3

MVS update

Special edition

In this issue

A xephon PUBLICATION

MVS Update

Published by

Xephon Inc
PO Box 550547
Dallas, Texas 75355
USA

Phone: 214-340-5690
Fax: 214-341-7081

Editor

Trevor Eddolls
E-mail: trevore@xephon.com

Publisher

Colin Smith
E-mail: info@xephon.com

Subscriptions and back-issues

A year's subscription to *MVS Update*, comprising twelve monthly issues, costs $505.00 in the USA and Canada; £340.00 in the UK; £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 2000 issue, are available separately to subscribers for £29.00 ($43.50) each including postage.

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.

Contributions

When Xephon is given copyright, articles published in *MVS Update* are paid for at the rate of $160 (£100 outside North America) per 1000 words and $80 (£50) per 100 lines of code for the first 200 lines of original material. The remaining code is paid for at the rate of $32 (£20) per 100 lines. To find out more about contributing an article, without any obligation, please download a copy of our *Notes for Contributors* from www.xephon.com/nfc.

Printed in England.

Quick HFS free space report

Every now and then it happens that a storage administrator or system programmer encounters an HFS file system flagging an out-of-space condition. This can happen when installing a new application, a product that is not part of the standard ServerPac order, or simply because of an application's high write activity. By taking a proactive approach to planning and monitoring HFS free space, one can avoid the situation where the file/volume becomes so full that there is a risk of not writing out data from the buffer when it is time to unmount the HFS dataset (in order to add candidate volumes). Thus, when installing a new application or a product that installs into the HFS, consider doing the following:

- Creating new directories where the files associated with the new application/product will be installed.
- If possible, creating a new HFS dataset and mounting it for the new directory. After installation of the product/application, all the files will reside in the new HFS dataset.
- Keeping new applications/products in different HFS datasets, which offers better file system management while maintaining a more stable root file system. This also ensures easier maintenance when applying service.

Before proceeding any further, it might be helpful to remember that in order to update an HFS file, DFSMS 1.5 uses a shadow write technique that maintains duplicate pages in the dataset until the update is completed. This technique improves the integrity of the data, but on the other hand it also requires that there always must be a certain number of free pages within the dataset. In order to shadow write, it needs to make a copy of the attribute page(s) it is updating along with any index pages that need to change because of pointers to different pages. Usually the tree depth is small and it needs only a few pages. HFS keeps a 30-60 byte block in reserve for this purpose in case the file system runs out of user space. Please note that some free space is needed even if only reading data! It is good

to know that a new parameter has been added to the PARM keyword on the MOUNT statement to control the number of pages HFS should reserve for Sync processing of the file system metadata. When this parameter is specified on the MOUNT statement, it will override HFS's internal reserved page estimation algorithm. This parameter should be used only if it you find that the internal algorithm is not providing the desired results. The new parameter is SYNCRESERVE(*nn*), where *nn* represents the percentage of the file system space to be reserved for the Sync shadow write mechanism. Valid values for *nn* are between 1 and 50. The trade-off is that less space will be available for user file data in the HFS (see APAR OW43771). How to recover from an out-of-space error during sync on the HFS root file system or /etc directory is described in great detail by info APAR II13537.

As already stated, it is necessary to monitor the utilized space within each dataset regularly, as well as to take preventative action when you find a dataset that is close to exhausting its available space. The storage administrator's attention should be especially focused on high I/O activity datasets, since one may see a performance benefit by allocating particularly active application files across a number of different HFS datasets as well as predicting and thus preventing out-of-space errors. Unfortunately, there do not seem to be many tools available that help you easily to identify your most active files. There is some activity data recorded in SMF type 92 records that may prove helpful, but it will take some manipulation on your part.

To help alleviate the burden of monitoring HFS free space, a simple yet easy-to-use REXX procedure was written. It uses the USS command df, which displays the amount of free space in the file system. By default the df measures space in units of 512-byte disk sectors. One can specify a particular file system by naming any filename on that file system. If you do not give an argument, df reports on space for all mounted file systems known to the system. The total space reported is the space in the already allocated extents (primary and any already allocated secondary extents) of the HFS dataset that holds this file system. Therefore, the total space may increase as new extents are allocated. An additional option, –S, was used because it provides SMF I/O accounting for mounted files. A

detailed description of the df command and the output it produces can be obtained from *z/OS Unix System Services Command Reference* (SA22-7802). The report this procedure produces (HFS free space and I/O activity) can shed some light on your HFS file system status and activity, thus helping you to prevent the out-of-space problems:

```
                                                                    - Dir
I/O blocks -  Total bytes
Filesystem          Allocated   Used   Free Free %  reads writes total
read write   read written
-------------------------------------------------------------------------
-------------------------
OMVS.DB2710.DSNHFS       3600   1051   2549  70.80    70     0    2348
63     0   154550     0
OMVS.DB2710.HFS.DB2TX   10800  10573    227   2.10   144     0    3322
148     0   305247     0
OMVS.DB2710.HFS.DB2EXT  10800   5899   4901  45.37    54     0    2498
74     0   215262     0
OMVS.DB2.HFSWHSE         5760    112   5648  98.05    43     0    1092
181     0   640123     0
OMVS.DTW.SDTWHFS         4680   2233   2447  52.28   174     0    4158
165     0   371080     0
OMVS.WAS.CONFIG.HFS     32400  14942  17458  53.88  4008     0  567710
5227     0  3635399     0
OMVS.JV390              81900   3631  78269  95.56  1034     0   86470
1104     0  2723247     0
OMVS.AS390             135000  39771  95229  70.54   736     0   57282
1402     0  4192596     0
OMVS.ILMSPDM             3600   3578     22   0.61     6     0      10
5     0     4370     0
OMVS.ILMUC               3600   3578     22   0.61     6     0      10
5     0     4370     0
OMVS.VAR                 3600   3475    125   3.47    71     2     807
53     4    95267  9969
OMVS.ETC                10800  10171    629   5.82  2740   131   21013
3168   131  9040493  6727
OMVS.HFS.ROOT          882000  18003 863997  97.95 32473   243 1534188
79883   243  6302481 34581
```

Detecting out-of-space conditions was greatly facilitated by applying APAR OW44631, which has provided an early warning capability to let users know when a file system is becoming full. This APAR shipped a new function for HFS called HFS MONITOR.

It provides support to monitor how full an HFS is and will issue operator message IGW023A when the HFS exceeds a user-specified full threshold. The fullness of an HFS is based on the number of

pages currently in use versus the currently allocated HFS file system size. The user can specify threshold and increment values via the parmlib member BPXPRM*x* to set default values to be used for all HFS file systems. The values can also be specified on the Mount command to set values for a specific file system. Parameters on the Mount command will override parmlib values. If no values are specified in either place, no threshold checking will be done. Message IGW023A will be automatically removed if the HFS is extended to bring the HFS below its threshold, if files are deleted to bring the HFS below its threshold, or if the HFS is unmounted.

The new parameter syntax is:

```
FSFULL(threshold,increment)
```

where *threshold* means that when the HFS exceeds threshold% full, operator messages will start to be generated (default is 100%). *increment* means that with each increment% increase/decrease in file system fullness beyond the threshold, the message will be updated (default is 5%).

HFSFREE

```
/* REXX ***************************************************************
   Procedure: HFSfree
   Description: Get information on HFS free space and I/O activity
   Install: - Download BPXWUNIX function (a part of "REXX Function
              Package for REXX in OpenEdition") from the IBM's "USS
              Tools and Toys page"
            - Restore it using the TSO/E receive inda() command.
            - Place this where REXX EXECs can be found.
  ********************************************************************/
signal ON ERROR
Address TSO
userid=SYSVAR(SYSUID)
outds  =userid||'.hfs.out'               /*  Change dataset name */
x = MSG('ON')                            /* to fit your standards */
if SYSDSN(outds) = 'OK'
Then "DELETE "outds" PURGE"
"ALLOC FILE(PRC) DA("outds")",
   " UNIT(SYSALLDA) NEW TRACKS SPACE(2,1) CATALOG",
   " REUSE LRECL(150) RECFM(F B)"
/*-----------------------------------------------------------------*/
/* Allocate BPXWUNIX load library:                                 */
/* supply the name of received REXX function load library          */
```

```
/*-------------------------------------------------------------------*/
 arg hlq
 if hlq = "" then HLQ = 'uid.REXXFUNC.LOAD'
Address ISPEXEC
"LIBDEF ISPLLIB DATASET ID('"hlq") STACK"
call syscalls 'ON'
/*-------------------------------------------------------------------*/
/* Return USS information                                            */
/*-------------------------------------------------------------------*/
Address SYSCALL
'uname sys.'
/*-------------------------------------------------------------------*/
/* Print headers and labels                                          */
/*-------------------------------------------------------------------*/
sis.1 = left('HFS Quick report - produced on:',32,),
        ||left(' ',1,' ')||left(date(),11),
        ||left(' ',1,' ')||left('at ',3,' '),
        ||left(time(),10)
sis.2 = left(' ',1)
sis.3 = left('System identification:',22)
sis.4 = left('Sysname: ',11)||sys.U_SYSNAME
sis.5 = left('Version: ',11)||sys.U_VERSION
sis.6 = left('Release: ',11)||left(sys.U_RELEASE,10)
sis.7 = left('Node   : ',11)||left(sys.U_NODENAME,10)
sis.8 = left('Hardware:',11)||left(sys.U_MACHINE,10)
sis.9 = left(' ',112,' ') left('- Dir I/O blocks -',21),
        left('Total bytes',12)
hf.1  = left('Filesystem',10) left(' ',13)   left('Allocated',13),
        left('Used',8)        left('Free',5) left('Free %',6),
        left('Mounted on',10) left(' ',22,' ') left('reads',7),
        left('writes',7)      left('total',8)  left('read',6),
        left('write',6)       left('read',6)   left('written',7)
hf.2  = left('-',150,'-')
/*-------------------------------------------------------------------*/
/* Get HFS data                                                      */
/*-------------------------------------------------------------------*/
Address SH
  call BPXWUNIX "df -S",,out.
 dfrc = rc
  If dfrc  <>0 Then Do
    Say "Return Code         =" rc
    Say "OMVS Return Value   =" retval
    Say "OMVS Return Code    =" errno
    Say "OMVS Reason Code    =" errnojr
  End
  j = 1
  Do i = 2 to OUT.0                 /* Process each entry returned*/
  parse var out.i  mount . '(' HFS ')' rawdata .
  parse var  rawdata  ava  '/' tot
  fre  =  tot   - ava
```

```
  pct = trunc((1-(ava/tot))*100, 2); i = i + 1;
  read = word(out.i,5);   i =i + 1;
  write= word(out.i,5);   i =i + 1;
  ioblk= word(out.i,6);   i =i + 1;
  reblk= word(out.i,6);   i =i + 1;
  wrblk= word(out.i,6);   i =i + 1;
  byread = word(out.i,7); i =i + 1;
  bywrite= word(out.i,7)
  HFS = left(HFS,25)
rrw.j=left(HFS,25),                  /* Filesystem               */
      right(tot/8,8),                /* Total 4K pages allocated  */
      right(ava/8,8),                /* Available pages (4K)      */
      right(fre/8,8),                /* Free pages (4K)           */
      format(pct,3,2),               /* Percent free              */
      left(mount,32),                /* Mounted on                */
      right(read,7) ,                /* Number of reads           */
      right(write,7) ,               /* Number of writes          */
      right(ioblk,7) ,               /* Number directory I/O block*/
      right(reblk,7) ,               /* Number read I/O blocks    */
      right(wrblk,7) ,               /* Number write I/O blocks   */
      right(byread,7) ,              /* Total number bytes read   */
      right(bywrite,7) ; j = j + 1   /* Total number bytes writte */
    End
call syscalls 'OFF'
/*-------------------------------------------------------------------*/
/* Write out USS System info and HFS info data                       */
/*-------------------------------------------------------------------*/
Address ISPEXEC "LIBDEF ISPLLIB";
Address TSO
"EXECIO * DISKW PRC (STEM sis.)"
"EXECIO * DISKW PRC (STEM hf. )"
"EXECIO * DISKW PRC (STEM rrw. )"
 /*-------------------------------------------------------------------*/
 /* Close & free allocated report file; then display result          */
 /*-------------------------------------------------------------------*/
 "EXECIO 0 DISKW PRC (FINIS "
  "free FILE(PRC)"
   Address ISPEXEC
  "ISPEXEC BROWSE DATASET('"outds"')"
 exit 0
 /*-------------------------------------------------------------------*/
 /* Error exit routine                                                */
 /*-------------------------------------------------------------------*/
 ERROR: say 'The following command produced non-zero RC =' RC
        say SOURCELINE(SIGL)
        exit
```

Mile Pekic
Systems Programmer (Serbia)

Checking tape volids

We received a delivery of pre-initialized and pre-labelled 3490-type cartridges. Unfortunately, the internal and external labels did not match in all cases! I wrote the following program to cross-check tape volids. Sample JCL is included in the program.

Output for a mismatch (via WTO, also gives RC=4):

```
 +============================================================
+CHKCARTS: MISMATCH, EXTERNAL LABEL="702404", VOL1="702434"
 +============================================================
```

Output for a match (also gives RC=0):

```
 +CHKCARTS: CART "702404" VERIFIED OK...
```

