# 227

# MVS update

*August 2005*

## In this issue





A xephon PUBLICATION

## 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.*

# Monitoring HFS out of space conditions

## INTRODUCTION

This article describes a new DFSMS function implemented by APAR OW44631 (R1E0 – UW74186/R1F0 – UW74187).

This new function provides support for monitoring how full a Hierarchical File System (HFS) is and will issue an operator warning message IGW023A when the HFS exceeds a user-specified threshold, eg:

```
*IGW023A HFS SYS2.USS.SUFHFS EXCEEDS 85% FULL
```

The fullness of an HFS is based on the number of pages currently in use versus the currently allocated HFS filesystem size.

You can display the current usage of an HFS using the **confighfs** USS command (the **confighfs** command is located in the */usr/lpp/dfsms/bin* directory):

```
SXSP001:/: >confighfs /u/suf
Statistics for file system SYS2.USS.SUFHFS
( 03/13/03 11:16am )
 File system size:______1620    (pages)          - allocated pages
                  _________6.328(MB)
 Used pages:      ______1393    (pages)          - used pages
                  _________5.441(MB)
 Attribute pages: _________9    (pages)
                  _________0.035(MB)
 Cached pages:    ______1622    (pages)
                  _________6.336(MB)
 Seq I/O reqs:    ____________________0
 Random I/O reqs: ____________________0
 Lookup hit:      ___________________110
 Lookup miss:     ____________________0
 1st page hit:    ____________________0
 1st page miss:   ___________________47
 Index new tops:  ____________________1
 Index splits:    ____________________3
 Index joins:     ____________________0
 Index read hit:  ___________________91
 Index read miss: ____________________1
```

```
 Index write hit: ____________________189
 Index write miss:____________________Ø
 RFS flags        ____________________43(HEX)
 RFS error flags: ____________________Ø(HEX)
 High foramt RFN: ____________________59C(HEX)
 Member count:    ____________________62
 Sync interval:   ____________________3Ø(seconds)
SXSPØØ1:/: >
```

Another way to get the same kind of information is to use ISPF Option 3.2 on the HFS dataset:

```
                              Data Set Information
 Command ===>

 Data Set Name  . . . : SYS2.USS.SUFHFS

 General Data                              Current Allocation
  Management class . . : MIG3Ø              Allocated cylinders : 9
  Storage class  . . . : HFSSC              Allocated extents . : 3
   Volume serial . . . : HFSØØ1             Maximum dir. blocks : NOLIMIT
   Device type . . . . : 339Ø
  Data class . . . . . : **None**
   Organization  . . . : PO                 Current Utilization
   Record format . . . : U                   Used pages  . . . . : 1,393
   Record length . . . : Ø                   % Utilized  . . . . : 85
                                                                   - above 7Ø%
   Block size  . . . . : Ø                   Number of members . : 62
   1st extent cylinders: 7
   Secondary cylinders : 1
   Data set name type  : HFS

   Creation date . . . : 2ØØ3/Ø3/13         Referenced date . . : 2ØØ3/Ø3/13
   Expiration date . . : ***None***
```

At this point, you can see that this HFS is really more than 70% full!

You can increase the HFS allocation by 2MB using the **hfsconfig** command:

```
SXSPØØ1:/: >confighfs -x 2m /u/suf
```

