I have introduced the use of temporary datasets, which are dynamically allocated in the routine. The first temporary dataset is for a control file requested by the end user. This is produced just once per job; however it is sent together with the individual data file to each user directory. The second temporary dataset is constructed per data owner (agent). Basically as the agent/PC user key changes a couple of things happen. First a temporary dataset is allocated, then the records are extracted to the dataset from the array using a 'from' and 'to' range. Next comes the preparation of the FTP commands, written to a stack, and finally the temporary datasets are transferred to their new location on the server via FTP, using the stack as command input.

AISPUT JCL

```
//FTPRETRN JOB ,'USERID',
//             MSGLEVEL=(1,1),MSGCLASS=X,
//             CLASS=A,REGION=4M,
//             NOTIFY=&SYSUID,
//             USER=,GROUP=,PASSWORD=
//*
//          SET    ENV=T
//*
//* LIB: USERID.JCL.CNTL(FTPRETRN)
//* DOC: TRANSFER OF FILES RMJJMMDD.RML PER USER FROM SINGLE MAINFRAME
//*      DATASET &ENV.HLQ.FTPRETRN.AØØØØ1.BESTR. AFTER SORTING FROM
//*      POSITION 7 IN LENGTH 6
//*-------------------------------------------------------------------
//* SORT STEP
//*-------------------------------------------------------------------
//*
//STEP1    EXEC PGM=SORT
//SORTIN  DD DSN=&ENV.HLQ.FTPRETRN.AØØØØ1.BESTR.GØØØ1VØØ,
//            DISP=SHR
//SORTOUT DD DSN=&ENV.HLQ.FTPRETRN.AØØØØ1.BESTR.GØØØ1VØØ,
//            DISP=SHR
//SYSOUT  DD SYSOUT=*
//SYSIN   DD  *
 SORT FIELDS=(7,6,CH,A)
/*
//*
//*******************************************************************
//* CALL:
//* AISPUT <FILEDATE YYMMDD> <USER> <PASS> <HLQ>
//*******************************************************************
//STEP2    EXEC TSOBATCH
//SYSPROC DD DSN=SYNPU1.SYST.CLIST,DISP=SHR
//           DD DSN=SYNPU1.V2R6.CLIST,DISP=SHR
//OUTPUT  DD SYSOUT=*
//RETRNM  DD DSN=&ENV.HLQ.FTPRETRN.AØØØØ1.BESTR.GØØØ1VØØ,
//              DISP=SHR
//SYSTSIN DD *
%AISPUT Ø1Ø4Ø7 USERIDØ2 PASSWORDØ2 TMPHLQ
//*
```