PROGRAM CODE

```
CHKCARTS TITLE 'CHECK VOL1 LABELS AGAINST EXTERNAL LABELS'
*********************************************************************
*    MODULE:   CHKCARTS                                             *
*    DESC:     READ A CART BLP AND VERIFY THAT THE VOL1 MATCHES     *
*              WHAT WE ARE EXPECTING...                             *
*                                                                   *
*              THIS IS BECAUSE PRE-INITIALIZED CARTRIDGES WERE      *
*              DELIVERED TO US WITH THE INTERNAL/EXTERNAL LABELS    *
*              NOT MATCHING.                                        *
*    JCL:                                                           *
*           ------------------------------------------------------  *
*          ¦   //jobcard                                         ¦  *
*          ¦   //S1      EXEC PGM=CHKCARTS                       ¦  *
*          ¦   //STEPLIB  DD  DSN=<user.loadlib>,DISP=SHR         ¦  *
*          ¦   //SYSPRINT DD  SYSOUT=*                          ¦  *
*          ¦   //INPUT    DD  DSN=anyname,UNIT=3490,             ¦  *
*          ¦   //         VOL=SER=nnnnnn,LABEL=(1,BLP),          ¦  *
*          ¦   //         EXPDT=98000,RECFM=U,BLKSIZE=4096       ¦  *
*           ------------------------------------------------------  *
*                                                                   *
*********************************************************************
         PRINT NOGEN
*-------------------------------------------------------------------*
* HOUSEKEEPING, ETC...                                              *
*-------------------------------------------------------------------*
CHKCARTS CSECT
```

```
         BAKR  R14,Ø                    SAVE CALLER DATA ON STACK
         LR    R12,R15                  GET ENTRY POINT
         USING CHKCARTS,R12             ADDRESSABILITY TO MODULE
*-------------------------------------------------------------------*
* GET JFCB INFORMATION FOR LATER CHECKING...                        *
*-------------------------------------------------------------------*
         RDJFCB INPUT                   GET JFCB FOR INPUT DATASET
         MVC   VOLXJFCB,JFCB+118        SAVE VOLSER FROM JFCB
*
         OPEN  INPUT
*
*-------------------------------------------------------------------*
* READ THE VOL1 FROM THE CART AND VERIFY THAT THE VOLSER ON THE CART  *
* MATCHES THAT SPECIFIED IN THE JCL...                              *
*-------------------------------------------------------------------*
READVOL1 DS    ØH
         GET   INPUT                  READ VOL1
         LR    R2,R1                  GET ADDRESS OF VOL1
         CLC   Ø(3,R2),=C'VOL1'       IS IT A VOL1?
         BNE   BADVOL1                NO...
         MVC   VOLSER,4(R2)           .....SAVE VOLSER
         CLC   VOLSER,VOLXJFCB        VOLSER IN HDR1 SAME AS IN JCL?
         BNE   MISMATCH               NO...OH DEAR...
         MVC   OKWTO+24(6),VOLSER     SHOW ITS OK
OKWTO    WTO   'CHKCARTS: CART "......" VERIFIED OK...',              X
               ROUTCDE=11
*-------------------------------------------------------------------*
* GET OUT OF HERE...                                                *
*-------------------------------------------------------------------*
RETURN   DS    ØH
         CLOSE (INPUT)
         L     R15,RETC                 SET RC
         PR    ,                        RETURN
*-------------------------------------------------------------------*
* COULD NOT IDENTIFY VOL1 RECORD...                                 *
*-------------------------------------------------------------------*
BADVOL1  DS    ØH
         WTO   MF=(E,WTOHDR),ROUTCDE=11
         MVC   WTOVOL1+26(47),Ø(R2)    MOVE RECORD TO WTO
         WTO   MF=(E,WTOVOL1),ROUTCDE=11
         WTO   MF=(E,WTOHDR),ROUTCDE=11
         B     RETURN
*-------------------------------------------------------------------*
* EITHER DSNAME OR VOLSER SPECIFIED IN THE JCL DOES NOT MATCH WHAT IS *
* IN THE HDR1...                                                    *
*-------------------------------------------------------------------*
MISMATCH DS    ØH
         WTO   MF=(E,WTOHDR),ROUTCDE=11
         MVC   WTOMISS+4Ø(6),VOLXJFCB  MOVE JFCB VOL TO WTO
```

```
         MVC   WTOMISS+55(6),VOLSER    MOVE VOL1 VOL TO WTO
         WTO   MF=(E,WTOMISS),ROUTCDE=11
         WTO   MF=(E,WTOHDR),ROUTCDE=11
         MVC   RETC,RC4                SET RC=4
         B     RETURN
*--------------------------------------------------------------------*
* E-O-V EXIT: SHOULD NEVER COME HERE, SO ISSUE MSG AND ABEND...       *
*--------------------------------------------------------------------*
EOVEXIT  DS    ØH
         LR    R3,R15                  GET ENTRY POINT ADDRESS
         USING *,R3                    ADDRESSABILITY TO EXIT
         WTO   MF=(E,WTOHDR),ROUTCDE=11
         WTO   'CHKCARTS: E-O-V EXIT DRIVEN...ABENDING',ROUTCDE=11
         WTO   MF=(E,WTOHDR),ROUTCDE=11
         DS    F                       RETURN FROM EXIT
*--------------------------------------------------------------------*
* E-O-F REACHED: SHOULD NEVER COME HERE, SO ISSUE MSG AND ABEND...    *
*--------------------------------------------------------------------*
BADEOF   DS    ØH
         LR    R3,R15                  GET ENTRY POINT ADDRESS
         USING *,R3                    ADDRESSABILITY TO EXIT
         WTO   MF=(E,WTOHDR),ROUTCDE=11
         WTO   'CHKCARTS: E-O-F REACHED...ABENDING',ROUTCDE=11
         WTO   MF=(E,WTOHDR),ROUTCDE=11
         DS    F                       RETURN FROM EXIT
         DROP  R3
*--------------------------------------------------------------------*
*        CONSTANTS                                                   *
*--------------------------------------------------------------------*
*
         YREGS
*
         DS    ØD
         DC    CL16'***EYECATCHER***'
VOLSER   DC    CL6' '
VOLXJFCB DC    CL6' '
RETC     DC    F'Ø'
RC4      DC    F'4'
*
INPUT    DCB   DDNAME=INPUT,                                          X
               DSORG=PS,                                            X
               MACRF=GL,                                            X
               EXLST=EXLSTIN,                                       X
               EODAD=BADEOF
*
EXLSTIN  DS    ØF
         DC    X'Ø7'                   SHOWS THIS IS A READ JFCB EXIT
         DC    AL3(JFCB)               ADDRESS OF JFCB AREA
         DC    X'86'                   SHOWS THIS IS AN E-O-V EXIT
```

```
         DC    AL3(EOVEXIT)            ADDRESS OF E-O-V EXIT
JFCB     DC    176C' '
*
WTOHDR   WTO    '=================================================X
               ====',ROUTCDE=11,MF=L
WTOVOL1  WTO    'CHKCARTS: DODGY VOL1 "..............................X
               ............."',ROUTCDE=11,MF=L
WTOMISS  WTO    'CHKCARTS: MISMATCH, EXTERNAL LABEL="......", VOL1="....X
               .."',ROUTCDE=11,MF=L
*
         END
```

Grant Carson
Systems Programmer (UK)

Multi-threading in COBOL

Unlike Java and C++, until recently you could not run your COBOL programs in more than one thread. If you tried to invoke the COBOL program from an application server in the second thread, it used to crash with a run-time error, 'COBOL found in multiple threads'. This heavily limited the possibility of using COBOL subroutines as building blocks for Web applications.

With Enterprise COBOL, IBM provides toleration-level support of POSIX threads and asynchronous signals. This article aims to explain the level of multi-threading support provided by COBOL and the associated features.

THREAD COMPILER OPTION

The THREAD compiler option enables a COBOL program to be multi-threaded (ie it can be called in more than one thread in a single process). This allows the program to run in multiple threads under batch, TSO, IMS, or Unix environments.

To support multi-threading, the program must be thread-safe, allowing multiple copies of it to run in the same run-unit. With the THREAD option, the storage and control blocks get appropriately

allocated at invocation, rather than per program. Also, additional serialization logic is generated automatically (which in turn can degrade performance).

Programs compiled with compilers pre-dating Enterprise COBOL are treated as compiled with NOTHREAD.

Note that a program that has been compiled with the THREAD option can:

- Run in CICS/IMS environments.
- Run in AMODE 24.
- Be used in a non-threaded application.
- Call programs that are not enabled for multi-threading (provided the application doesn't have multiple threads).

The following are the prerequisites to be met for running COBOL programs in a multi-threaded environment:

- All the programs within the run-unit must be compiled with the THREAD option.
- Programs must be compiled and link edited with the RENT option.
- Programs must be RECURSIVE.

RECURSIVE PROGRAM

A recursive call is where a called program can directly or indirectly execute its caller. For example, program X calls program Y, program Y calls program Z, and program Z then calls program X (the original caller).

To make a recursive call, you must code the RECURSIVE clause (IBM extension) on the PROGRAM-ID paragraph of the recursively called program. If the optional RECURSIVE clause is specified, the program can be re-entered recursively while a previous invocation is still active.

LOCAL STORAGE AND WORKING STORAGE

A multi-threaded program needs to be recursive and the persistence of the data for each call depends on whether it is in local storage or working storage.

Multiple threads that run simultaneously share a single copy of the WORKING-STORAGE data (statically allocated and initialized on first entry to a program, and available in the last-used state for the recursive invocations).

A separate copy of LOCAL-STORAGE data is allocated and made available for each call of a program (or invocation of a method); it gets released on returning from the program. If the VALUE clause is specified, the data is re-initialized to be the same for every invocation.

THREAD MANAGEMENT

Currently COBOL doesn't manage the threads; rather, it expects the application server or the calling program (in Java, C/C++, PL/I) to manage them. The threaded application must run within a single Language Environment enclave (created using the CEEPIPI routine).

FILE ACCESS IN MULTI-THREADED PROGRAMS

Multi-threaded COBOL programs can have file operations on QSAM, VSAM, and sequential files. Automatic serialization happens using the implicit lock on the file definition, during the execution of OPEN, CLOSE, READ, WRITE, REWRITE, START, and DELETE statements.

All threads of execution share the storage associated with the file definition (FD). It is important to note that the FD records and the records with the SAME RECORD AREA clause have the data available in the last-used state.

Serialization is not automatic between uses of these statements – and to avoid coding your own serialization logic (using POSIX APIs), IBM suggests the following:

- For input, the recommended usage pattern is OPEN, READ, process the record, CLOSE.
- For output, the recommended usage pattern is OPEN, construct the output, WRITE, CLOSE.
- Also, define the data items that are associated with the file (such as file-status data items and key arguments) in the LOCAL-STORAGE SECTION.

Sharing in a multi-threaded environment:

- For programs compiled with the THREAD option, the special registers (like ADDRESS-OF, RETURN-CODE, SORT-CONTROL, TALLY, XML-CODE, and XML-EVENT) are allocated (and reset to the initial value) on a per-invocation basis.
- All programs and all threads in an application share a single copy of UPSI switches. If you modify switches in a threaded application, you must code appropriate serialization logic.
- Indexes are normally allocated in static memory associated with the program and are in the last-used state when a program is re-entered. However, if compiled with the THREAD option, the indexes are allocated on a per-invocation basis and must be SET on every entry.
- If you compile your program with the THREAD compiler option, data that is defined in the LINKAGE SECTION is not accessible on subsequent invocations of the program. The address of the record in the Linkage section must be re-established for that execution instance.

Ending a program in a multi-threaded environment:

- When you use GOBACK from the first program in a thread, the thread is terminated. If that thread is the initial thread in an enclave, the entire enclave is terminated.
- With the EXIT program, the thread is not terminated unless the program is the first (oldest) one in the thread.

- With STOP RUN, the entire LE enclave is terminated, including all threads executing within the enclave.

Other factors to consider when multi-thread enabling COBOL programs:

- In a multi-threaded environment, a program cannot CANCEL a program that is active on any thread. If you try to cancel an active program, a severity-3 Language Environment condition occurs.
- A program executing on multiple threads can execute the same or different XML statements simultaneously.
- In a threaded application, the COBOL program can be interrupted by asynchronous signals, which the program should be able to tolerate. Alternatively, using C/C++ functions, the interrupts can be disabled by setting the signal mask appropriately.
- The RANDOM function can be used in threaded programs. For an initial seed, a single sequence of pseudo-random numbers is returned, regardless of the thread that is running when RANDOM is invoked.

Constraints related to multi-threaded COBOL programs:

- You cannot run multi-threaded applications in the CICS environment (though you can run programs compiled with the THREAD option).
- If the COBOL program has been compiled with the THREAD option with AMODE 24, then it can only be part of a non-threaded application.
- Nested programs are not supported for programs compiled with the THREAD option.
- Segmentation is not supported for programs compiled with the THREAD option.
- You cannot use the RERUN clause in I/O control in programs compiled with the THREAD option

- Priority-numbers are not valid for programs compiled with the THREAD option.
- You cannot specify ALTER or altered GO TO statements in programs compiled with the THREAD option.
- The MERGE statement is not supported for programs compiled with the THREAD compiler option.
- The SORT statement is not supported for programs compiled with the THREAD option.
- Do not use the STOP literal statement in programs compiled with the THREAD compiler option.
- Debugging sections are not permitted in programs compiled with the THREAD compiler option.
- Environments created by IGZERRE or ILBOSTP0 or by using the RTEREUS run-time options do not support the THREAD option.
- Do not use IGZBRDGE, the macro for converting static calls to dynamic calls, because it is not supported.
- Do not use the modules IGZETUN (for storage tuning) or IGZEOPT (for run-time options) because these CSECTs are ignored.

CONCLUSION

Currently what IBM provides with Enterprise COBOL is basic support for invoking COBOL programs in a multi-threaded environment. As can be seen, there are quite a few restrictions and behavioural differences when the THREAD option is used. Enabling existing COBOL programs to be multi-threaded can range from just recompiling (best) to modifying program logic that requires thorough testing. It is expected that multi-threading would be handled more naturally in COBOL in future versions.

Sasirekha Cota
Tata Consultancy Services (India)

Retrieve SMS information and DASD usage statistics using a REXX tool

The purpose of the tool DSINFO (DataSet INFOrmation) is to generate a report giving SMS (Storage Management Subsystem) information about a set of datasets. This tool also gives information about the total DASD (Direct Access Storage Device) space used by a user(s)/set of datasets.

This information is required for analysis and design of procedures/ processes with the objective of performance tuning. A few reasons for using this tool are listed below:

1 Classification of datasets based on their storage unit (DASD/ TAPE).

2 Capacity planning for the processes (individual files/total DASD space used).

3 List all datasets catalogued by a user under his user-id along with SMS information.

4 Details like storage unit, space allocated, dataset organization, record length, and record format of given datasets could be obtained in one shot for miscellaneous datasets.

TOOL DETAILS

The report could be generated using either of two options.

The first option is to get a report for all the datasets starting with a particular prefix. For example:

```
C714060.TEST
```

This will create a report for all the datasets starting with C714060.TEST.*. See Figure 1.

The second option is to get a report for all the miscellaneous datasets given by a user. The user needs to create a master dataset before executing the tool. This master dataset must contain the names of all

the miscellaneous datasets. This dataset can have any name. This master dataset will be used as input to the DSINFO tool. For example, sequential file C714060.TEST.MYLIST will contain a list of datasets for which SMS information is required. See Figure 2.

The tool will automatically create the report dataset USERID.TEST.REPORT.DSSIZE, where USERID is the user's mainframe user-id. If the report dataset is already catalogued, the tool will overwrite the contents of the report. (To save the reports generated with different input, copy the dataset USERID.TEST.REPORT.DSSIZE into another dataset.)

The tool provides an option of creating a report with or without calling migrated datasets. The default option is blank (ie no recall of migrated datasets). If you specify 'Y', all the migrated datasets will be recalled and it may take more time to create the report.

Even if the dataset is migrated, its SMS information is obtained, though only partially. Details like dataset storage unit, space, and organization are reported. The information that is not listed is record length and record format. So if record format and length are not important for a particular analysis, having the recall option blank would be a wise decision. This way, the tool would be used optimally.

If the tool is not able to decipher the complete information about the dataset SMS information, it may throw a few messages on the screen. There is no problem with such messages. The tool will continue working fine.

A typical report is shown in Figure 3.

HOW TO USE THE TOOL

Upload the REXX tool DSINFO as a member of a CLIST library or EXEC library and concatenate it with SYSPROC or SYSEXEC respectively. This depends on your mainframe default installations and may vary from one mainframe to another.

Upload panel REPORT1 as a member in USERID.TEST.ISPPLIB. If this library is not available then create this library with a record

format of FB, record length of 80 bytes, and dataset organization of PDS. The USERID is the user's mainframe user-id.

Then go to the ISPF command shell (to use TSO commands). Just type DSINFO and the panel REPORT1 will pop up. Enter your options and press *Enter* to create the report.

Limitations:

1 For incorrect datasets, it will give proper error messages and at times it may throw the user out of the panel. Please correct the input and reuse the tool.

2 For datasets residing on tape, the tool gives partial SMS information. However, it mentions the storage unit as TAPE.

3 For VSAM datasets, the base along with data and index files will be shown in the report. The complete SMS information about VSAM datasets will be indicated against the VSAM base. Information about the data and index part will remain blanks/ nulls. Note that, in such cases, there is no loss of SMS information.