You can use the **confighfs** command to get more detailed information:

```
SXSPØØ1:/: >confighfs /u/suf
Statistics for file system SYS2.USS.SUFHFS
( Ø3/13/Ø3 3:24pm )
```

```
 File system size:______2160    (pages)            - new allocation
                  _________8.438(MB)
 Used pages:      ______1393    (pages)
                  _________5.441(MB)
 Attribute pages: _________9    (pages)
                  _________0.035(MB)
 Cached pages:    _________0    (pages)
                  _________0.000(MB)
 Seq I/O reqs:    ____________________0
 Random I/O reqs: ____________________0
 Lookup hit:      ____________________2
 Lookup miss:     ___________________17
 1st page hit:    ____________________0
 1st page miss:   ____________________0
 Index new tops:  ____________________0
 Index splits:    ____________________0
 Index joins:     ____________________0
 Index read hit:  ___________________47
 Index read miss: ____________________1
 Index write hit: ____________________0
 Index write miss:____________________0
 RFS flags        ___________________43(HEX)
 RFS error flags: ____________________0(HEX)
 High foramt RFN: __________________59C(HEX)
 Member count:    ___________________62
 Sync interval:   ___________________30(seconds)

SXSP001:/: >
```

Using ISPF Option 3.2, you can see that the HFS dataset now has four extents and that the space usage is 64%:

```
Data Set Information
 Command ===>

 Data Set Name  . . . : SYS2.USS.SUFHFS

 General Data                          Current Allocation
  Management class . . : MIG30          Allocated cylinders : 12
  Storage class  . . . : HFSSC          Allocated extents . : 4
   Volume serial . . . : HFS001         Maximum dir. blocks : NOLIMIT
   Device type . . . . : 3390
  Data class . . . . . : **None**
   Organization  . . . : PO            Current Utilization
   Record format . . . : U              Used pages  . . . . : 1,393
   Record length . . . : 0              % Utilized  . . . . : 64
                                                        - below 70%
   Block size  . . . . : 0              Number of members . : 62
   1st extent cylinders: 7
   Secondary cylinders : 1
```

```
   Data set name type  : HFS

   Creation date . . . : 2003/03/13     Referenced date . . : 2003/03/13
   Expiration date . . : ***None***
```

## IMPLEMENTATION

To implement this new monitoring option, you should modify SYS1.PARMLIB(BPXPRM00).

You can specify a default threshold and an incremental value through the **FILESYSTYPE TYPE(HFS)...** entry to set default values to be used for all HFS filesystems.

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 default values.

If no values are specified in either place, no threshold checking will take place.

The values applied to a filesystem can be changed only when the file system is mounted.

A **FSFULL(70,10)** parameter would cause DFSMS to issue message IGW023A when the HFS is 70% full, and then issue additional IGW023A messages when the file system is 80% and 90% full.

Parmlib member BPXPRM00 looks like this:

```
FILESYSTYPE TYPE(HFS)
  ENTRYPOINT(GFUAINIT)
  PARM('SYNCDEFAULT(30) FIXED(000) VIRTUAL(032) FSFULL(70,5)')
                                                  /* default value */

MOUNT FILESYSTEM('SYS2.USS.ETCHFS')
  TYPE(HFS)
  MODE(RDWR)
  MOUNTPOINT('/etc')
  PARM('FSFULL(60,5)')

                                                  /* override      */
```

*Alexandre Goupil*
*Systems Programmer (France)*

# Examining an LE trace for virtual storage memory leaks

Application programs frequently acquire storage dynamically as a way to minimize their static virtual storage requirements, as well as to be able to satisfy re-entrancy requirements. It is imperative, especially in applications that are intended to stay running for long periods of time, that dynamically-acquired storage be released when it is no longer required. Failure to do so will eventually cause the application to go short on storage and necessitate an application restart. With the long-running nature of many of today's applications, an application restart to simply alleviate a storage shortage condition is rarely an acceptable, recurring option.

High-level language applications generated from languages such as C, C++, PL/I, or COBOL and running on z/OS use Language Environment (LE) to manage internal storage requirements. Although LE makes use of traditional GETMAIN or STORAGE OBTAIN operations to acquire storage and FREEMAIN or STORAGE RELEASE operations to release storage, these requests are usually done at a macro level – that is, LE acquires relatively large blocks of virtual storage as necessary and parcels it out to the application as requests are made for dynamic storage. LE manages the storage acquisition/release requests, not the z/OS VSM (Virtual Storage Manager). This process renders the traditional GFS (Get Free Storage) trace somewhat useless in detecting application memory leaks. In these scenarios, the LE trace can prove to be invaluable.