AISPUT.EXEC

```
/* REXX ************************************************************ */
/*                           ** AISPUT  **                           */
/*                                                                   */
/*  PUT RETURN FILES FROM ONE INPUT FILE TO SEVERAL FILES ON SERVER  */
/*                                                                   */
/*  Dataset sorted on positions 7-12 in previous step                */
/*  workstation-id 7 digits, 1st 6 from positions 7-12 padded with Ø  */
/*                                                                    */
/*  read input file into array                                        */
/*  build temp file Etiketten  (control file)                         */
/*  loop until end of file ( writing all records for a workstation-id */
/*                           to a temp file, then transfer with FTP to*/
/*                           root/workstation-id/rmlhost the file with*/
/*                           name rmyymmdd.rml together with control  */
/*                           file rmyymmdd.eti )                      */
/*    delete temp file                                                */
/*    loop until workstation-id changes or EOF                        */
/*      write record to temp file                                     */
/*    close temp file                                                 */
/*    connect to TCP/IP server "ais-ablage"  (AIS-FTP)                */
/*    change directory to root/xxxxxxx/rmlhost                        */
/*    transfer temp file to rmyymmdd.eti on AIS-FTP server            */
/*    transfer 2nd temp file to rmyymmdd.dat on AIS-FTP server        */
/*                                                                    */
/* ****************************************************************** */
trace o                             /* trace control r=on , o=off     */
arg filedate aisuser aispass env .  /* receive parameters from JCL    */
                                    /* filedate = yymmdd              */
                                    /* aisuser = userid               */
                                    /* aispass = password             */
                                    /* end = environment hlq          */
call init                           /* initialize                     */
"execio * diskr retrnm (STEM retrnm. "
                          /* read retrnmeldung file into array        */

if retrnm.Ø = Ø then return         /* if empty exit routine          */
                                    /* = nothing to transfer          */
do i = 1 to retrnm.Ø                /* process array line by line     */
  if i = 1 then                     /* first record? then initialize  */
  do
    call firstrec
  end
  else                              /* otherwise continue             */
  do
    call nextrec
  end
end
return rc                           /* return to JCL with return code */

/* ****************************************************************** */
init:
/* initialization                                                     */
  retrnrml   = "rm"filedate".dat"   /* NT name = rmyymmdd.dat         */
  retrneti   = "rm"filedate".eti"   /* NT name = rmyymmdd.eti         */
/* allocate temporary file for timestamp file */

  tmpfile1 = env || '.TEMPETI.DATAF'     /* envhlq.TEMPETI.DATAF      */
  "listds '"tmpfile1"'"
  if rc = Ø then do                      /* does file exist ?         */
    "delete '"tmpfile1"'"                /* if so then delete it      */
    say tmpfile1 " deleted"
  end
  say " allocating " tmpfile1
 "alloc da('"tmpfile1"') dd(tempeti) new blksize(35) reuse ,
      lrecl(35) catalog recfm(f b) dsorg(ps) "
  if rc ¬= Ø then do
    say "error in alloc, rc = " rc
    exit
  end

/* ************************************** */
/* prepare put statement for control file */

  rc = LISTDSI(tempeti "FILE")
  dstempeti = SYSDSNAME
  dstempeti = "'"dstempeti"'"
  puteti   = "put " dstempeti
  puteti   = puteti retrneti
                        /* "put envhlq.TEMPETI.DATAF rmyymmdd.eti"  */

/* build control statement date:time stamp */

  disptime = time('N')
  disptime1 = substr(disptime,1,2)
  disptime2 = substr(disptime,4,2)
  disptime3 = substr(disptime,7,2)
  disptime  = disptime1 || disptime2 || disptime3
  filedate = date('S')
  timestmp = filedate || disptime
  eti.1 = "ØØØØ1 Ø"
  eti.1 = eti.1 || timestmp

  /* write control statement to temp control file */
  "execio * diskw tempeti (STEM eti. FINIS "
return

/* ************************************************************ */
firstrec:
  call ftpscmds           /* display list of FTP subcommands          */
  call ftpcodes           /* display list of FTP reply codes          */

  startpos = 1            /* record range "from" set to 1             */
  currentpc = substr(retrnm.1,7,6)     /* current workstation-id nmr  */
  currentpc = currentpc || 'Ø'         /* current workstation-id nmr  */
return
/* ************************************************************ */
```

```
nextrec:
  newpc = substr(retrnm.i,7,6)      /* new workstation-id nmr. ?      */
  newpc = newpc || 'Ø'              /* workstation-id padded with     */