DSINFO

```
/******************************** REXX ********************************/
/*** Purpose: Generate a report giving details of storage unit,    ***/
/***        : space allocated, dataset organization, record length,***/
/***        : and record format for given datasets.                ***/
/*** Input  : Option 1: Get the report for all the datasets starting**/
/***        : with a particular prefix                             ***/
/***        : Option 2: Get the report for all the datasets required**/
/***        : by user. User needs to create a dataset before        **/
/***        : executing the tool. This dataset must contain the    ***/
/***        : names of specific DSN.                               ***/
/*** Output : The report with the above stated details.            ***/
/*Execution : Run the macro in TSO by entering macro name.         ***/
/***                                                               ***/
/*** Author : Yash (Jun 21, 2003) - Longest Day of the Year        ***/
/*********************************************************************/
/***                  Modification Log                             ***/
/***-----------------------------------------------------------------***/
/*********************************************************************/
 panelpds = "'"||userid()||".TEST.ISPPLIB"||"'"
"Ispexec libdef ispplib dataset id ("panelpds")"
```

```
Do forever
   "Ispexec display PANEL(report1) CURSOR(y)"
   IF rc >  4  then do
      exit
   end
   "ispexec vget (y op1 op2 v keypress) profile"
   IF keypress = "" then do
     upper op1 op2 v keypress
     X = MSG('OFF')
 /*   trace ?i */
     validation_done = TRUE
     If validation_done = TRUE then do
       If (y ¬= 1) & ( y ¬= 2) then do
          zedsmsg = "Option Incorrect"
          zedlmsg = "Option must be either 1/ 2"
          "ispexec setmsg msg(isrz001)"
          validation_done = FALSE
       END
     END
     If validation_done = TRUE then do
       If (v ¬= 'Y') & (v ¬= "") then do
           zedsmsg = "Recall Option Incorrect"
           zedlmsg = "Recall Option = Y or Blank"
           "ispexec setmsg msg(isrz001)"
           validation_done = FALSE
       END
     END
     If validation_done = TRUE then do
         IF (y = 1) & (op1 = ' ') then do
               zedsmsg = "Enter DSN name"
               zedlmsg = "Enter the DSN for Option 1"
               "ispexec setmsg msg(isrz001)"
               validation_done = FALSE
           END
     END
     If validation_done = TRUE then do
         IF (y = 2) & (op2 = ' ') then do
               zedsmsg = "Enter DSN name"
               zedlmsg = "Enter the DSN for Option 2"
               "ispexec setmsg msg(isrz001)"
               validation_done = FALSE
           END
     END
     If validation_done = TRUE then do
         IF y = 1 then indsn = op1
         If y = 2 then indsn = op2
 parse var indsn v1 '.' v2 '.' v3 '.' v4 '.' v5 '.' v6 '.' v7 '.' v8
      If (length(v1) > 8) | (length(v2) > 8) | (length(v3) > 8) |,
         (length(v4) > 8) | (length(v5) > 8) | (length(v6) > 8) |,
         (length(v7) > 8) | (length(v8) > 8) | (length(v9) > 8)
```

```
          then  validation_done = FALSE
        If validation_done = FALSE then do
          zedsmsg = "Incorrect DSN"
          zedlmsg = "DSN qualifier has length greater than 8 bytes"
          "ispexec setmsg msg(isrz001)"
        END
      END
      If validation_done = TRUE then do
        Select
          When y = 1 then do
                         call Generate_report_1
                         call Write_the_report
                         end
          When y = 2 then do
                         call Generate_report_2
                         call Write_the_report
                         end
          Otherwise nop
        End /* select */
      END /* validation_done = TRUE */
    End  /* keypress = "" */
End /* Do forever */
Exit 0
Generate_report_1:
/*-----------------------------------------------------------------*/
/* Delete & Create the file temporary dataset storage file         */
/*-----------------------------------------------------------------*/
gatdsn = "'"||userid()||".test.gather.dsn"||"'"
"DELETE "|| gatdsn
IF SYSDSN(gatdsn) <> 'OK' THEN DO
   "ALLOCATE DA("gatdsn") NEW SPACE(30,20) TRACK LRECL(80)
    FILE(file1) RECFM(F,B) BLKSIZE(27920) UNIT(sysda)"
END
/*-----------------------------------------------------------------*/
/* Gather the full dataset name that is catalogued in the system*/
/*-----------------------------------------------------------------*/
"ISPEXEC LMDINIT LISTID(ID1) LEVEL("indsn")"
"ISPEXEC LMDLIST LISTID("ID1") DATASET(DSVAR)"
COUNT = 0
DO WHILE RC = 0
  COUNT = COUNT + 1
  record.COUNT = DSVAR
  "ISPEXEC LMDLIST LISTID("ID1") DATASET(DSVAR)"
END
If COUNT = 0 then do
  zedsmsg = "Incorrect Partial DSN"
  zedlmsg = "Partial Qualifier does not match any dataset"
  "ispexec setmsg msg(isrz001)"
  return
END
```

```
/*-------------------------------------------------------------*/
/* Write all the dataset names in the temporary sequential file */
/*-------------------------------------------------------------*/
"ALLOC FI(file1) DS ("gatdsn")"
"EXECIO * DISKW file1 (STEM record. FINIS"
"FREE FILE(file1)"
/*-------------------------------------------------------------*/
/*   process the datasets stored in a sequential file          */
/*-------------------------------------------------------------*/
call Main_processing
return
Generate_report_2:
/*-------------------------------------------------------------*/
/*   process the datasets supplied by user                     */
/*-------------------------------------------------------------*/
gatdsn = "'"||indsn||"'"
IF SYSDSN(gatdsn) ¬= 'OK' THEN DO
   zedsmsg = "Invalid DSN"
   zedlmsg = "Enter the correct Dataset Name"
   "ispexec setmsg msg(isrzØØ1)"
   exit Ø
End
call Main_processing
return
Main_processing:
/*-------------------------------------------------------------*/
/*   Main logic of the tool                                    */
/*-------------------------------------------------------------*/
repdate = "Date:"|| DATE()
reptime = "Time:"|| TIME('C')
Reptitle1 = 'REPORT GENERATED USING OPTION '|| y
queue left(repdate,2Ø) center(Reptitle1,46) left(reptime,12)
queue left('',8Ø,'-' )
line1A  = "  Dataset Name                                  "
line2A  = "   Unit  " " Space  " "Org" "Lrecl" "Recfm"
line1B  = "----------------------------------------"
line2B  = " --------" " ---- " "--" "----" "----"
line3   = line1A || line2A
line4   = line1B || line2B
queue line3
queue line4
spaceinBYTES = Ø
"ALLOC FI(file2) DS("gatdsn")"
"EXECIO * DISKR file2 (STEM IN. FINIS"
DO I=1 TO IN.Ø
  PARSE VAR IN.I dsn1 " " JUNK
  file = "'"|| dsn1 || "'"
   If v = 'Y' then
     x = listdsi(file recall)
   else
```

```
  x = listdsi(file norecall)
Select
  When sysreason = Ø then nop /* sysreason Ø ==> DASD  */
  When sysreason = 1 then do  /* sysreason 1 ==> Invalid DSN */
    sysunits   = 'INVALID '
    sysalloc   = Ø
    sysblksize = Ø
    sysblkstrk = Ø
    systrkscyl = Ø
    sysdsorg   = '    '
    syslrecl   = '     '
    sysrecfm   = '      '
  End
  When sysreason = 5 then do  /* sysreason 5 ==> Dataset not */
    sysunits   = 'UNCATLOG'    /* catalouged                  */
    sysalloc   = Ø
    sysblksize = Ø
    sysblkstrk = Ø
    systrkscyl = Ø
    sysdsorg   = '    '
    syslrecl   = '     '
    sysrecfm   = '      '
  End
  When sysreason = 8 then do  /* sysreason 8 ==> Tape       */
    sysunits   = 'TAPE    '
    sysalloc   = Ø
    sysblksize = Ø
    sysblkstrk = Ø
    systrkscyl = Ø
    sysdsorg   = '    '
    syslrecl   = '     '
    sysrecfm   = '      '
  End
  When (sysreason = 9) | (sysreason = 12) then do
    /* sysreason 9  ==>  Migrated  */
    /* sysreason 12 ==>  VSAM  */
    tempfile = "'"||userid()||".temp.hilst.dsn"||"'"
    /* HLIST command automatically allocates tempfile */
    "DELETE "|| tempfile
    "HLIST dsname("||"'"dsn1||"'"||") ODS("tempfile")"
    "ALLOC FI(file3) DS("tempfile")"
    "EXECIO * DISKR file3 (STEM ab. FINIS"
    "FREE FILE(file3)"
     migtrk = substr(ab.6,75,6) /* track info is at 75th position */
     migorg = substr(ab.6,96,2) /* Org info is at 96th position */
     migtrk = strip(migtrk,'L','Ø')
     migorg = strip(migorg)
     If migtrk = ' ' then do
        migtrk   = Ø
        migtrk   = Ø
```

```
            sysalloc = 0
         End
         sysunits   = 'TRACK'
         syslrecl   = '     '
         sysrecfm   = '      '
         If (sysalloc = '????') | (sysalloc = ' ') then do
            sysalloc   = 0
            sysblksize = 0
            sysblkstrk = 0
            systrkscyl = 0
            sysdsorg   = '    '
         end
         else do
           sysalloc   = migtrk * 55840   /* 1 trk = 55840 bytes */
           sysdsorg   = migorg
           sysblksize = 1
           sysblkstrk = 1
         End
      End
      When sysreason = 25 then do /* sysreason 25 ==> Unknown storage */
        sysunits = 'Unknown '
        sysalloc   = 0
        sysblksize = 0
        sysblkstrk = 0
        systrkscyl = 0
        sysdsorg   = '    '
        syslrecl   = '     '
        sysrecfm   = '      '
      End
      Otherwise nop
    END /*select*/
        queue left(dsn1,44) left(sysunits,9) left(sysalloc,8),
              left(sysdsorg,4) left(syslrecl,5) left(sysrecfm,6)
        If sysunits = 'BLOCK' then
          spaceinBYTES = spaceinBYTES + sysalloc * sysblksize
        If sysunits = 'TRACK' then do
          bytesPerTrack = sysblksize * sysblkstrk
          spaceinBYTES = spaceinBYTES + sysalloc * bytesPerTrack
        End
        If sysunits = 'CYLINDER' then do
          bytesPerCylinder = sysblksize * sysblkstrk * systrkscyl
          spaceinBYTES = spaceinBYTES + sysalloc * bytesPerCylinder
        End
END
"FREE FILE(file2)"
If spaceinBYTES > 0 then
Total  = spaceinBYTES / (1024 * 1024)
else Total = 0
queue centre('',80,'-')
queue centre('Summary of the space Used',80,'.')
```

```
queue centre('',8Ø,'-')
queue centre('Total space used by above datasets = ',4Ø),
      left(Total,1Ø) left('Megabytes',15)
queue centre('Total files in this report         = ',4Ø) left(IN.Ø,1Ø)
queue centre('',8Ø,'-')
return
Write_the_report:
/*-------------------------------------------------------------*/
/* Write the report                                            */
/*-------------------------------------------------------------*/
 IF queued() > Ø then do
       address tso
       outdsn = userid()||".test.report.dssize"
       IF sysdsn("'"outdsn"'")= 'OK' then
          do
            "delete '"||outdsn ||"'"
          end
  "alloc dd(report) ds('"outdsn"') recfm(f) dsorg(ps),
  lrecl(8Ø) space(4Ø,3Ø) tracks new reu"
  IF rc > Ø then exit 8
    "execio "queued()" diskw report ( finis"
/*IF rc > Ø then exit 8 */
  "Ispexec browse dataset('"outdsn"')"
  "FREE FILE(report)"
 End
```

REPORT1

```
)ATTR
# TYPE(INPUT) INTENS(HIGH) CAPS(ON) COLOR(TURQUOISE) PAD('_') JUST(LEFT)
_ TYPE(INPUT) INTENS(HIGH) CAPS(ON) COLOR(GREEN)     PAD('_')
@ TYPE(TEXT)  INTENS(HIGH) CAPS(ON) COLOR(PINK)
! TYPE(TEXT)  INTENS(HIGH) CAPS(ON) COLOR(YELLOW)             SKIP(ON)
[ TYPE(TEXT)  INTENS(LOW)          COLOR(RED)                SKIP(ON)
{ TYPE(TEXT)  INTENS(HIGH)         COLOR(BLUE)               SKIP(ON)
} TYPE(TEXT)  INTENS(HIGH)         COLOR(GREEN)              SKIP(ON)
^ TYPE(TEXT)  INTENS(LOW)
)BODY
}%Userid :%&ZUSER  @REPORT :: DSN-UNIT-SPACE-ORG-LRECL-RECFM }%Date
:%&ZDATE
@                  ------   ------------------------------
%Command@===>_ZCMD
^
%-------------------------------------------------------------------------
--------
%Option @===>#y^
^
[1.!Report of Space Occupied by Particular Family of Datasets
   }==>#op1                                      [(Enter Partial
Qualifier)
```

```
^
} Enter User Id to see the DASD space occupied
^
[2.!Report of Space Occupied by Distinct Set of Datasets
   }==>#op2                                          [(Enter Dataset
Name)
^
} Do You want to recall the[Migrated Datasets}for creating the reports
  1 and 2 ?}==>#v!(Y/Blank)
  Use this option if you need LRECL & RECFM of Migrated Datasets.
@([CAUTION:@Recalling Migrated Dataset will take more execution time.)
^
%-------------------------------------------------------------------------
--------
{                          PF3 - Exit ; Enter - Process
%-------------------------------------------------------------------------
--------
)INIT
 Vget (keypress) PROFILE
&ZCMD = ' '
)PROC
 &KEYPRESS = .PFKEY
 VER(&y,NB)
 VER(&y,NUM)
 VPUT (y op1 op2 v
       Keypress) PROFILE
)END
```

Yash Pal Samnani
Program Analyst
Infosys Technologies Limited (USA)

DFSMSdss ENQ exit routine

INTRODUCTION

A common problem with shared DASD is managing DFSMSdss full dump with minimal impact on ENQ/RESERVE lockout.

During a standard full dump, DFSMSdss issues an ENQ/RESERVE macro during the entire 'read' of the volume, preventing other systems from accessing the disk for several minutes.

This has a potential impact on shared datasets' activity, which can affect MIM, SLS, RMM, or other system products whose control datasets are shared.

The DFSMSdss enqueue exit routine, ADRUENQ, allows DFSMSdss to enqueue the VTOC for only the read of the VTOC, not the entire read of the volume. In that configuration, the ENQ is held for only a few seconds.

This article explains how we have implemented this exit in our installation.

ADRUENQ INSTALLATION EXIT ROUTINE

To access a volume while it is being dumped, either by another job under the control of a second initiator or by another processor in a shared DASD environment, you can use the ADRUENQ exit routine to enqueue the VTOC only until it is processed.

This exit is called only for physical DUMP, COPY, or PRINT operations.

You can use this exit to prevent DFSMSdss from enqueueing the VTOC for a long period of time. By not enqueueing the VTOC, you can reduce the chances of a deadlock.

However, there is a trade-off – with the reduced chance of a deadlock there is also decreased data integrity.

By default, DFSMSdss enqueues the VTOC for the entire operation of a full or tracks COPY, a full DUMP, or a tracks PRINT.

The sample ADRUENQ routine provided by IBM changes the duration of the ENQ for all DUMP, COPY, and PRINT operations:

```
 ********************************************************
* ADRUENQ USER EXIT.                                    *
*   SETS RETURN CODE TO 4 INDICATING THAT THE VOLUME *
*   WILL ONLY BE ENQUEUED FOR THE DURATION OF THE    *
*   VTOC ACCESS FOR DUMP AND COPY OPERATIONS.        *
 ********************************************************

ADRUENQ  CSECT
ADRUENQ  AMODE 31
```

```
    ADRUENQ  RMODE 24
             STM   14,12,12(13)           SAVE REGS IN PREVIOUS
SAVEAREA
             USING ADRUENQ,15             SET ADDRESSABILITY TO THE
EXIT
             LM    14,12,12(13)           RESTORE OTHER REGISTERS
             LA    15,4                   SET RETURN CODE TO 4
             BR    14                     RETURN
             END
```

This is a very global and basic approach, hence the reason we decided to write our own exit.

First, we wanted to use different default values for DUMP, COPY, and PRINT operations.

We decided to implement the following default ENQ duration:

- DUMP – short ENQ duration
- COPY – long ENQ duration
- PRINT – short ENQ duration.

In certain circumstances, we wanted to be able to override the default ENQ duration of the DFSMSdss operation. This is why we decided to implement the use of a dummy DD statement to override the default setting:

```
//VTOCENQL DD DUMMY              - to issue a LONG ENQ
//VTOCENQS DD DUMMY              - to issue a short ENQ
```

ADRUENQ installation

ADRUENQ is an installation exit routine: it is a replaceable module that modifies DFSMSdss system functions.