## QUALIFYING THE PROBLEM

Although this problem can exist for any of the previously-mentioned programming languages, the balance of this article will focus on the problem as it specifically relates to using C or C++. There are three fundamental function calls in C/C++

that attempt to acquire dynamic storage. These are:

- calloc()
- malloc()
- realloc().

When these functions are successful, they cause a block of storage of the specified size to be set aside by LE for use by the application. This storage can then be used internally by the application and will remain available for its use until a free() function call is made to release the referenced storage area for reuse.

If an application makes use of calloc(), malloc(), or realloc() and does not use free() to release the storage when it is feasible, the storage block associated with the allocation request will remain flagged as 'in use' by LE. With enough of these situations, the application could eventually run into a storage shortage condition.

## USING THE LE TRACE

C/C++ supports the activation of the LE trace through run-time options. This can be done in a couple of different ways. One of those options is to enable tracing right within the program's source code. The C/C++ program code directives would look similar to:

```
#pragma runopts("TRACE(ON,1638ØK,DUMP,LE=8)")
#pragma runopts("RPTSTG(ON)")
```

This will cause the LE trace to be active and to cause storage events (allocation or free) to be recorded in the trace.

Trace events will be recorded by LE in its internal trace buffer. When the application terminates, trace information will be written to two output DDs. The SYSOUT output will contain summary information about HEAP and STACK usage. The CEEDUMP output will contain the LE trace table output. It is

```
Displacement                        Trace Entry in Hexadecimal                                  Trace Entry in EBCDIC
------------  ------------------------------------------------------------------------     --------------------------------
  +000000     Time 15.40.44.353229     Date 2005.05.24     Thread ID... 8000000000000000
  +000010     Member ID.... 03     Flags..... 000000     Entry Type..... 00000001
  +000018     94818995 40404040 40404040 40404040  40404040 40404040 40404040 40404040  |main                                |
  +000038     60606E4D F1F2F35D 40869985 854DF0A7  F0F0F0F1 C5F2F4F0 5D404040 40404040  |-->(123) free(0x0001E240)           |
  +000058     40404040 40404040 40404040 40404040  40404040 40404040 40404040 40404040  |                                    |
  +000078     40404040 40404040                                                         |                                    |

  +000080     Time 15.40.44.353251     Date 2005.05.24     Thread ID... 8000000000000000
  +000090     Member ID.... 03     Flags..... 000000     Entry Type..... 00000002
  +000098     4C60604D F1F2F35D 40D9F1F5 7EF0F0F0  F3F0F3F2 C140C5D9 D9D5D67E F0F0F0F0  |<--(123) R15=0003032A ERRNO=0000|
  +0000B8     F0F0F0F0 00000000 00000000 00000000  00000000 00000000 00000000 00000000  |0000............................|
  +0000D8     00000000 00000000 00000000 00000000  00000000 00000000 00000000 00000000  |................................|
  +0000F8     00000000 00000000                                                         |........                            |

  +000100     Time 15.40.44.353283     Date 2005.05.24     Thread ID... 8000000000000000
  +000110     Member ID.... 03     Flags..... 000000     Entry Type..... 00000001
  +000118     94818995 40404040 40404040 40404040  40404040 40404040 40404040 40404040  |main                                |
  +000138     60606E4D F1F2F45D 40948193 9396834D  F1F2F8F0 F0F05D40 40404040 40404040  |-->(124) malloc(128000)             |
  +000158     40404040 40404040 40404040 40404040  40404040 40404040 40404040 40404040  |                                    |
  +000178     40404040 40404040                                                         |                                    |
```