                                    /* Ø to 7 digit number            */

  if newpc ¬= currentpc then        /* write and transfer file -      */
    do                              /* when workstation ID changes    */
      endpos = i - 1
      say "output for " currentpc
      call writefile
    end

  if i = retrnm.Ø then              /* when last record reached       */
  do                                /* write and transfer file        */
    endpos = i
    say "output for " currentpc
    call writefile
  end
return

/* ************************************************************** */
writefile:

/* allocate temporary file for rerturn file */

  tmpfile2 = env || '.TMPRETRN.DATAF'    /* "envhlq.TMPRETRN.DATAF"   */
  "listds '"tmpfile2"'"
  if rc = Ø then do                 /* does file exist?               */
    "delete '"tmpfile2"'"           /* then delete it                 */
    say tmpfile2 " deleted"
  end
  say "allocating " tmpfile2
  "alloc da('"tmpfile2"') dd(tmpretrn) new blksize(46) reuse ,
       lrecl(46) catalog recfm(f b) dsorg(ps) "
  if rc ¬= Ø then do
    say "error in alloc, rc = " rc
    exit
  end

/* ************************************************************** */
/* prepare put statement for FTP */

  rc = LISTDSI(tmpretrn "FILE")      /* dsn name from DD statement    */
  dstmpretrn = SYSDSNAME             /* SYSDSNAME = JCL dataset name  */
  dstmpretrn = "'"dstmpretrn"'"      /* in quotes to avoid prefix     */
  putretrn   = "put "dstmpretrn      /* build FTP "put" statement     */
  putretrn   = putretrn retrnrml
                          /* "put envhlq.TMPRETRN.DATAF rmyymmdd.dat  */
/* ************************************************************** */
  k = 1
  do j = startpos to endpos         /* write all records for a user   */
    say "user."j"=" retrnm.j        /* display user name to output    */
    retrnmtmp.k = retrnm.j
    k = k + 1
  end
  "execio * diskw tmpretrn (FINIS STEM retrnmtmp. "
                                 /* write to temp file            */
  drop retrnmtmp.                /* clear temporary storage array  */

  startpos = endpos + 1          /* new start pos for next user    */
  say "transfer for " currentpc
  call senddata                  /* call transfer routine          */
  currentpc = newpc              /* change to next user            */
return

/* ************************************************************** */
/* build FTP commands together in the stack then call FTP */
senddata:
  changedir = "cd "currentpc"/rmlhost"

  "newstack"
  queue aisuser aispass          /* userid password                */
  queue changedir                /* change directory               */
  queue putretrn                 /* put datafile >> server         */
  queue puteti                   /* put control file eti >> server */
  queue "quit"                   /* exit FTP                       */
  queue ""

/* call FTP using stack as input                                   */
  "ftp ais-ablage    (exit "     /* ftp to "ais-ablage"  host      */

   ftpcode = rc                  /* FTP returns 5 digit returncode */
   ftpcode = right(ftpcode,5,'0')
   ftpscmd = substr(ftpcode,1,2)    /* subcommand = 1st 2 digits      */
   ftprply = substr(ftpcode,3,3)    /* reply code = last 3 digits     */
                       /* display subcommand and reply code        */
   say "*****************************"
   say "FTP SUBCOMMAND = " ftpscmd
   say "FTP REPLY CODE = " ftprply
   say "*****************************"
   ftpretcd = ''
   select
          when ftpscmd = "00"  then ftpretcd = 0
          when ftprply = "110" then ftpretcd = 0
          when ftprply = "120" then ftpretcd = 0
          when ftprply = "125" then ftpretcd = 0
          when ftprply = "150" then ftpretcd = 0
          when ftprply = "200" then ftpretcd = 0
          when ftprply = "211" then ftpretcd = 0
          when ftprply = "212" then ftpretcd = 0
          when ftprply = "213" then ftpretcd = 0
          when ftprply = "214" then ftpretcd = 0
          when ftprply = "215" then ftpretcd = 0
          when ftprply = "220" then ftpretcd = 0
          when ftprply = "221" then ftpretcd = Ø
          when ftprply = "226" then ftpretcd = Ø
          when ftprply = "23Ø" then ftpretcd = Ø
          when ftprply = "25Ø" then ftpretcd = Ø
          when ftprply = "257" then ftpretcd = Ø
          when ftprply = "331" then ftpretcd = Ø
          when ftprply = "332" then ftpretcd = Ø
          when ftprply = "55Ø" then ftpretcd = 4
   otherwise ftpretcd = 8
   end
   if ftpretcd > 4 then
   do
     "delstack"           /* delete unused commands from stack        */
     exit ftpretcd                  /* exit when FTP call in error    */
   end
/* ************************************************************** */
   "delstack"             /* delete unused commands from stack        */

return ftpretcd
/* ************************************************************** */

ftpcodes:

/* table of FTP reply codes to be displayed in JOB output list        */
...
/* see aisget */
...
return

/* ************************************************************** */
ftpscmds:

/* table of FTP subcommand codes to be displayed in JOB output list   */
....
/* see aisget */
....

return
```