You should use SMP/E to link-edit the ADRDSSU module with your own version of ADRUENQ.

```
//ASSEM    EXEC PGM=ASMA90,PARM=('NODECK,OBJECT,XREF(SHORT)')
//SYSLIB   DD  DISP=SHR,DSN=SYS1.MODGEN
//         DD  DISP=SHR,DSN=SYS1.MACLIB
//SYSUT1   DD  DSN=&SYSUT1,SPACE=(1024,(120,120),,,ROUND),UNIT=SYSDA
//SYSPUNCH DD  SYSOUT=*
//SYSPRINT DD  SYSOUT=*
//SYSLIN   DD  DISP=SHR,DSN=ZOSR14.FB80(ADRUENQ)
//SYSIN    DD  *
```

```
*******************************************************
* ADRUENQ USER EXIT.                                  *
*   SETS RETURN CODE TO 4 INDICATING THAT THE VOLUME *
*   WILL ONLY BE ENQUEUED FOR THE DURATION OF THE    *
*   VTOC ACCESS FOR DUMP AND COPY OPERATIONS.        *
*******************************************************
        TITLE 'DF/DSS EXIT, VTOC ENQ/DEQ FOR DFDSS'
ADRUENQ  CSECT
ADRUENQ  AMODE 31
ADRUENQ  RMODE ANY
*
* DESCRIPTION : THIS EXIT IS CALLED BY DFDSS TO DETERMINE WHETHER
* A VTOC ENQ SHOULD BE HELD FOR THE LIFE OF THE COMMAND (FULL
* VOLUME COPY & DUMP, TRACK PRINTGS, AND PHYSICAL
* DATASET DUMP) OR WHETHER THE VTOC ENQ SHOULD BE HELD ONLY
* WHILE THE VTOC IS BEING PROCESSED
*
* RETURN CODES:
*               Ø - HOLD VTOC DURING ENTIRE OPERATION
*               4 - RELEASE VTOC AFTER PROCESSED
*
         STM R14,R12,12(R13)
         LR R12,R15
         USING ADRUENQ,R12
         B START
         DC CL8'ADRUENQ'
         DC C'&SYSDATE'
         DC C'&SYSTIME'
START    DS ØH
         LR R5,R1
         USING ADRUNQB,R5
         GETMAIN RC,LV=DSECTLEN,SP=Ø,LOC=ANY
         LTR R15,R15
         BNZ GETERROR
         ST R1,8(R13)               SAVE FORWARD CHAIN
         ST R13,4(,R1)              SAVE BACKWARD CHAIN
         LR R13,R1
         USING WORKAREA,R13         ADDRESSIBILITY TO WORK AREA
*
         WTO 'XXXXXXXX ADRUENQ EXIT:'
*
         XC FLAG,FLAG
         LA R2,DSSDEQ               GET DEQ DDNAME
         LA R3,DEVINFO              GET RETURN AREA
*
         DEVTYPE (R2),((R3),8)      CHECK FOR DDNAME
         C R15,=F'Ø'                DEQ FLAG SET?
         BNE NODEQ                  NO
         OI FLAG,X'8Ø'              SET FLAG TO SHORT ENQ
NODEQ    DS ØH
```

```
         LA R2,DSSENQ                GET ENQ DDNAME
         LA R3,DEVINFO
         DEVTYPE (R2),((R3),8)       CHECK AGAIN
         C R15,=F'Ø'                 FIND IT?
         BNE NOENQ                   YES, CAUSE ENQ
         OI FLAG,X'4Ø'               SET FLAG TO LONG ENQ
*
NOENQ    DS ØH
*
         TM UNFLG1,UNDUMP            DUMP ?
         BZ NODUMP
*
         TM FLAG,X'4Ø'
         BZ DUMPMSG
         WTO 'XXXXXXXX OVERRIDE DUMP DEFAULT - USING LONG VTOC RESERVE'
         B ENDRCØ
*
DUMPMSG  DS ØH
         WTO 'XXXXXXXX USING DUMP DEFAULT- USING SHORT VTOC RESERVE'
         B ENDRC4
*
NODUMP   DS ØH
         TM UNFLG1,UNCOPY            COPY ?
         BZ NOCOPY
*
         TM FLAG,X'8Ø'
         BZ COPYMSG
         WTO 'XXXXXXXX OVERRIDE COPY DEFAULT - USING SHORT VTOC RESERVE+
               '
         B ENDRC4
*
COPYMSG  DS ØH
         WTO 'XXXXXXXX USING COPY DEFAULT - USING LONG VTOC RESERVE'
         B ENDRCØ
*
NOCOPY   DS ØH
         TM UNFLG1,UNPRINT           PRINT ?
         BZ BADFLAGS
*
         TM FLAG,X'4Ø'
         BZ PRTMSG
         WTO 'XXXXXXXX OVERRIDE PRINT DEFAULT - USING LONG VTOC RESERVE+
               '
         B ENDRCØ
*
PRTMSG DS ØH
         WTO 'XXXXXXXX USING PRINT DEFAULT - USING SHORT VTOC RESERVE'
         B ENDRC4
*
BADFLAGS DS ØH
```

```
         WTO 'XXXXXXXX UNKNOWN FUNCTION - USING LONG VTOC RESERVE'
         B ENDRC4
*
ENDRC4   DS ØH
         LA R6,4
         B EOJ
*
ENDRCØ   DS ØH
         LA R6,Ø
         B EOJ
*
EOJ      DS ØH
         LR R1,R13
         L R13,4(,R13)
         ST R15,16(,R13)
         FREEMAIN  RU,LV=DSECTLEN,SP=Ø,A=(1)
         LR R15,R6
         RETURN (14,12),RC=(15)
GETERROR DS ØH
        WTO 'XXXADRØ4 COULD NOT GETMAIN MEMORY, WILL NOT DO LONG RESERV+
              E'
         RETURN (14,12),RC=4
*
DSSDEQ   DC C'VTOCENQS'
DSSENQ   DC C'VTOCENQL'
*
         LTORG
WORKAREA DSECT
SAVEAREA DS 18F
RC       DS F
DEVINFO  DS 2F
FLAG     DS X
         DS ØD
DSECTLEN EQU *-WORKAREA
         ADRUNQB
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
R1Ø      EQU   1Ø
R11      EQU   11
R12      EQU   12
R13      EQU   13
R14      EQU   14
```

```
R15      EQU   15
/*
//SMP     EXEC PGM=GIMSMP,REGION=4M,
//         PARM='DATE=U,CSI=ZOSR14.GLOBAL.CSI'
//*
//SMPHOLD  DD  DUMMY
//SMPCNTL  DD  *
  SET BDY (GLOBAL).
  REJECT  SELECT(EXIT057) BYPASS(APPLYCHECK) .
  RESETRC .

  RECEIVE SELECT(EXIT057) .


  SET BDY (MVST100).
  APPLY SELECT(EXIT057)
           REDO
           .
/*
//SMPPTFIN DD  *
++USERMOD(EXIT057).
++VER(Z038) FMID(HDZ11G0) PRE(UW88403) .
++MOD(ADRUENQ) DISTLIB(AADRLIB) .
//       DD DISP=SHR,DSN=ZOSR14.FB80(ADRUENQ)
```

ADRUENQ usage

Using default DUMP setting

This first example shows the result of a basic DASD full dump using the default ENQ setting:

```
//STEP1   EXEC PGM=ADRDSSU
//SYSPRINT DD SYSOUT=*
//DASD     DD DISP=SHR,UNIT=SYSALLDA,VOL=SER=RES$01
//BACKUP   DD DUMMY
//SYSIN    DD  *
  DUMP FULL INDD(DASD) OUTDD(BACKUP)
/*
```

Full DUMP SYSOUT using default settings:

```
$HASP373 SXSP001D STARTED - WLM INIT  - SRVCLASS JES_30   - SYS SMVS
IEF403I SXSP001D - STARTED - TIME=10.38.10
TSS7000I SXSP001 Last-Used 18 Aug 03 10:37 System=SMVS Facility=BATCH
TSS7001I Count=56926 Mode=Warn Locktime=None Name=******
XXXXXXXX ADRUENQ EXIT:                                                -
XXXXXXXX USING DUMP DEFAULT- USING SHORT VTOC RESERVE                 -
TSS7000I SXSP001 Last-Used 18 Aug 03 10:38 System=SMVS Facility=BATCH
```

```
TSS7001I Count=56927 Mode=Warn Locktime=None Name******
-                                        --TIMINGS (MINW.)--
-JOBNAME  STEPNAME PROCSTEP    RC   EXCP    CPU    SRB  ELAPS   SERV
-SXSP001D STEP1                00  45713    .06    .02    2.6   708K
IEF404I SXSP001D - ENDED - TIME=10.40.48
-SXSP001D ENDED.  NAME-WILFORD               TOTAL CPU TIME=   .06
TOTAL
$HASP395 SXSP001D ENDED
```

Overriding DUMP setting

In order to issue a long ENQ during the job, you should code a //VTOCENQL DD card to override the default setting:

```
//STEP1   EXEC PGM=ADRDSSU
//SYSPRINT DD SYSOUT=*
//DASD     DD DISP=SHR,UNIT=SYSALLDA,VOL=SER=RES$01
//BACKUP   DD DUMMY
//VTOCENQL DD DUMMY                                - to issue a LONG ENQ
//SYSIN    DD  *
  DUMP FULL INDD(DASD) OUTDD(BACKUP)
/*
```

Full DUMP SYSOUT overriding default setting:

```
$HASP373 SXSP001D STARTED - WLM INIT  - SRVCLASS JES_30   - SYS SMVS
IEF403I SXSP001D - STARTED - TIME=10.38.10
TSS7000I SXSP001 Last-Used 18 Aug 03 10:37 System=SMVS Facility=BATCH
TSS7001I Count=56926 Mode=Warn Locktime=None Name******
XXXXXXXX ADRUENQ EXIT:
XXXXXXXX OVERRIDE DUMP DEFAULT - USING LONG VTOC RESERVE
TSS7000I SXSP001 Last-Used 18 Aug 03 10:38 System=SMVS Facility=BATCH
TSS7001I Count=56927 Mode=Warn Locktime=None Name******
-                                        --TIMINGS (MINW.)--
-JOBNAME  STEPNAME PROCSTEP    RC   EXCP    CPU    SRB  ELAPS   SERV
-SXSP001D STEP1                00  45713    .06    .02    2.6   708K
IEF404I SXSP001D - ENDED - TIME=10.40.48
-SXSP001D ENDED.  NAME-WILFORD               TOTAL CPU TIME=   .06
TOTAL
$HASP395 SXSP001D ENDED
```

BIBLIOGRAPHY

More information about ADRUENQ user exit can be found in *z/OS DFSMS Installation Exits* (SC26-7396).

Patrick Renard
Sytems Programmer (France)

Data conversion

This utility takes a COBOL copybook and converts it into a REXX include file. If you compile REXX programs, the REXX compiler parses the program looking for special INCLUDE instructions and then copies that code into the source before finishing the compile – much like a pre-processor.

The utility was written to perform data conversion. A project leader in the group decided that we REXX users had to use the same names for our variables as the COBOL people. The COBOL people were getting their copybooks from a data-mapping group, but that left us REXXers out in the cold. Initially, some of us started to manually convert COBOL copybooks to a REXX-like format, but with frequent changes in the data-mapping. Well, you can see that it had problems.

So I wrote the attached utility. With it we did the majority of the conversion using REXX rather than COBOL (much to the chagrin of the aforementioned project leader), and we came in a full five months ahead of schedule.

I have recently found another use for it, and once again it's a lifesaver.

COPYBOOK REXX

```
/* ---------------------------- Rexx  --------------------------------
    - Title:  Convert COBOL Copy Books To REXX Include Format          -
     - ================================================================-
      - Program Function:                                              -
      -                                                                -
      -  Converts a COBOL copybook to something REXX can work with.   -
      -                                                                -
       - ------------------------------------------------------------ -
      -      Feel free to modify this program as you see fit.          -
       - ------------------------------------------------------------ -
      -                                                                -
      - Syntax:                                                        -
      -                                                                -
      -          COPYBOOK MEMBER SOURCEPDS DESTPDS                     -
      -                                                                -
      -  Where:                                                        -
```

```
      -          COPYBOOK  ->  This program.                          -
      -          MEMBER    ->  Name of COBOL copybook member to be    -
      -                            converted to REXX.                 -
      -          SOURCEPDS ->  PDS containing COBOL copybook members. -
      -          DESTPDS   ->  PDS that will contain the converted    -
      -                            copybook in REXX format.           -
      -                                                               -
       ---------------------------------------------------------------*/
ConvertCobolCopybooks:
      arg member source dest .
      call Initialize
      call PerformCopyBookConversion
      call Finalize
return
/*  ---------------------------------------------------------------*/
Initialize:
      k = 0; offset = 1; true = 1; false = 0 ; fieldlen = 0
      level.0 = ''; variable.0 = ''; offset.0 = ''; stackptr = 0
/* Initialize stacks */
      levelq.0 = ''; variableq.0 = ''; offsetq.0 = ''; queuetop = 0
/* Initializq queues */
      lastlevel = 0
/* Just what it says */
      i = 0
return
/*  ---------------------------------------------------------------*/
PerformCopyBookConversion:
      call ReadCopybook
      do while (i < copybook.0)
         i = i + 1
         copybook = PreProcessCopybook(copybook.i)
         parse var copybook level variable pic field remainder
         if ((level < lastlevel) | (pic = '' & level = LevelStackTop()))
then
              do while (level <= LevelStackTop())
                 call PopStacks
              end
         lastlevel = level
         select
            when (left(copybook,1) = "*") then
                nop
            when (pic = 'REDEFINES') then
                do
                   offset = RedefinesProcessing(field)
                   parse var copybook level variable redword redvar
pic field remainder
                  if (field ¬= '') then
                     call ProcessCobolLevel
                  else
                     call PushStacks
```

```
                end
            when level = 88 then
                call Level88Processing
            when pic = '' then
                call PushStacks
            otherwise
                call ProcessCobolLevel
         end
      end
      do while (stackptr ¬= Ø)
         call PopStacks
      end
return
/* ---------------------------------------------------------------------*/
PreProcessCopybook:
         arg copybook
         if (left(copybook,1) ¬= " ") | (left(copybook,1) ¬= "*") then
            parse var copybook 1 junk 7 copybook 73 .
         copybook = strip(copybook)
         select
            when (copybook = '') then
               copybook = '*'
            when (left(copybook,1) ¬= '*') then
               copybook = BuildCopybookLine(copybook)
            otherwise nop
         end
return copybook;
/* ---------------------------------------------------------------------*/
BuildCopyBookLine:
      arg line
      line = strip(line)
      do while ((substr(line,length(line),1) ¬= '.') & (left(line,1)
¬= "*") & (i < copybook.Ø))
         i = i+1; line =line' 'strip(copybook.i)
      end
      line = substr(line,1,length(line)-1) /* gets rid of period */
return line
/* ---------------------------------------------------------------------*/
LevelStackTop:
return level.stackptr
/* ---------------------------------------------------------------------*/
VariableStackTop:
return variable.stackptr
/* ---------------------------------------------------------------------*/
PushStacks:
     stackptr = stackptr + 1
     variable.stackptr = variable
     level.stackptr = level
     offset.stackptr = offset
return
```

```
/* ---------------------------------------------------------------*/
PopStacks:
     newoffset = offset.stackptr
     newlevel = level.stackptr
     newvariable = translate(variable.stackptr,'_','-')
     stackptr = stackptr - 1
     k = k + 1
     outrec.k = "  parse var record" left(newoffset,6)
left(newvariable, 4Ø) offset
     x = Enqueue(newlevel,newvariable,newoffset)
return
/* ---------------------------------------------------------------*/
Level88Processing:
     variable = translate(variable,'_','-')
     field = TranslateField(field)
     k = k + 1
     outrec.k = "     /* 88-Level */ "||variable||" =
("||lastVariable||"="||field||")";
     if (strip(remainder) ¬= "") then
       do
          string = (pos("'",remainder)¬=Ø);
          if (string) then
             do while (remainder ¬= "")
                parse var remainder "'"val"'" remainder
                outrec.k = outrec.k||" |
("||lastVariable||"='"||val||"')";
             end
          else
             do while (remainder ¬= "")
                parse var remainder val remainder
                outrec.k = outrec.k||" |
("||lastVariable||"="||val||")";
             end
       end
return
/* ---------------------------------------------------------------*/
TranslateField: procedure expose fieldlen
     arg field
     select
        when (field = 'SPACES') | (field = 'SPACE') then
             field = "'"copies(" ",fieldlen)"'"
        when (field = 'ZERO') | (field = 'ZEROS') then
            field = Ø
        when (field = 'LOW-VALUES') then
            field = copies('ØØ'x,fieldlen)
        when (field = 'HIGH-VALUES') then
            field = copies('FF'x,fieldlen)
        otherwise nop
     end
return field
```

```
/* ------------------------------------------------------------------*/
ProcessCobolLevel:
      value = ''
      computational = (pos("COMP-3",remainder) ¬= Ø)
      binary = (pos("COMP ",remainder) ¬= Ø)
      if pos("VALUE",remainder) ¬= Ø then
         parse var remainder "VALUE" value
      fieldlen = DetermineFieldLength(field)
      newoffset = fieldlen + offset
      k = k + 1
      variable = translate(variable,'_','-')
      lastvariable = variable   /* storage bucket in case of 88 levels*/
      outrec.k = "  parse var record" left(offset,6) left(variable,4Ø)
newoffset
      x = Enqueue(level,variable,offset)
      offset = newoffset
      if value ¬= '' then
         do
            k = k + 1
            outrec.k = variable" = "translateField(value);
         end
return
/* ------------------------------------------------------------------*/
RedefinesProcessing: procedure expose variableq. offsetq. queuetop
     arg redefinesvariable
     queueptr = 1
     redefinesvariable = translate(redefinesvariable,'_','-')
     do while redefinesvariable ¬= variableq.queueptr & queueptr <
queuetop
        queueptr = queueptr + 1
     end
return offsetq.queueptr
/* ------------------------------------------------------------------*/
Enqueue: procedure expose levelq. variableq. offsetq. queuetop
     parse arg level,variable,offset
     queuetop = queuetop + 1
     levelq.queuetop = level
     variableq.queuetop = variable
     offsetq.queuetop = offset
return rc
/* ------------------------------------------------------------------*/
DetermineFieldLength: procedure expose computational binary
      arg picmap
      parse var picmap decimal'V'fraction
      fieldlen = Ø
      if decimal ¬= '' then
         do
             parse var decimal pre'('len')'
             if len = '' then
                do
                   if substr(pre,1,1) = 'S' then
```

```
                  len = length(pre) - 1
               else
                  len = length(pre)
            end
          fieldlen = fieldlen + len
      end
    if fraction ¬= '' then
      do
          parse var fraction pre'('len')'
          if len = '' then
             len = length(pre)
          fieldlen = fieldlen + len
      end
    select
      when computational then
         fieldlen = trunc((fieldlen + 1)/2)
      when binary then
         select
            when fieldlen <= 4 then fieldlen = 2
            when fieldlen <= 9 then fieldlen = 4
            when fieldlen <= 16 then fieldlen = 6
            otherwise fieldlen = 8
         end
      otherwise nop
    end
return fieldlen
/* ---------------------------------------------------------------------*/
ReadCopybook:
    source = source'('member')';
    if sysdsn("'"source"'") ¬= "OK" then
      exit 8
    else
      x = ReadDataset(source,"COPYBOOK")
return
/* ---------------------------------------------------------------------*/
ReadDataset:
    parse arg file,stemvar .
    address tso "alloc dd(INDD) da('"file"') shr"
    "execio * diskr INDD (stem "stemvar". finis"
    address tso "free dd(INDD)"
return rc
/* ---------------------------------------------------------------------*/
Finalize:
   convertedcopybook = dest"("member")"
   if sysdsn("'"dest"'") ¬= 'OK' then
      do
       address tso "alloc dd(TEMPDD) da('"dest"') new reuse tr dir(2Ø)",
                 "sp(3Ø 3Ø) lrecl(255) recfm(V B) dsorg(PO) blksize(Ø)"
         address tso "free dd(TEMPDD)"
      end
   address tso "alloc dd(OUTDD) da('"convertedcopybook"') shr"
```

```
     "execio "k" diskw OUTDD (stem outrec. finis"
     address tso "free dd(OUTDD)"
return
/*  -*/
```