*Figure 1: Excerpt of CEEDUMP output (cont.)*

```
+000180     Time 15.40.44.353436     Date 2005.05.24     Thread ID... 8000000000000000
+000190     Member ID.... 03     Flags..... 000000     Entry Type..... 00000002
+000198     4C6060 4D F1F2F45D 40D9F1F5 7EF0F8C3   F2F5F0F2 F840C5D9 D9D5D67E F0F0F0F0   |<--(124) R15=08C25028 ERRNO=0000|
+0001B8     F0F0F0F0 00000000 00000000 00000000   00000000 00000000 00000000 00000000   |0000............................|
+0001D8     00000000 00000000 00000000 00000000   00000000 00000000 00000000 00000000   |................................|
+0001F8    00000000 00000000                                                            |........                        |

+000200     Time 15.40.44.379540     Date 2005.05.24     Thread ID... 8000000000000000
+000210     Member ID.... 03     Flags..... 000000     Entry Type..... 00000001
+000218     94818995 40404040 40404040 40404040   40404040 40404040 40404040 40404040   |main                            |
+000238     60606E4D F1F2F45D 40948193 9396834D   F1F2F8F0 F0F05D40 40404040 40404040   |-->(124) malloc(128000)         |
+000258     40404040 40404040 40404040 40404040   40404040 40404040 40404040 40404040   |                                |
+000278     40404040 40404040                                                           |                                |

+000280     Time 15.40.44.379721     Date 2005.05.24     Thread ID... 8000000000000000
+000290     Member ID.... 03     Flags..... 000000     Entry Type..... 00000002
+000298     4C6060 4D F1F2F45D 40D9F1F5 7EF0F8C3   F4F9F0F2 F840C5D9 D9D5D67E F0F0F0F0   |<--(124) R15=08C49028 ERRNO=0000|
+0002B8     F0F0F0F0 00000000 00000000 00000000   00000000 00000000 00000000 00000000   |0000............................|
+0002D8     00000000 00000000 00000000 00000000   00000000 00000000 00000000 00000000   |................................|
+0002F8    00000000 00000000                                                            |........                        |
```

*Figure 1: Excerpt of CEEDUMP output*

the output contained in CEEDUMP that the RPTSTGA program, provided with this article, will examine to ascertain whether or not the program that produced the CEEDUMP output has any potential memory leak conditions.

Figure 1 is a brief excerpt of CEEDUMP output.

Two trace entry types exist in the excerpt in Figure 1. A trace entry type 1 event is produced for base C library function calls. A trace entry type 2 event is produced for base C library function returns. What this means is that two trace events are logged for each function call that is traced – one for the initiation of the function call and one for the function call return. A CEEDUMP RPTSTG trace with TRACE runopts LE=8 will contain eight possible entries:

- Entry type 1 for calloc() function calls. Look for '-->(122)' in the EBCDIC display for the trace entry.
- Entry type 2 for calloc() function returns. Look for '<--(122)' in the EBCDIC display for the trace entry.
- Entry type 1 for free() function calls. Look for '-->(123)' in the EBCDIC display for the trace entry.
- Entry type 2 for free() function returns. Look for '<--(123)' in the EBCDIC display for the trace entry.
- Entry type 1 for malloc() function calls. Look for '-->(124)' in the EBCDIC display for the trace entry.
- Entry type 2 for malloc() function returns. Look for '<--(124)' in the EBCDIC display for the trace entry.
- Entry type 1 for realloc() function calls. Look for '-->(125)' in the EBCDIC display for the trace entry.
- Entry type 2 for realloc() function returns. Look for '<--(125)' in the EBCDIC display for the trace entry.

For calloc(), malloc(), and realloc() entry type 2 events, a non-zero R15 in the trace entry indicates a successful allocation request. For free() entry type 2 events, an R15 value of zero indicates a successful storage free request.

The LE trace supports other entry types, but they are outside the scope of the discussion in this article. For more information on the LE trace and LE trace table entries see the section entitled 'Understanding the trace table entry (TTE)' in the *z/OS Language Debugging Guide* from the Language Environment bookshelf.