Editor's note: In the next edition of MVS Update *we will see how the project has been extended to encompass distribution in other areas.*

Rolf Parker
Systems Programmer (Germany)

Dynamic Channel-path Management – deployment issues

In the last edition of *MVS Update* we reviewed some of the advantages of running Dynamic Channel-path Management (DCM) in the zArchitecture. To complete this overview we will highlight some of the hardware and software planning issues associated with its deployment.

HARDWARE ISSUES

DCM currently supports only the more recent DASD devices. Therefore, non-synchronous control units such as tape, CTC, comms, printers, and those control units attached through an ESCON converter are not supported. There are two principal reasons for this. First, DASD tend to be more response time-sensitive than most devices, and, second, DASD can deliver large data volumes, requiring large channel bandwidths; however, when these control units are inactive large quantities of channel capacity are tied up.

All the managed paths to a control unit need to be defined as shared and attached through a switch. Although, it is possible to connect the non-managed paths point-to-point. To achieve the maximum connectivity, all DASD paths should be routed through a switch.

DCM requires information about the system set-up. To do this it is essential that the Control Unit Port (CUP) on every switch that is attached to managed channels is defined to HCD. This is different to prior levels of OS/390 where it was recommended that users define the CUP, but it was not an absolute requirement. With z/OS it is essential, because if the CUP is not defined in HCD, and accessible to every LPAR in a cluster, then DCM will be unable to gather the topology information it needs.

A useful tip, when you are defining the control unit device in HCD, is to use the switch ID as the last two digits of the device number. This makes it easier to link the device number back to the switch definition in HCD. For example, if the switch ID is 3C, the device number could be B43C.

z/OS Version 1 Release 1 provides two new commands to help manage switches. The VARY SWITCH command allows or bars DCM from using a specified port. The D M=SWITCH command displays information about what is connected to each switch port. This also provides information on whether a port can be used by DCM.

Since OS/390 Version 2 Release 1, systems connected to a director will check the CUP at a set time period to see whether there have been any relevant changes to the director configuration. The problem is that if there are a large number of systems connected, this checking can lead to sluggish response times. This is why IBM recommends that the MIH time for the director devices is set to three minutes. The MIH setting can be changed in the IECIOS*xx* member.

Another consideration is the mixture of ESCON or FICON Bridge and FICON Native channels on a control unit. Although, these can be mixed on a control unit, it is strongly recommended that ESCON and FICON Native channels are mixed only during transition to DCM. It is definitely not a long-term solution.

When deciding which channels should be managed, users should try to spread the managed channels across as many switches as possible. This will provide DCM with a greater selection of potential paths, and therefore will increase availability.

SOFTWARE

Essentially, DCM is responsible for adding and removing paths to control units. Therefore, the only software products that are affected by its presence are those that monitor, define, or control the configuration.

It is important to note that the only operating systems that can use DCM is z/OS Version 1 Release 1 and higher, in zArchitecture mode. You will note in the literature that OS/390 Version 2 Release 10 ‘supports’ DCM, but this merely extends to the use of some of the new DCM commands. A Release 10 system cannot actually use a managed path.

New features in PL/I for z/OS

The latest release of PL/I for z/OS and OS/390 Version 3 Release 1 contains enhancements that are aimed at improving performance, compatibility, and flexibility. It also contains new features that clearly illustrate IBM’s commitment towards creating a completely integrated environment for users of OS/390 and z/OS. These features include better Java interoperability, an XML parser, improved compatibility with older PL/I compilers, plus integrated CICS and SQL preprocessors. This new functionality will result in cost and time savings, and will make the jobs of systems programmers a little bit easier.

The first improvement of note is the enhancements relating to back compatibility. There is now considerable object code compatibility with the code generated by the PL/I for MVS and OS PL/I Version 2 Release 3 compilers. It supports the CMPAT(V2) and CMPAT(V1) options. This is an important step forward because, previously, only non-string scalars could be passed to or received from old code, but now strings, arrays, and structures can too. Object code produced by the VisualAge PL/I for OS/390 Version 2 compiler is compatible with that produced by the Version 3 Release 1 compiler, as long as the matching CMPAT option is used. The new version of PL/I also makes it easier to use existing objects libraries and linking schemes because of its support of the NORENT compiler option.

Many of you probably run CICS at your sites. With the new integrated CICS pre-processor, there is no need to run a separate job step that precompiles EXEC CICS statements into PL/I code. Instead, the compile step will handle EXEC CICS statements in the same way that it handles any use of the macro facility. Similarly, with the integrated SQL preprocessor, there is no need to run a separate job step that pre-compiles EXEC SQL statements into PL/I code.