SAMPLE COBOL COPYBOOK

This is an example of a COBOL copybook:

```
*****************************************************************
*  CABT0025 - CUI JURISDICTION TABLE RECORD *
*              GTE HISTORY SECTION                    TCID *
*   DATE     INTIAL     RVL      DESCRIPTION          DPSR *
*  07/01/87   KAN     R1V01L01   CABS ENHANCEMENT *
*  12/28/88   SAC     R2V01L01   DATE SENSITIVITY   CABD0070 *
*  12/28/88   SAC     R2V01L01   FGB MPB INDICATOR  CABD0119 *
*  06/05/90   REW     R2V01L01   AT&T 0 EMR1101 CABD0273 *
*  04/28/91   SAC     R2V06L01   DB800 INDICATORS   CABD0306 *
*  04/28/91   SAC     R2V06L01   CELL NATL THRESHLD CABD0307 *
*  03/12/93   LB      R1V24L01   ADD VALUE OF 'L'   CBSD0838 *
*                                TO TRAFFIC-TYPE- *
*                                BILLABLE-INDICATOR *
*****************************************************************
   03  T0025-TABLE-IDENT               PIC X(05).
       88  T0025-TABLE-ID                VALUE 'T0025'.
   03  T0025-TABLE-DATA.
     05  T0025-KEY.
         10  T0025-PARTIAL-KEY.
             15  T0025-FROM-NPA-NXX.
                 20 T0025-FROM-NPA         PIC X(03).
                 20 T0025-FROM-NXX         PIC X(03).
             15  T0025-TO-NPA              PIC X(03).
         10  T0025-EFFECTIVE-START         PIC 9(6).
         10  T0025-EFFECTIVE-END           PIC 9(6).
     05  T0025-FUNCTION.
         10  T0025-TABLE-LEVEL         PIC X(1).
         10  T0025-JURISDICTION        PIC 9.
         10  T0025-STATE-CODE          PIC X(02).
         10  T0025-BILLING-LOCATION-CODE. 00002400
             15  T0025-OPERATING-GROUP PIC X(01).
             15  T0025-CONSOL-COMPANY  PIC X(01).
             15  T0025-COMPANY         PIC X(01).
             15  T0025-AREA-DIV        PIC X(01).
             15  T0025-DIV-DIST        PIC X(02).
             15  T0025-PLANT-CODE      PIC X(04).
         10  T0025-BAN-STATE-CODE      PIC X(01).
         10  T0025-ORIG-MMU            PIC X(01).
         10  T0025-ORIG-RATE-CNTR      PIC 9(03).
         10  T0025-OLATA-CODE          PIC 9(03).
         10  T0025-BILLING-RAO         PIC X(03).
```

```
10  T0025-ASSOC-BELL-RAO      PIC X(03).
10  T0025-RCC-THRESHOLD-SEC   PIC 9(02).
10  T0025-IND-EQUAL-ACCESS    PIC X(01).
    88  T0025-EQUAL-ACCESS        VALUE 'Y'.
    88  T0025-NON-EQUAL-ACCESS    VALUE 'N'.
10  T0025-FGB-MPB-IND         PIC X(01).
10  T0025-FGCD-MPB-IND        PIC X(01).
10  T0025-CONV-OPH-INTRA      PIC S9V9(02) COMP-3.
10  T0025-CONV-OPH-INTER      PIC S9V9(02) COMP-3.
10  T0025-CONV-DIAL           PIC S9V9(02) COMP-3.
10  T0025-RECORD-POINT-DDD    PIC S9(06)  COMP-3.
10  T0025-RECORD-POINT-OPH    PIC S9(06)  COMP-3.
10  T0025-RECORD-POINT-TERM   PIC S9(06)  COMP-3.
10  T0025-ATT-SOURCE-OF-DATA.
    15  T0025-ATT-OPH-ZERO-PLUS      PIC X(01).
        88  T0025-ATT-TYPE-1-TOLL      VALUE 'T' 'L' 'C'
        88  T0025-ATT-TYPE-1-UMS       VALUE 'U'.
        88  T0025-ATT-TYPE-1-EMR       VALUE 'D'.
    15  T0025-ATT-OPH-ZERO-MINUS     PIC X(01).
        88  T0025-ATT-TYPE-2-TOLL      VALUE 'T' 'L' 'C'
        88  T0025-ATT-TYPE-2-UMS       VALUE 'U'.
        88  T0025-ATT-TYPE-2-EMR       VALUE 'D'.
    15  T0025-ATT-MTS-ORIG           PIC X(01).
        88  T0025-ATT-TYPE-3-TOLL      VALUE 'T' 'L'.
        88  T0025-ATT-TYPE-3-UMS       VALUE 'U'.
    15  T0025-ATT-700-ORIG           PIC X(01).
        88  T0025-ATT-TYPE-4-TOLL      VALUE 'T' 'L'.
        88  T0025-ATT-TYPE-4-UMS       VALUE 'U'.
    15  T0025-ATT-800-ORIG           PIC X(01).
        88  T0025-ATT-TYPE-5-TOLL      VALUE 'T' 'L'.
        88  T0025-ATT-TYPE-5-UMS       VALUE 'U'.
    15  T0025-ATT-DB800-ORIG         PIC X(01).
        88  T0025-ATT-DB800-TOLL       VALUE 'T' 'L'.
        88  T0025-ATT-DB800-UMS        VALUE 'U'.
        88  T0025-ATT-DB800-EMR        VALUE 'E'.
    15  T0025-ATT-900-ORIG           PIC X(01).
        88  T0025-ATT-TYPE-6-TOLL      VALUE 'T' 'L'.
        88  T0025-ATT-TYPE-6-UMS       VALUE 'U'.
    15  T0025-ATT-MTS-TERM           PIC X(01).
        88  T0025-ATT-TYPE-7-UMS       VALUE 'U'.
10  T0025-NONATT-SOURCE-OF-DATA.
    15  T0025-NONATT-OPH-ZERO-PLUS   PIC X(01).
        88  T0025-NONATT-TYPE-1-TOLL   VALUE 'T' 'L'.
        88  T0025-NONATT-TYPE-1-UMS    VALUE 'U'.
    15  T0025-NONATT-OPH-ZERO-MINUS  PIC X(01).
        88  T0025-NONATT-TYPE-2-TOLL   VALUE 'T' 'L'.
        88  T0025-NONATT-TYPE-2-UMS    VALUE 'U'.
    15  T0025-NONATT-MTS-ORIG        PIC X(01).
        88  T0025-NONATT-TYPE-3-UMS    VALUE 'U'.
    15  T0025-NONATT-700-ORIG        PIC X(01).
        88  T0025-NONATT-TYPE-4-UMS    VALUE 'U'.
```

```
        15  T0025-NONATT-800-ORIG        PIC X(01).
            88  T0025-NONATT-TYPE-5-TOLL   VALUE 'T' 'L'.
            88  T0025-NONATT-TYPE-5-UMS    VALUE 'U'.
        15  T0025-NONATT-DB800-ORIG      PIC X(01).
            88  T0025-NONATT-DB800-TOLL    VALUE 'T' 'L'.
            88  T0025-NONATT-DB800-UMS     VALUE 'U'.
            88  T0025-NONATT-DB800-EMR     VALUE 'E'.
        15  T0025-NONATT-900-ORIG        PIC X(01).
            88  T0025-NONATT-TYPE-6-TOLL   VALUE 'T' 'L'.
            88  T0025-NONATT-TYPE-6-UMS    VALUE 'U'.
        15  T0025-NONATT-MTS-TERM        PIC X(01).
            88  T0025-NONATT-TYPE-7-UMS    VALUE 'U'.
    10  T0025-CIC000-DB800-ORIG        PIC X(01).
        88  T0025-CIC000-DB800-TOLL    VALUE 'T' 'L'.
        88  T0025-CIC000-DB800-UMS     VALUE 'U'.
        88  T0025-CIC000-DB800-EMR     VALUE 'E'.
```

SAMPLE COPYBOOK REXX

Here is the same sample after it has been run through the utility and converted:

```
parse var record 1      T0025_TABLE_IDENT                       6
   /* 88-Level */ T0025_TABLE_ID = (T0025_TABLE_IDENT='T0025')
parse var record 6      T0025_FROM_NPA                          9
parse var record 9      T0025_FROM_NXX                          12
parse var record 6      T0025_FROM_NPA_NXX                      12
parse var record 12     T0025_TO_NPA                            15
parse var record 6      T0025_PARTIAL_KEY                       15
parse var record 15     T0025_EFFECTIVE_START                   21
parse var record 21     T0025_EFFECTIVE_END                     27
parse var record 6      T0025_KEY                               27
parse var record 27     T0025_TABLE_LEVEL                       28
parse var record 28     T0025_JURISDICTION                      29
parse var record 29     T0025_STATE_CODE                        31
parse var record 31     T0025_OPERATING_GROUP                   32
parse var record 32     T0025_CONSOL_COMPANY                    33
parse var record 33     T0025_COMPANY                           34
parse var record 34     T0025_AREA_DIV                          35
parse var record 35     T0025_DIV_DIST                          37
parse var record 37     T0025_PLANT_CODE                        41
parse var record 31     T0025_BILLING_LOCATION_CODE             41
parse var record 41     T0025_BAN_STATE_CODE                    42
parse var record 42     T0025_ORIG_MMU                          43
parse var record 43     T0025_ORIG_RATE_CNTR                    46
parse var record 46     T0025_OLATA_CODE                        49
parse var record 49     T0025_BILLING_RAO                       52
parse var record 52     T0025_ASSOC_BELL_RAO                    55
parse var record 55     T0025_RCC_THRESHOLD_SEC                 57
parse var record 57     T0025_IND_EQUAL_ACCESS                  58
```

```
      /* 88-Level */ T0025_EQUAL_ACCESS = (T0025_IND_EQUAL_ACCESS='Y')
      /* 88-Level */ T0025_NON_EQUAL_ACCESS =
(T0025_IND_EQUAL_ACCESS='N')
   parse var record 58     T0025_FGB_MPB_IND                        59
   parse var record 59     T0025_FGCD_MPB_IND                       60
   parse var record 60     T0025_CONV_OPH_INTRA                     62
   parse var record 62     T0025_CONV_OPH_INTER                     64
   parse var record 64     T0025_CONV_DIAL                          66
   parse var record 66     T0025_RECORD_POINT_DDD                   69
   parse var record 69     T0025_RECORD_POINT_OPH                   72
   parse var record 72     T0025_RECORD_POINT_TERM                  75
   parse var record 75     T0025_ATT_OPH_ZERO_PLUS                  76
      /* 88-Level */ T0025_ATT_TYPE_1_TOLL =
(T0025_ATT_OPH_ZERO_PLUS='T') | (T0025_ATT_OPH_ZERO_PLUS='L') |
(T0025_ATT_OPH_ZERO_PLUS='C')
      /* 88-Level */ T0025_ATT_TYPE_1_UMS =
(T0025_ATT_OPH_ZERO_PLUS='U')
      /* 88-Level */ T0025_ATT_TYPE_1_EMR =
(T0025_ATT_OPH_ZERO_PLUS='D')
   parse var record 76     T0025_ATT_OPH_ZERO_MINUS                 77
      /* 88-Level */ T0025_ATT_TYPE_2_TOLL =
(T0025_ATT_OPH_ZERO_MINUS='T') | (T0025_ATT_OPH_ZERO_MINUS='L') |
(T0025_ATT_OPH_ZERO_MINUS='C')
      /* 88-Level */ T0025_ATT_TYPE_2_UMS =
(T0025_ATT_OPH_ZERO_MINUS='U')
      /* 88-Level */ T0025_ATT_TYPE_2_EMR =
(T0025_ATT_OPH_ZERO_MINUS='D')
   parse var record 77     T0025_ATT_MTS_ORIG                       78
      /* 88-Level */ T0025_ATT_TYPE_3_TOLL = (T0025_ATT_MTS_ORIG='T')
| (T0025_ATT_MTS_ORIG='L')
      /* 88-Level */ T0025_ATT_TYPE_3_UMS = (T0025_ATT_MTS_ORIG='U')
   parse var record 78     T0025_ATT_700_ORIG                       79
      /* 88-Level */ T0025_ATT_TYPE_4_TOLL = (T0025_ATT_700_ORIG='T')
| (T0025_ATT_700_ORIG='L')
      /* 88-Level */ T0025_ATT_TYPE_4_UMS = (T0025_ATT_700_ORIG='U')
   parse var record 79     T0025_ATT_800_ORIG                       80
      /* 88-Level */ T0025_ATT_TYPE_5_TOLL = (T0025_ATT_800_ORIG='T')
| (T0025_ATT_800_ORIG='L')
      /* 88-Level */ T0025_ATT_TYPE_5_UMS = (T0025_ATT_800_ORIG='U')
   parse var record 80     T0025_ATT_DB800_ORIG                     81
      /* 88-Level */ T0025_ATT_DB800_TOLL = (T0025_ATT_DB800_ORIG='T')
| (T0025_ATT_DB800_ORIG='L')
      /* 88-Level */ T0025_ATT_DB800_UMS = (T0025_ATT_DB800_ORIG='U')
      /* 88-Level */ T0025_ATT_DB800_EMR = (T0025_ATT_DB800_ORIG='E')
   parse var record 81     T0025_ATT_900_ORIG                       82
      /* 88-Level */ T0025_ATT_TYPE_6_TOLL = (T0025_ATT_900_ORIG='T')
| (T0025_ATT_900_ORIG='L')
      /* 88-Level */ T0025_ATT_TYPE_6_UMS = (T0025_ATT_900_ORIG='U')
   parse var record 82     T0025_ATT_MTS_TERM                       83
      /* 88-Level */ T0025_ATT_TYPE_7_UMS = (T0025_ATT_MTS_TERM='U')
   parse var record 75     T0025_ATT_SOURCE_OF_DATA                 83
```

```
   parse var record 83     T0025_NONATT_OPH_ZERO_PLUS                84
      /* 88-Level */ T0025_NONATT_TYPE_1_TOLL =
(T0025_NONATT_OPH_ZERO_PLUS='T') | (T0025_NONATT_OPH_ZERO_PLUS='L')
      /* 88-Level */ T0025_NONATT_TYPE_1_UMS =
(T0025_NONATT_OPH_ZERO_PLUS='U')
   parse var record 84     T0025_NONATT_OPH_ZERO_MINUS               85
      /* 88-Level */ T0025_NONATT_TYPE_2_TOLL =
(T0025_NONATT_OPH_ZERO_MINUS='T') | (T0025_NONATT_OPH_ZERO_MINUS='L')
      /* 88-Level */ T0025_NONATT_TYPE_2_UMS =
(T0025_NONATT_OPH_ZERO_MINUS='U')
   parse var record 85     T0025_NONATT_MTS_ORIG                     86
      /* 88-Level */ T0025_NONATT_TYPE_3_UMS =
(T0025_NONATT_MTS_ORIG='U')
   parse var record 86     T0025_NONATT_700_ORIG                     87
      /* 88-Level */ T0025_NONATT_TYPE_4_UMS =
(T0025_NONATT_700_ORIG='U')
   parse var record 87     T0025_NONATT_800_ORIG                     88
      /* 88-Level */ T0025_NONATT_TYPE_5_TOLL =
(T0025_NONATT_800_ORIG='T') | (T0025_NONATT_800_ORIG='L')
      /* 88-Level */ T0025_NONATT_TYPE_5_UMS =
(T0025_NONATT_800_ORIG='U')
   parse var record 88     T0025_NONATT_DB800_ORIG                   89
      /* 88-Level */ T0025_NONATT_DB800_TOLL =
(T0025_NONATT_DB800_ORIG='T') | (T0025_NONATT_DB800_ORIG='L')
      /* 88-Level */ T0025_NONATT_DB800_UMS =
(T0025_NONATT_DB800_ORIG='U')
      /* 88-Level */ T0025_NONATT_DB800_EMR =
(T0025_NONATT_DB800_ORIG='E')
   parse var record 89     T0025_NONATT_900_ORIG                     90
      /* 88-Level */ T0025_NONATT_TYPE_6_TOLL =
(T0025_NONATT_900_ORIG='T') | (T0025_NONATT_900_ORIG='L')
      /* 88-Level */ T0025_NONATT_TYPE_6_UMS =
(T0025_NONATT_900_ORIG='U')
   parse var record 90     T0025_NONATT_MTS_TERM                     91
      /* 88-Level */ T0025_NONATT_TYPE_7_UMS =
(T0025_NONATT_MTS_TERM='U')
   parse var record 83     T0025_NONATT_SOURCE_OF_DATA               91
   parse var record 91     T0025_CIC000_DB800_ORIG                   92
      /* 88-Level */ T0025_CIC000_DB800_TOLL =
(T0025_CIC000_DB800_ORIG='T') | (T0025_CIC000_DB800_ORIG='L')
      /* 88-Level */ T0025_CIC000_DB800_UMS =
(T0025_CIC000_DB800_ORIG='U')
      /* 88-Level */ T0025_CIC000_DB800_EMR =
(T0025_CIC000_DB800_ORIG='E')
   parse var record 27     T0025_FUNCTION                            92
   parse var record 6      T0025_TABLE_DATA                          92
```

Rick Myers
Technical Trainer
Verizon Communications (USA)

Some useful ISPF utilities

The following ISPF utilities are provided below:

- VIEWHELP – allows users to view the TSO HELP output in a dataset.
- VIEWDD – a routine to view a dataset given the DDname. It is used by VIEWHELP.
- MEMFIND – allows users to search multiple datasets for a particular member.
- MEMCHK – a routine that can be used to check whether a specific member is present in a dataset. It can be invoked directly as a command or a routine. Used by MEMFIND.
- MEMDISP – a command, when invoked with the dataset name as a parameter displays the members list panel.

VIEWHELP

VIEWHELP allow users to view the TSO HELP output in a dataset, which provides the following benefits:

- Allows searching for strings.
- Allows scrolling forward or backward.
- The information, if required, can be saved in a dataset.

Note that **TSO VIEWHELP <hlptopic>** is to be invoked within an ISPF environment, eg **TSO VIEWHELP ALLOC** or **TSO VIEWHELP LISTCAT**.

```
/*************************REXX*****************************************\
*  REXX Program - VIEWHELP                                             *
*  REXX Program that shows the TSO HELP information in a VIEW PANEL    *
*  Used for allowing search and scrolling of HELP information.         *
\*********************************************************************/
arg hlptopic parm
```

```
y = outtrap("hlp.",'*',"concat")
"help "hlptopic parm
if (POS('HELP NOT AVAILABLE',hlp.1) ¬= Ø) then
do
   err_msg_pos = pos('ENTER HELP',hlp.2)
   err_msg_pos = err_msg_pos + 5
   hlp.2 = insert('VIEW',hlp.2,err_msg_pos)
end
x = outtrap("off")
address ispexec "control errors return"
"alloc dd($hlpfl$) unit(vio) lrecl(8Ø) blksize(312Ø) dsorg(ps)",
     "space(1,1) track new reu"
if rc¬=Ø then
do
   if rc=12 then
   do
      say ' '
      say 'YOU ARE ALREADY IN VIEWHELP!!!!'
      say 'EXIT FROM THIS PANEL TO INVOKE VIEWHELP AGAIN'
   end
   exit
end
"execio * diskw  $hlpfl$ (stem hlp. finis"
if rc=Ø then
   address tso     "%viewdd $hlpfl$ "
"free dd($hlpfl$)"
address ispexec "control errors cancel"
exit
```

VIEWDD

The VIEWDD REXX routine can be used to view a dataset given its DDname. It is invoked by VIEWHELP. A view can be modified to browse or edit, depending on the requirement.

```
/*************************REXX*************************************\
*  REXX Routine - VIEWDD                                          *
*  Used to VIEW a dataset given the DDNAME                        *
\******************************************************************/
arg ddn
address ispexec "lminit dataid("did") ddname(&ddn) enq(shr)"
if rc ¬= Ø Then
Do
   say 'LMINIT - Failed with RC ' || rc
   exit
end
/* If required, change the following to Browse or Edit */
```

```
address ispexec "view dataid("did")"
if rc ¬= Ø Then
Do
   say 'VIEW - Failed with RC ' || rc
end
address ispexec "lmfree dataid("did")"
return
```

MEMFIND

The MEMFIND command can be invoked to get a list of datasets having a particular member. It takes the dataset pattern with wild characters – similar to option 3.4 – and displays the list of datasets having the specific member.

The routine can be modified to return the list of datasets matching the pattern along with the required dataset attributes, or to filter the list of datasets based on their attributes – eg list of migrated datasets.

```
/*************************REXX*****************************************\
*  REXX Routine - MEMFIND                                              *
*  INPUT:                                                              *
*           Dataset Pattern - with wild characters                     *
*           memname- Member to be searched for                         *
*  RETURNS:                                                            *
*           List of datasets containing the member                     *
*  Invokes MEMCHK routine to find if a member is present in a dataset*
\**********************************************************************/
arg dspattern memname
address ispexec
if dspattern = "" | memname = "" | dspattern = "?" then
do
   say "Command syntax is MEMFIND <Dataset Pattern> <Member to find>"
   exit
end
/* Get the list of datasets matching the given pattern */
"lmdinit listid("lstid") level("dspattern")"
if rc ¬= Ø Then
Do
   say 'LMDINIT - Failed with RC ' || rc
   exit
end
"lmdlist listid("lstid") option(LIST) dataset(dsvar) STATS(YES)"
if rc = 4 Then
Do
   say 'No DATASET matching this Pattern ' || dspattern
```

```
   exit
end
say 'Dataset Pattern is ' || dspattern
mcnt = Ø
do while rc=Ø
   /* check ONLY if the DATASET is not migrated and a PDS      */
   if ZDLMIGR = 'NO' & ZDLDSORG = 'PO' then
   do
      address tso "%memchk " dsvar memname "NODISP"
      if rc = Ø then
      do
         say 'Dataset ' || dsvar
         mcnt = mcnt + 1
      end
   end
   /* Get the next Dataset matching the pattern               */
   "lmdlist listid("lstid") option(LIST) dataset(dsvar) STATS(YES)"
end
if rc > 8 Then
do
   say 'LMDLIST - Failed with RC ' || rc
exit
end
if mcnt = Ø then
   say 'Member ' || memname || ' NOT FOUND in qualifying DATASETS '
else
   say 'Member ' || memname || ' is FOUND in ' || mcnt || ' Datasets'

say 'NOTE: The Migrated datasets are not considered '
"lmdlist listid("lstid")"
exit
```

MEMCHK

The MEMCHK command can be used to check whether a specific member is present in a dataset. It can be invoked directly as a command or as a routine.

```
/*************************REXX*****************************************\
*  REXX Routine - MEMCHK                                               *
*  Checks whether a specific member is found in the dataset            *
*  Can be invoked as a TSO COMMAND or as a routine from programs       *
*  INPUT:                                                              *
*           dsname - fully-qualified dataset name                      *
*           memname- member to be searched for                         *
*           dispopt- "NODISP" value to be sent to avoid displays       *
*  RETURNS:                                                            *
```

```
*           Ø if the member is FOUND in the dataset                  *
*           1 if the member is NOT FOUND in the dataset              *
\*************************************************************************/
arg dsname memname dispopt
address ispexec
if dsname = "" | memname = "" | dsname = "?" then
do
   say "Command syntax is MEMCHK <Dataset> <Member to find>"
   exit
end
retval = 1  /* Set the RETURN CODE to NOT FOUND as Default value     */
address ispexec
"lminit dataid("did") dataset('&dsname') enq(shr) org(orgds)"
if rc ¬= Ø Then
Do
   say 'LMINIT - Failed with RC ' || rc || ' for dataset ' || dsname
   exit
end
/* Check to ensure that the Dataset is a PDS */
if orgds <> "PO" then
do
   "lmfree dataid("did")"
   exit
end
"lmopen dataid("did") option(input)"
if rc ¬= Ø Then
Do
   say 'LMOPEN - Failed with RC ' || rc || ' for dataset ' || dsname
   exit
end
"lmmfind dataid("did") member(&memname)"
if rc > 8 Then
Do
   say 'LMMFIND - Failed with RC ' || rc
   exit
end
/* set the return value to Ø, if member is FOUND */
if rc = Ø Then
   retval = Ø
/* Display if the dataset if found or not - ONLY if NODISP is not set */
if dispopt <> 'NODISP' then
do
   if retval = Ø then
      say 'Member ' || memname || ' FOUND in ' || dsname
   else
      say 'Member ' || memname || ' NOT FOUND in ' || dsname
end
address ispexec "lmfree dataid("did")"
exit retval
```

MEMDISP

The MEMDISP command can be invoked with the dataset name as a parameter to display the members list panel.

The dataset name can be given with or without quotes, depending on whether you want the prefix to be included or not.

```
/*************************REXX*************************************\
*  REXX Routine - MEMDISP                                         *
*  Directly displays the MEMBER LIST of a given DATASET           *
*  Dataset name to be passed as command line argument             *
*  Dataset name can be with or without Quotes depending on the reqt. *
\******************************************************************/
arg dsname
address ispexec
if dsname = "" Then
Do
   say 'Enter the Dataset Name as an Argument '
   exit
end
"lminit dataid("did") dataset(&dsname) enq(shr) org(orgds)"
if rc ¬= Ø Then
Do
   say 'Invalid Dataset '
   say 'LMINIT - Failed with RC ' || rc
   exit
end
if orgds <> "PO" then
do
   say 'Dataset ' || dsname || ' is not a PDS'
   "lmfree dataid("did")"
   exit
end
address ispexec "memlist dataid("did") member(*)"
if rc ¬= Ø Then
Do
   say 'MEMLIST - Failed with RC ' || rc
end

address ispexec "lmfree dataid("did")"
return
```

Sasirekha Cota
Tata Consultancy Services (India)

Monitoring HFS performance

This article will focus on monitoring the performance of HFS and is a sequel to a previous article (see 'Monitoring USS performance from z/OS – an introduction', *MVS Update*, issues 213 and 214, June and July 2004). The primary focus is on understanding HFS performance metrics as well as on monitoring and managing critical HFS file systems. The sample technique for collecting and analysing HFS performance data will be demonstrated. Tuning recommendations will be briefly discussed too.

INTRODUCTION

Starting with earlier releases, but primarily in OS/390 R5, many of the base MVS components began using OS/390 Unix System Services (USS) and, therefore, Unix file services as supported by IBM's Hierarchical File System (HFS). These components and facilities include TCP/IP, Notes Domino, WebServer, ERP applications, and many others. As already known, you will not be able to avoid HFS any longer because it has now become an integral part of OS/390. It is a fact that each new release of DFSMS continues to build on the previous version to provide enhanced storage management, data access, and device support. For example, in OS/390 V2R7 and DFSMS 1.5 there were dramatic changes to the way OS/390 HFS file systems work. The most notable of these changes are the addition of HFS global buffers, the ability to perform HFS I/O asynchronously, and the ability to mount multi-volume file systems. These enhancements, and others, resulted in dramatic HFS performance improvements. Thus, the proper tuning of OS/390 HFS is becoming a critical and non-trivial component of overall OS/390 performance. Many of OS/390's performance problems can be traced to poor decisions related to HFS datasets. The good news is that DFSMS 1.5 vastly improved HFS performance, and added some new controls for tuning these important datasets. However, the bad news is that most of these new controls have not been well documented and explained. This article will attempt to provide some practical suggestions for monitoring and improving HFS

performance. It provides you with the information you need to understand and evaluate the performance of the HFS file system, along with practical hints and tips. It provides sufficient information for you to start monitoring HFS and evaluate its performance in your DFSMS environment.

ONLINE MONITORING OF HFS PERFORMANCE

When it comes to online performance monitoring tools for HFS file systems there is currently only one option available within the standard IBM's toolkit – RMF Monitor II HFS report. It is invoked by specifying 5 on the I/O Report Selection menu of the RMF Monitor II panel. The HFS File System statistics report it produces provides data for basic performance analysis of HFS, which enables you to identify potential problems and bottlenecks within the HFS component and to take corrective actions. The contents of the report as well as its field descriptions are described in the *RMF: Report Analysis* (SC33-7991) manual. A systems programmer is well aware of the fact that when setting report options (RO) only one file system name can be selected, and, if you try to select several files, RMF will complain and not a single file system name will be selected at all! On the other hand, it was found that RMF Monitor II HFS report is not as informative as it should be because it does not help you easily to identify your most I/O active files. This limitation was the initial impetus that prompted me to look for a tool/procedure that would allow me to get information about all mounted HFS file systems in a single run. To help alleviate the burden of monitoring HFS performance, as well as overcoming RMF's shortcomings, a simple yet easy to use REXX procedure was constructed. The USS confighfs query HFS global statistics command was used to display a system snapshot for all HFS datasets. In addition, the USS command df with option –S was used because it provides SMF I/O accounting for mounted files. It was also used to obtain the mount point for an HFS, which was needed for the confighfs command in order to get complete statistics for the dataset. The full meaning of the command and its invoking argument list can be obtained from *Unix System Services Command Reference* (SA22-7802).