We have seen Java functionality incorporated across the IBM product set. With this latest version of PL/I Java interoperability is addressed with a thread-safe PL/I library, multi-threading statements (ATTACH, WAIT, DETACH) as part of the PL/I language supported by the compiler, and IEEE, as a representation for floating-point values and variables.

Version 3 Release 1 also provides an XML parser that can be invoked by calling a new PL/I built-in subroutine, giving PL/I programs the ability to parse XML documents (this can be in EBCDIC, ASCII, or UTF-16 Unicode) directly within their PL/I applications. The XML support can be used to enhance existing IMS transactions written in PL/I in a B2B environment by receiving and sending XML documents.

Other enhancements include support for the ANSWER statement in the macro facility, support for the CSECT and NAME compiler options, and better detection of uninitialized variables.

The mainframe interactive debug tool is offered with the Enterprise PL/I compiler in the full-function version, and supports Enterprise COBOL for z/OS and OS/390, COBOL for OS/390 and VM, COBOL for MVS and VM, Enterprise PL/I for z/OS and OS/390, VisualAge PL/I for OS/390, PL/I for MVS and VM, z/OS C/C++ optional feature, and OS/390 C/C++ optional feature.

Systems Programmer (UK) © Xephon 2002

The effects of implementing snapshot copy – update

The article called *The effects of implementing snapshot copy* in Issue 183 of *MVS Update*, December 2001, has a few printing errors which may confuse a careful reader. On page 17, Figure 2: *Overall review of production batch jobs only*, the column headers got mixed up. The correct sequence of columns headers is:

```
Year  / Total number of batch jobs /  Total  elapsed time /  Average elapsed
time / Total CPU time / Average CPU time.
```

On page 18, Figure 4: *Tape mounts for production jobs by group*, the column headers were incorrectly specified and are not accurate. The correct column headers are:

```
Jobs not using tapes / Jobs using tapes / Total.
```

In addition, the second figure at row cell level (one after /) represents the row per cent.

© Xephon 2002

MVS news

Landmark Systems has announced Version 1.1 of its TMON for Unix System Services (USS) for managing USS workloads and resource consumption within OS/390 and z/OS sites.

The latest version supports z/OS, including the z/FS file system, and contains new bits such as a diagnostic manoeuverability feature, allowing technical staff to better and more easily answer Help Desk-type calls, particularly when callers report trouble and only have a single piece of information to identify the issue. Version 1.1 also has improved process navigation, with summary statistics available for all process activity. It lets companies start research at a summary level and drill down to the detailed reports; said to be useful for companies with a number of USS applications because it obviates the need to comb through extensive detailed information as a first step in research or proactive management efforts.

The new version now also enables exception monitoring of the Hierarchical File System (HFS) and Temporary File System (TFS), giving users early warnings of file systems filling up. And it lets users access commonly used menus more quickly and easily through new Quick Access menus.

For further information, contact:

Landmark Corporation, 12700 Sunrise Valley Drive, Reston, Virginia 20191-5804, USA.

Tel: (703) 464 1300
Fax: (703) 464 4918

http://www.landmark.com

* * *

IBM has announced Debug Tool for z/OS and OS/390 Version 1 Release 3, for compiled applications, which has support for Assembler via disassembly view, plus a range of new functions. It is the renamed version of CoOperative Development Environment/370, or CODE/370.

Besides Assembler, the debugger currently supports applications written in C/C++, COBOL, High Performance Java (HPJ), and PL/I. Other new features include support for multi-threaded applications, including new support delivered in Version 3 of the COBOL and PL/I compilers and run times. There is also support for dynamic debug of COBOL applications, interactive debug of JES batch applications from a 3270 interface, and an ISPF set-up and invocation utility to help start debug sessions.

More specifically, Debug Tool can now debug COBOL production load modules where the new compiler option has been utilized to create the debug information in a separate file.

Debug Tool works in CICS, IMS, DB2, WebSphere, TSO, JES/Batch, Unix System Services, and CMS. Via the full-screen interface, users can interactively debug any application as it runs, including batch applications. The tool can be invoked when an application is initialized, or it can be started dynamically when a condition occurs. The application itself can also invoke Debug Tool.

Contact your local IBM representative for further information.

http://www.ibm.com/servers/esrvers

* * *