HFSPM EXEC

```
/* REXX ************************************************************
   Procedure: HFSPM
   Description: Get current information on HFS performance
   Install: - Download BPXWUnix function (a part of "REXX Function
              Package for REXX in OpenEdition") from the IBM's "USS
              Tools and Toys page"
            - Restore it using the TSO/E receive inda() command.
            - Place this where REXX EXECs can be found.
  ******************************************************************/
signal ON ERROR
Address TSO
userid=SYSVAR(SYSUID)
outds  =userid||'.cnf.out'               /* Change dataset name   */
x = MSG('ON')                            /* to fit your standards */
if SYSDSN(outds) = 'OK'
Then "DELETE "outds" PURGE"
"ALLOC FILE(PRC) DA("outds")",
   " UNIT(SYSALLDA) NEW TRACKS SPACE(2,1) CATALOG",
   " REUSE LRECL(7Ø) RECFM(F B)"
 arg hlq
 if hlq = "" then HLQ = 'SYSTMØ5.USER'
Address ISPEXEC
"LIBDEF ISPLLIB DATASET ID('"hlq".LOAD') STACK"
call syscalls 'ON'
/*-----------------------------------------------------------------*/
/* Return USS information                                          */
/*-----------------------------------------------------------------*/
Address SYSCALL
'uname sys.'
/*-----------------------------------------------------------------*/
/* Print headers and labels                                        */
/*-----------------------------------------------------------------*/
sis.1 = left('HFS Performance report - produced on:',37,),
        ||left(' ',1,' ')||left(date(),11),
        ||left(' ',1,' ')||left('at ',3,' '),
        ||left(time(),1Ø)
sis.2 = left(' ',1)
sis.3 = left('System identification:',22)
sis.4 = left('Sysname: ',11)||sys.U_SYSNAME
sis.5 = left('Version: ',11)||sys.U_VERSION
sis.6 = left('Release: ',11)||left(sys.U_RELEASE,1Ø)
sis.7 = left('Node   : ',11)||left(sys.U_NODENAME,1Ø)
sis.8 = left('Hardware:',11)||left(sys.U_MACHINE,1Ø)
sis.9 = left(' ',1,' ')
/*-----------------------------------------------------------------*/
/* Get HFS data                                                    */
/*-----------------------------------------------------------------*/
Address SH
```

```
/*-------------------------------------------------------------*/
/* Construct confighfs command (fixed part of it). NOTE:       */
/* Unlike most z/OS Unix commands, which reside in /bin,       */
/* confighfs is found in the /usr/lpp/dfsms/bin directory      */
/*-------------------------------------------------------------*/
 fix  ='cd /usr/lpp/dfsms/bin;./confighfs -q '
/*-------------------------------------------------------------*/
/* Call display file command (df) to get file mount point      */
/* and  I/O activity since mounted                             */
/*-------------------------------------------------------------*/
  call BPXWUnix "df -S",,out.
  s = 1
  dfrc = rc
   If dfrc  <>Ø Then Do
     Say "Return Code         =" rc
     Say "OMVS Return Value   =" retval
     Say "OMVS Return Code    =" errno
     Say "OMVS Reason Code    =" errnojr
   End
  Do i = 2 to OUT.Ø                 /* Process each entry returned*/
  parse var out.i  mount . '(' HFS ')' rawdata .
  i = i + 1;
  read      =word(out.i,5); i =i + 1;       /* Read count         */
  write     =word(out.i,5); i =i + 1;       /* Write count        */
  ioblk     =word(out.i,6); i =i + 1;       /* Dir I/O bk.total   */
  reblk     =word(out.i,6); i =i + 1;       /* Dir I/O bkread     */
  wrblk     =word(out.i,6); i =i + 1;       /* Dir I/O bkwritten  */
  byread    =word(out.i,7); i =i + 1;       /* Total bytes read   */
  bywrite   =word(out.i,7)                  /* Total bytes written*/
  HFS      = left(HFS,25)                   /* Filesystem name    */
 if (HFS ¬= "/tmp") & (HFS ¬= "/dev")  then
  do
 Rw.s  =left('PERFORMANCE DATA FOR FILE:',27)||left(HFS,25); s=s+1;
 Rw.s  =left('Part 1',6);  s=s+1;
/*-------------------------------------------------------------*/
/* Construct confighfs command (variable part of it is added)  */
/*-------------------------------------------------------------*/
 cmd =fix||mount
/*-------------------------------------------------------------*/
/* Call confighfs command and process each entry returned      */
/*-------------------------------------------------------------*/
 call BPXWUnix cmd,,ott.
  cfgc = rc
   If cfgc  <>Ø Then Do
     Say "Return Code         =" rc
     Say "OMVS Return Value   =" retval
     Say "OMVS Return Code    =" errno
     Say "OMVS Reason Code    =" errnojr
   End
   Do k = 3 to OTT.Ø - 2
```

```
 VSTOR =strip(word(ott.k,3),L,'_'); k=k+2;   /* Virtual Storage   */
 FSTOR =strip(word(ott.k,3),L,'_'); k=k+2;   /* Fixed   Storage   */
 Lch   =strip(word(ott.k,4),L,'_'); k=k+1;   /* Lookup cache hit  */
 Lcm   =strip(word(ott.k,4),L,'_'); k=k+1;   /* Lookup cache miss */
 h6   =hitr(Lch Lcm)
 Fdph  =strip(word(ott.k,5),L,'_'); k=k+1;   /* 1st data page hit */
 Fdpm  =strip(word(ott.k,5),L,'_'); k=k+2;   /* 1st data page miss*/
 h5   =hitr(Fdph Fdpm)
 titl  = ott.k; k = k + 1;
/*-----------------------------------------------------------------*/
/* Get Storage allocated and buffer pool statistics                */
/*-----------------------------------------------------------------*/
 Rw.s  =left('Current Buffer pool use:',40); s=s+1;
 Rw.s  =left('Virtual Storage:',16)||right(vstor,5); s=s+1;
 Rw.s  =left('Fixed   Storage:',16)||right(fstor,5); s=s+1;
 Rw.s  =left(' ',3,' ' ,)||left(' ',29,' '),
        left('Already',7)  left('Not al.',7); s=s+1;
 Rw.s  =left('Pool',6)||left('Size',8),
        ||left('#DS',4)||left('BP_pages',9)||left('Fixed',7),
        ||left('fixed',8)||left('fixed',8); s=s+1;
 Rw.s  =left('-',50,'-'); s=s+1;
    do f = 1 to 4
    p.f   = word(ott.k,1)
    ps.f  = strip(word(ott.k,2),L,'_')
    pds.f = strip(word(ott.k,3),L,'_')
    pbp.f = strip(word(ott.k,4),L,'_')
    pf.f  = strip(word(ott.k,5),L,'_')
    paf.f = strip(word(ott.k,6),L,'_')
    pnf.f = strip(word(ott.k,7),L,'_')
 Rw.s =  right(p.f,3),                  /* Pool number             */
    right(ps.f,6),                      /* Pool size               */
    right(pds.f,6),                     /* Data spaces in pool     */
    right(pbp.f,6),                     /* # pages in pool - in use*/
    right(pf.f,6),                      /* Permanently fixed pages */
    right(paf.f,7),                     /* Already fixed pages     */
    right(pnf.f,7); s=s+1;              /* Not already fixed       */
    k = k+1;
     end
   k = k+3
 FSsz =strip(substr(ott.k,19,11),L,'_')    /* File system         */
 k= k+2
 Used =strip(word(ott.k,3),L,'_'); k=k+2;  /* Used pages          */
 Apgs =strip(word(ott.k,3),L,'_'); k=k+2;  /* Attribute pages     */
 Cpgs =strip(word(ott.k,3),L,'_'); k=k+2;  /* Cached pages        */
 Sio  =strip(word(ott.k,4),L,'_'); k=k+1;  /* Seq I/O reqs        */
 Rio  =strip(word(ott.k,4),L,'_'); k=k+1;  /* Random I/O reqs     */
 Lh   =strip(word(ott.k,3),L,'_'); k=k+1;  /* Look-up hit         */
 Lm   =strip(word(ott.k,3),L,'_'); k=k+1;  /* Look-up miss        */
 h2   =hitr(Lh Lm)
 Fph  =strip(word(ott.k,4),L,'_'); k=k+1;  /* 1st page hit:       */
```

```
 Fpm  =strip(word(ott.k,4),L,'_'); k=k+1;  /* 1st page miss       */
 h1   =hitr(Fph Fpm)
 Ixnt =strip(word(ott.k,4),L,'_'); k=k+1;  /* Index new tops      */
 Ixsp =strip(word(ott.k,3),L,'_'); k=k+1;  /* Index splits        */
 Ixjo =strip(word(ott.k,3),L,'_'); k=k+1;  /* Index joins         */
 Ixrh =strip(word(ott.k,4),L,'_'); k=k+1;  /* Index read hit      */
 Ixrm =strip(word(ott.k,4),L,'_'); k=k+1;  /* Index read miss     */
 h3   =hitr(Ixrh Ixrm)
 Ixwh = strip(word(ott.k,4),L,'_'); k=k+1; /* Index write hit     */
 Ixwm = strip(substr(ott.k,19,20),L,'_');  /*  Index write miss   */
 h4   =hitr(Ixwh Ixwm); k=k+1;
 Rflg = strip(substr(ott.k,19,20),L,'_'); k=k+1; /* RFS flags     */
 Rerr = strip(substr(ott.k,19,20),L,'_'); k=k+2; /* RFS errors    */
 Memc = strip(word(ott.k,3),L,'_'); k=k+1  /* Member count        */
 Sync = strip(word(ott.k,3),L,'_'); k=k+1  /* Sync interval       */
/*-----------------------------------------------------------------*/
/* Process File attributes (formatted to match RMF display)       */
/*-----------------------------------------------------------------*/
 Rw.s   =left(' ',3,' ');  s=s+1;
 Rw.s   =left('Part 2',6); s=s+1;
 Rw.s   =left('File attributes:',30); s=s+1;
 Rw.s   =left(' ',3,' ' ,); s=s+1;
 Rw.s   =left('---- File allocation (pages): --',34)||left(' ',3,' '),
         ||left('---- Index Events --',21); s=s+1;
 Rw.s   =left('System',9)    left(' ',1,' ') right(FSsz,7),
         ||left(' Used ',8)||left(' ',1,' ')||right(Used,6),
         ||left(' ',10,' '),
         ||left('New tops',9)||right(ixnt,4); s=s+1;
 Rw.s  = left('Attr.dir',9)||left(' ',3,' ')||right(Apgs,6),
         ||left('  Cached',8)||left(' ',1,' ')||right(Cpgs,7),
         ||left(' ',10,' '),
         ||left('Splits',9)||right(ixsp,4); s=s+1;
 Rw.s  = left('Members ',9)||left(' ',3,' ')||right(Memc,6),
         ||left('  Sync.int',10)||Left(' ',1,' ')||right(sync,12),
         ||left(' ',3,' ')||left('Joins',9)||right(ixjo,4); s=s+1;
 Rw.s  = left('RFS flag',9)||left(' ',3,' ')||right(rflg,6),
         ||left(' ',2,' ')||left('RFS error',9),
         ||right(rerr,5); s=s+1;
 Rw.s   =left(' ',3,' '); s=s+1;
/*-----------------------------------------------------------------*/
/* Process File's current I/Os (formatted to match RMF display)   */
/*-----------------------------------------------------------------*/
 Rw.s   =left('Part 3',6); s=s+1;
 Rw.s   =left('Current I/O activity count:',40); s=s+1;
 Rw.s   =left(' ',3,' '); s=s+1;
 Rw.s   =left(' ',9,' ')||left('-- File --',12),
         ||left('-- Metadata --',16)||left(' ',3,' '),
         ||left('-- Index --',13); s=s+1;
 Rw.s   =left('Cache',10)||right(Fph,5)||left(' ',8,' '),
         ||right(Lh,6)||left(' ',8,' '),
```

```
         ||left('read:',5)||right(ixrh,5)||left(' ',2,' '),
         ||left('write:',7)||right(ixwh,3); s=s+1;
 Rw.s   =left('DASD',10)||right(Fpm,5)||left(' ',8,' '),
         ||right(Lm,6)||left(' ',8,' '),
         ||left('read:',5)||right(ixrm,5)||left(' ',2,' '),
         ||left('write:',7)||right(ixwm,3); s=s+1;
 Rw.s   =left('Hit Ratio ',10)||right(h1,5)||left(' ',8,' '),
         ||right(h2,6)||left(' ',13,' ')||right(h3,5)||left(' ',6,' '),
         ||right(h4,6); s=s+1;
 Rw.s   =left('Seq.I/O',10)||right(sio,5); s=s+1;
 Rw.s   =left('Random',10)||right(rio,5); s=s+1;
 Rw.s   =left(' ',3,' '); s=s+1;
/*-----------------------------------------------------------------*/
/* File I/O activity since mounted - also found in SMF 92 rec.    */
/*-----------------------------------------------------------------*/
 Rw.s   =left('Part 4',6); s=s+1;
 Rw.s   =left('File I/O activity since mounted:',40); s=s+1;
 Rw.s   =left(' ',3,' ' ,); s=s+1;
 Rw.s  = left(' ',18,' ') left('- Dir I/O blocks -',25),
         left('Total bytes',12); s=s+1;
 Rw.s  = left('Reads',8) left('Writes',8),
         left('Total',8) left('Read',6)  left('Write',8),
         left('read',6)  left('written',7); s=s+1;
 Rw.s  = left('-',60,'-'); s=s+1;
 Rw.s = right(read,5) ,                 /* Number of reads         */
        right(write,8) ,                /* Number of writes        */
        right(ioblk,8) ,                /* Number dir. I/O block   */
        right(reblk,7) ,                /* Number read I/O blocks  */
        right(wrblk,7) ,                /* Number write I/O blocks */
        right(byread,8) ,               /* Total number bytes read */
        right(bywrite,8); s=s+1;        /* Total num.bytes written */
/*-----------------------------------------------------------------*/
/* Cache usage since mounted                                       */
/*-----------------------------------------------------------------*/
 Rw.s   =left(' ',3,' '); s=s+1;
 Rw.s   =left('Part 5',6); s=s+1;
 Rw.s   =left('Cache usage since mounted:',40); s=s+1;
 Rw.s   =left(' ',3,' '); s=s+1;
 Rw.s   =left(' ',7,' ')||left('File I/O count',16),
         ||left('Metadata I/O count',18); s=s+1;
 Rw.s   =left('-',41,'-'); s=s+1;
 Rw.s   =left('Cache',10)||right(Fdph,5)||left(' ',12,' '),
         ||right(Lch,6); s=s+1;
 Rw.s   =left('DASD',10)||right(Fdpm,5)||left(' ',12,' '),
         ||right(Lcm,6); s=s+1;
 Rw.s   =left('Hit Ratio ',10)||right(h5,5)||left(' ',12,' '),
         ||right(h6,6); s=s+1;
 Rw.s   =left(' ',3,' '); s=s+1;
     End
     End
```

```
    End /* main*/
call syscalls 'OFF'
/*-------------------------------------------------------------------*/
/* Write out USS System info and HFS info data                       */
/*-------------------------------------------------------------------*/
Address ISPEXEC "LIBDEF ISPLLIB";
Address TSO
"EXECIO * DISKW PRC (STEM sis.)"
"EXECIO * DISKW PRC (STEM Rw.)"
/*-------------------------------------------------------------------*/
/* Close & free allocated report file; then display result           */
/*-------------------------------------------------------------------*/
"EXECIO Ø DISKW PRC (FINIS "
  "free FILE(PRC)"
    Address ISPEXEC
   "ISPEXEC BROWSE DATASET('"outds"')"
  exit Ø
 /*-------------------------------------------------------------------*/
 /* Error exit routine                                                */
 /*-------------------------------------------------------------------*/
 ERROR: say 'The following command produced non-zero RC =' RC
        say SOURCELINE(SIGL)
        exit
HITR:
/* REXX - calculate Hit ratio */
arg a b
   SELECT
    when a ¬=  Ø Then do
       t  = a  + b
       hr = trunc((a/t)*1ØØ, 2)
      end
     otherwise  hr= Ø
   END
 return hr
```

The data returned by this procedure pertains to each HFS file system mounted and is grouped into five parts.

PERFORMANCE DATA FOR FILE OMVS.ETC

```
Part 1
Current Buffer pool use:
Virtual Storage: 4683
Fixed   Storage:    Ø
                                    Already Not al.
Pool  Size    #DS BP_pages Fixed  fixed   fixed
-------------------------------------------------
  1      1      1   4Ø83      Ø       Ø  686335
  2      4      1      8      Ø       Ø     2Ø4
```

```
  3     16      1     80      0       0     328
  4     64      1    512      0       0     616

Part 2
File attributes:

---- File allocation (pages): --    ---- Index Events ---
System       10800  Used       601          New tops    0
Attr.dir        30  Cached       6          Splits      0
Members        295  Sync.int  60(seconds)   Joins       0
RFS flag        43  RFS error    0

Part 3
Current I/O activity count:

         -- File --  -- Metadata --      -- Index --
Cache      1124          3667       read: 8346  write: 740
DASD        161          1557       read:  490  write:   0
Hit Ratio 87.47         70.19            94.45      100.00
Seq.I/O       0
Random      158

Part 4
File I/O activity since mounted:

                   - Dir I/O blocks -       Total bytes
Reads    Writes   Total    Read   Write    read   written
-------------------------------------------------------------
 5090      159    33533    7140     159 22369373    10423

Part 5
Cache usage since mounted:

       File I/O count  Metadata I/O count
----------------------------------------
Cache     15572            177703
DASD       1356            126692
Hit Ratio 91.98             58.37
```

Current buffer pool use (part 1) displays the number of virtual and permanently-fixed storage pages assigned to all four HFS I/O buffer pools. Comparing these actual usage numbers with the VIRTUAL and FIXED values may help you to determine when to adjust the storage thresholds. The table that follows is buffer pool assignment and it shows statistics for each of four buffer pools. This information is not provided by the RMF Monitor II HFS report. The following data is returned:

- *Pool* is the buffer pool ID. It designates one of the four HFS buffer pools. Under the current implementation this number is always in the range 1 to 4.
- *Size* is the buffer size for this pool (in pages 4KB, 16KB, 64K, and 256KB).
- *#DS* is the number of data spaces allocated to support the buffers in this buffer pool – usually set to 1 (one for each pool) except in the case of a very active system. HFS initially allocates to OMVS kernel four 2GB data spaces for four different buffer pools:
  - 4KB pool for small files (files that are less than or equal to 4KB in size), metadata, and most random requests.
  - 16KB and 64KB pool for intermediate sizes, for sequential file I/O if the system determines that this is an optimal buffer size, and for random file I/O if the block size of the file best fits in the buffer.
  - 256KB for large files, sequential file I/O if the system determines that this is an optimal buffer size, and for random file I/O if the block size of the file best fits in the buffer.
  - additional data spaces are allocated as needed.
- *BP_pages* is the number of virtual pages in this buffer pool currently in use.
- *Fixed* is the number of permanently fixed pages in this buffer pool.

  The last two columns are the measures of effectiveness of the page-fixed storage assigned to the buffer pools. For each buffer pool, the sum of these two columns will show the total number of buffer read and write requests – this is not a physical I/O count.
- *Already_fixed* is the number of times a buffer was already fixed prior to an I/O request in this buffer pool. This is a counter that is never decremented. By dividing this column by *sum* we will

get the hit rate percentage for the fixed storage assigned to each pool.

- *Not_already_fixed* is the number of times a buffer was not already fixed prior to an I/O request in this buffer pool. This is a counter that is never decremented.

File attributes (part 2) displays the allocation data (in pages) for each mounted HFS file system and index events. The first three space items (system, used, and attribute directory) may be used to make capacity-related decisions regarding your HFS datasets:

- *System* is the amount of storage allocated to this HFS.
- *Used* is the amount of storage pages internally used within HFS for data files, directories, and HFS internal structures (like the attribute directory).
- *Attr. Dir* is the amount of storage used for the attribute directory (AD). This number is included in the *Used* field. The attribute directory is the internal HFS structure (index) containing attribute information about individual file system objects as well as attributes of the file system itself.
- *Cached* is the amount of data within the HFS that has been moved into the virtual storage cache. This information may be used as a measure of activity of the file within the HFS. If enough virtual storage is made available, more pages of an HFS will be moved into cache as the files become more active.

  The index statistics are relative to all of the indices in the HFS dataset. The attribute directory (AD) is one index (the largest) but each directory (including the root) is also an index. The activity of index pages within the dataset is represented by three items:

  – *New tops* number shows how often HFS added a new level to its index structure, that is when the index is growing.

  – *Splits* number shows how often an index page was split into two pages because new records were inserted. This gives an idea of how much insertion activity there has been for the

index structure.

– *Joins*, contrary to a *new tops* and *splits* (both indicative of index growth), represents a shrinking of the index page. It shows how often HFS was able to combine two index pages into one, because enough index records had been deleted in the two pages.

The next four items in this part of report are not available if you use RMF Monitor II HFS report:

- *Members* is the number of nodes (entries) in the file system that have been used to represent files and directories. In fact, it is a crude approximation of the number of logical files contained within a dataset. Please refer to APAR OW39886 (USS confighfs command shows an incorrect member count) if your member count is incorrect (too high).
- *RFS flag* is HFS internal information and it shows the attributes of the file system at mount time.
- *RFS error* is HFS internal information that reports the errors that may have occurred while sync daemon was trying to harden data for this dataset. Watch for any non-zero value – it should be investigated before you lose more data.
- *Sync interval* is the interval used by the sync daemon for hardening all file data for a file system. Please remember that sync daemon uses vfs_sync operation in order to write to disk (or otherwise stabilizes) all changed data in a buffer cache for files in a mounted file system whenever all the HFS buffers of the file are filled. One should be very cautious when setting this parameter: if you specify SYNCDEFAULT=0 you will degrade your system performance by turning off deferred writes, but you will ensure data integrity of the application using that particular filesystem. (Remember that syncs are done at the HFS level, not file level, ie the sync process of the sync daemon will occur independently on each HFS file system. Even if sync intervals of all the HFS file systems are the same, the sync point of each HFS file system will be different.) On the other hand, be aware that if the system crashes before sync interval completes, the

user or metadata written since the last sync interval is lost.

Current I/O activity count (part 3) displays a snapshot of I/O accounting data for all mounted files, thus providing the data for understanding the throughput achieved by HFS, which, in turn, allows you to optimally use system resources. The reporting is done on three levels: file, metadata, and index.

File I/O count items:

- *Cache* is the number of times the first page of a data file was requested and found in virtual storage (cache).
- *DASD* is the number of times the first page of a data file was requested and was not found in virtual storage (cache), thus, I/O was required.
- *Hit Ratio* is the percentage of cache-found requests based on the total number of requests.
- *Seq I/O reqs* is the number of sequential file data I/O requests that have been issued. A sequential I/O is one of a series of I/Os to read or write a data file, where the first I/O started at the first byte of the file and each subsequent I/O was for the next sequential set of bytes. This is not meant to imply that actual disk I/O was required; the data may have resided in cache.
- *Random I/O reqs* is the number of random file data I/O requests that have been issued. A random I/O is an I/O that does not read or write the start of a file, and was not preceded by an I/O that read or wrote the immediately-preceding set of bytes. This is not meant to imply that actual disk I/O was required; the data may have resided in cache.

Metadata I/O count:

- *Cache* is the number of times the metadata for a file was found in virtual storage (cache) during file look-up (look-up hit).
- *DASD* is the number of times the metadata for the file was not found in virtual storage (cache) during file look-up and an index call was necessary, which may have resulted in I/O (look-up miss).

- *Hit Ratio* is the percentage of cache-found requests based on the total number of requests.

Index I/O count:

- *Cache* is the number of index page read or write hits.
- *DASD* is the number of index page read or write misses.
- *Hit Ratio* is the percentage of cache-found requests based on the total number of requests.

Note: these index hit/miss items report how efficient virtual storage cacheing was in providing the data needed without performing physical access to the data. When combining these indicators with cache items for the first page of a data file and metadata, one can get an idea about the total activity at the dataset level. I have noticed that the RMF Monitor II HFS report does not make any distinction between read and write activity. This is a serious shortcoming since any non-zero value in the *write miss* column should be taken very seriously because it indicates that the virtual storage cache is so heavily overloaded that output buffers cannot be provided.

The next two parts of this report are totally absent from the RMF Monitor II HFS report as well as from post-processor's HFS report.

The file I/O activity since mounted (part 4) report provides summary information on HFS I/O activity (number of reads/writes, directory activity, and number of bytes read/written) since the time the file system was mounted. It was found to be very useful because it can help you to identify high I/O activity files quickly.

The cache usage since mounted (part 5) report provides the data that can be used to analyse whether storage and buffer pool definitions are correct, or whether some adjustments should be performed to improve the performance of I/O activities for HFS files. This report is similar to the part 3 report except that it does not provide data on index activity.

COLLECTING THE HFS PERFORMANCE DATA

It is quite common to see a performance assessment being performed only after we have seen applications experiencing performance

problems or when analysts needs to know whether there is enough capacity to support growth or new USS workloads. When it comes to monitoring HFS performance we already know that data gathering for HFS file statistics will be performed in the Monitor III gatherer session. When enabled, the Monitor III gathers SMF record 74 subtype 6. If you want to get information about specific hierarchical file systems, you have to activate the Monitor III gatherer option HFSNAME(ADD(hfsname)). One can dynamically enable or disable this option by using the OS/390 operator commands:

```
F RMF,F III,HFSNAME(ADD(your.hfs.filename))
F RMF,F III,HFSNAME(DEL(your.hfs.filename))
```

After data gathering for HFS file statistics was performed in the Monitor III gatherer session, we can process collected SMF records either by invoking the RMF HFS postprocessor report or by running the code that I have provided here.

CODE

The code is a four-part stream (called HFSJOB). In the first step (DEL) auxiliary files are deleted, while in the second step (EXT746) the selected HFS-related SMF records are extracted from the SMF weekly/daily dataset and copied to a file that can be used as a base of archived records. It is worth noting that data related to granularity and quality of performance is very important. Too much data will slow the process and increase the resource consumption without providing additional benefit. Intervals for performance analysis should be chosen carefully: seven days of performance data is sufficient to ensure consistency and repeatability. To limit the amount of data collected, one may use the DATE and TIME filtering options the SMF dump program (IFASMFDP) provides. In the next step (SORT746) the extracted records are further filtered. In the fourth step (HFSREXX) the relevant records are formatted by invoking a corresponding REXX EXEC. The EXEC in this step, HFSREXX, is the EXEC that handles the 74.6 records.

SMF record 74 subtype 6 is a repository for HFS global activity, buffer pool statistics, and HFS file system statistics. These subtypes are generated only if HFS dataset names are included. This EXEC may seem to be unnecessary since there is a postprocessor for HFS

reports. The main reason for writing this EXEC is this: the HFS postprocessor report is interval based and each report it produces is in fact a collection of several reports each reporting on resource being monitored. This makes each single interval report very dense – one has to be quite skilful in finding out what to look for and where to look. Contrary to this kind of dense reporting, the HFSREXX EXEC is indicator oriented: each performance indicator is separately reported on, thus making it easier to notice the peaks. This is not meant to be a replacement for the postprocessor's report, but rather a supplement to it: once we have spotted a peak value or an exception, we can turn to the postprocessor's report and analyse all the interval reports.

There is a set of three reports produced by this stream, each providing in-depth information on a certain aspect or domain of HFS performance, which allows you to tune your system and make better use of HFS resources. The first one is HFS dataset I/O-related (buffer pool statistics), the second one is on HFS global statistics, and the last one is on HFS file system statistics.

Generally speaking, tuning HFS's I/O is not that different from what a performance analyst normally does to tune any kind of I/O subsystem, and therefore the same general rules apply: avoid unnecessary I/Os, complete most of the I/Os in memory, and complete the real I/Os as fast as possible. It was noted a long time ago that in most applications, I/O activity inversely correlates to performance. Therefore avoiding unnecessary I/Os is primarily an application issue: sometimes it is possible to do something from outside an application but normally we have to understand and change the application's behaviour. What we can really do depends on the application itself. What a performance analyst can recommend nevertheless is this: since USS stores its HFS files on the z/OS side and an I/O for USS may begin in the kernel address space, nonetheless it will use systems services that will also include processing serviced by z/OS. This means that minimizing the amount of data sharing that occurs between USS and z/OS-based applications will yield a performance gain.

In other words, try to isolate the HFS datasets as much as possible, so that USS I/O requests contend only with each other. One way to

achieve this I/O separation is to place HFS datasets on DASD that contain infrequently-referenced z/OS datasets. It is also a good idea is to spread high-activity HFS datasets across multiple volumes to keep user HFS datasets separate from system HFS datasets. Remember too that an HFS dataset is a standard OS/390 PDSE structure only in content. The PDSE access method is used only to open the file. After it is opened, it is managed by Unix services. Therefore, the HFS datasets do not benefit from enhanced internal processing for PDSE searches. DFSMS will not use expanded storage for hiperspaces staging.

The second guideline tells us to complete most of the I/Os in memory and at this stage there are essentially three techniques available – TFS, FILECACHE, and HFS global buffering.

Temporary File System (TFS) is an in-storage-only file system (similar to VIO) and can be used for read/write files. The main benefit of using a TFS is a dramatic improvement in file I/O performance since the I/O is as fast as a memory-to-memory-only access, and it does not incur the overhead of the HFS global buffers. The main issue to consider is that a TFS mounted file system is not backed up by physical disks. So if the system crashes, all data in TFS will be lost. It is for this reason that TFS is normally useful for temporary data only. A tuning tip: according to USS manuals, the storage assigned to TFS is accounted to the OMVS address space, so if there is a need to intensively use TFS one may like to consider the option of setting up a 'colony address space' so as to isolate TFS from the kernel. This would prevent virtual storage constraint in the kernel address space caused by the TFS. This means that one has to find the balance between memory one is willing to devote to TFS for improved performance and dedication of storage to TFS.

Filecache is a command that allows one to cache commonly-used read-only files in a data space belonging to the OMVS kernel address space. The amount of storage occupied by the filecache is equal to the sum of the size of each file – thus care should be taken that large files do not put a strain on processor storage (in a storage-constrained environment this could cause paging). Filecache improves system and end user read response times since just a

memory-to-memory copy is done and some 33% to 50% reductions in CPU time to access data have being reported. Good candidates for file cache include commonly-executed shell scripts, commonly-executed binaries, and commonly-referenced read-only files. Files cached using the filecache command could be read/write files, but consider that if cached data is being modified, that data is deleted from the cache and any further data access will be from disk. The only way you have to refresh the cache is by issuing the filecache -r command. This command will refresh all the cached files.

HFS global buffering is the latest option available and, despite its current limitations, the most interesting one. The underlying idea is to provide a cache for all HFS data and metadata. However, there is very little one can do to influence the system behaviour in utilizing HFS global buffers.

The buffer pool statistics report provides information about activities and storage usage within your z/OS Unix environment. This data can be used to analyse whether definitions are correct, or whether some adjustments should be performed to improve the performance of I/O activity for HFS files. The following parameters control the HFS buffer usage:

- *VlRTUAL(max)* specifies the maximum amount of virtual storage (in megabytes) that HFS data and metadata buffers can use. HFS may temporarily exceed the *max* limit to avoid failure of a file read or write request, but the amount of space used is reduced to the *max* specification more or less as soon as possible. As in many other cases, the amount of storage needed depends on the workload and system configuration. If you find your system is using more buffers because of heavy I/O to the HFS, and processor storage contention exists, setting a lower value on the *VIRTUAL* parameter may relieve this situation.
- *FIXED(min)* specifies the minimum amount of virtual storage (in megabytes) that is fixed at HFS initialization and permanently remains fixed even if HFS activity drops to zero. Basically, *FIXED* is used to ensure that storage is there when needed. The specified value of *min* must be less than or equal to *VIRTUAL(max)*. The benefit of *FIXED* is to avoid the overhead of page fixing and unfixing needed for I/O, and thus it minimizes

CPU utilization and reduces RSM lock contention. HFS will continue to temporarily fix additional buffers, as needed, during I/O requests. In addition, HFS will unfix storage and go below *FIXED(min)* if there is a pageable storage shortage in the system. By maintaining a pool of buffers already fixed, the system will shorten the path length of I/O operations, avoiding having to page fix and page free the buffers for every I/O operation out of the HFS global buffer pool. Obviously a fixed buffer uses real storage frames so you have to find the right trade-off to guarantee good HFS performance without degrading your system.

From a performance point of view it is interesting to observe that once assigned to the pools the fixed buffers will not be redistributed: we can have a lot of unused fixed buffers in one pool and none in the others. In order to avoid this, one may need to specify FIXED(0) in parmlib and use the confighfs command after the system start to raise the fixed minimum to the target value. By using this technique, fixed buffers will be allocated to the four pools, depending on workload demands. The choice of 'a right buffer pool' depends on a rather complex algorithm based on the following factors: the amount of virtual storage dedicated to HFS global buffers, the amount of fixed real storage dedicated to HFS global buffers, the distribution of storage to four buffer pools, the I/O pattern (random/sequential), the operation type (read/write), the I/O block size, the file size, and the file type. In brief, when an application needs to acquire an HFS buffer, HFS chooses a buffer based on a preference scheme. Since the purpose of allocating a new buffer is generally to do I/O, which requires that the buffer be page-fixed, the first preference is to use a permanently fixed buffer. (This statement is obvious for reads, but even a deferred write will cause an I/O in the near future.) If no permanently fixed buffers are available, the second preference is to use a pageable buffer that is already backed by real storage if one is available. If all such buffers are in-use, then a new buffer is acquired, and the first time that the buffer is touched, the Real Storage Manager (RSM) will allocate some real storage.

Finally, when we consider the third guideline (to complete the 'real' I/Os as fast as possible) we have to be aware of the fact that a file system is usually a single dataset only from the z/OS side, while in

fact the underlying structure of directories and files inside a file system can be quite complex and often includes most of the files of a single application.

It is highly recommended that you review the potential for increased HFS buffers and how the increased virtual storage usage may affect the current system. Installations may be especially susceptible to the default specification if they do their migration testing on a system image with less available central storage configured than the target production system. Production systems will generally have more central storage configured, and/or have higher contention for the central storage. Additional information is available on how to set and measure the HFS buffer definition in *Hierarchial File System Usage Guide* (SG24-5482-00).

Two additional reports that are useful when monitoring HFS dataset activity and performance are HFS global statistics report and HFS file system statistics report.

HFS global statistics report provides overall data about I/O activities of HFS files. Fields in this report include total amount of virtual storage assigned to I/O buffers (virtual used), total amount of permanently fixed storage assigned to I/O buffers (fixed used), file I/O statistics (cache/DASD), and metadata I/O (cache/DASD).

The HFS file system statistics report includes data gathered about I/O activity and the internal structure (index) of the HFS datasets. Some key indicators to observe include: mount point, space (allocated/used/ cached/index), I/O activity (data/metadata), index activity (read/write/ split/join/create), and cache effectiveness (data/ metadata/index). The meaning of these fields can be obtained from the *RMF Report Analysis* (SC33-7991) manual.

The full version of the code is available to subscribers to MVS Update *at the Xephon Web site, www.xephonUSA.com/mvs/trial. Please see back cover for subscription information or contact csmith@tcipubs.com. You may reach Xephon by phone at (214) 340 5690 or fax (214) 341 7081.*

Mile Pekic
Systems Programmer (Serbia and Montenegro)

WANT TO SUBSCRIBE?

Each monthlyly issue of *MVS Update* is packed with ideas for improving *your* MVS installation – and all of the technical details necessary for *you* to put those ideas into practice.

With *MVS Update* ***you*** get:

- A practical toolkit of ready-made enhancements, developed and tested in a working environment by MVS experts throughout the world.
- Tutorial articles on system internals.
- Performance tuning tips and measurements.
- Early user reports on new products and releases.

Plus:

- *MVS Update* will repay the cost of subscribing many times over!

All for a fraction of the cost of a single training course! So what are *you* waiting for?

- Go to http://www.xephonUSA.com/subscribe now to subscribe!
- Subscribe now and receive 25% off of a 12-month subscription*!

***** using promotional code **MV8X43**.

* * *

WANT TO CONTRIBUTE?

MVS Update is written by technical professionals just like you with a desire to share their expert knowledge with the world.

Xephon is always seeking talented individuals to contribute articles to *MVS Update* – **and get paid for it!**

If you have insight into how to make MVS more functional, secure, reliable, user friendly, or to generally improve MVS performance, please visit http://www.xephonUSA.com/contribute for more details on how you can contribute to this definitive industry publication.