## VIRTUAL STORAGE USAGE ANALYSIS

To assess a C/C++ program for potential memory leak conditions, the CEEDUMP output described earlier must be captured to a real dataset. This can be done either by pre-allocating the dataset and specifying that dataset in the CEEDUMP DD in the program's JCL or by capturing the JES SYSOUT data for CEEDUMP to a dataset after the fact using SDSF or a comparable spool display tool. Once the CEEDUMP output has been captured to a dataset, the RPTSTGA program can be used to check for memory leaks. The premise of the RPTSTGA program is that it tries to find a free() request to correspond to any calloc(), malloc(), or realloc() request. Any calloc(), malloc(), or realloc() request that does not have a corresponding free() request is flagged as a memory leak condition and is reported on. Note: a realloc() request contains an implied free().

An example C program, ALLOCFRE, has been provided for testing purposes. The ALLOCFRE program establishes the necessary runopts and then runs through a series of malloc(), calloc(), and free() requests until the address space runs out of storage. The following JCL can be used to compile, prelink, and link-edit the ALLOCFRE program:

```
//PROCS    JCLLIB ORDER=(CBC.SCBCPRC)
//STEP1   EXEC EDCC,CPARM='LIST',
//         CPARM2='RENT,NOSEARCH,NOMAR,NOSEQ,NOOPT',
//         CPARM3='LANGLVL(EXTENDED),SOURCE,LONGNAME,SSCOMM',
//         INFILE=c.source.dataset(ALLOCFRE),
//         OUTFILE='object.code.pds(ALLOCFRX),DISP=SHR',
//         SYSLBLK=8000
//COMPILE.SYSLIB  DD DSN=CEE.SCEEH.H,DISP=SHR
//         DD DSN=CEE.SCEEH.SYS.H,DISP=SHR
```

```
//          DD DSN=TCPIP.SEZACMAC,DISP=SHR
//PLKED1   EXEC PGM=EDCPRLK,PARM='UPCASE',
//     REGION=2Ø48K
//SYSMSGS   DD  DSNAME=CEE.SCEEMSGP(EDCPMSGE),DISP=SHR
//SYSLIB    DD  DUMMY
//SYSOBJ    DD  DSN=object.code.pds,DISP=SHR
//SYSMOD    DD  DSN=object.code.pds(ALLOCFRO),DISP=SHR
//SYSOUT    DD  SYSOUT=*
//SYSPRINT DD  SYSOUT=*
//SYSIN     DD  *
  INCLUDE SYSOBJ(ALLOCFRX)
//IEWL      EXEC  PGM=HEWLHØ96,PARM='XREF,LIST,MAP,RENT'
//SYSPRINT DD    SYSOUT=*
//SYSUT1    DD    UNIT=SYSDA,SPACE=(CYL,(2,1))
//OBJECT    DD    DSN=object.code.pds,DISP=SHR
//SYSLIB    DD    DSN=TCPIP.SEZACMTX,DISP=SHR
//          DD    DSN=CEE.SCEELKEX,DISP=SHR
//          DD    DSN=CEE.SCEELKED,DISP=SHR
//          DD    DSN=SYS1.CSSLIB,DISP=SHR
//          DD    DSN=SYS1.LINKLIB,DISP=SHR
//SYSLMOD   DD    DSN=load.library,DISP=SHR
//SYSLIN    DD    *
   INCLUDE OBJECT(ALLOCFRO)
   ENTRY   CEESTART
   NAME    ALLOCFRE(R)
```

Expect the prelink step from the above JCL to complete with a condition code of 4, but the compile and link-edit steps should both complete with a condition code of 0. When the ALLOCFRE load module has been created, the following JCL can be run to create a CEEDUMP dataset:

```
//ALLOCFRE EXEC PGM=ALLOCFRE,REGION=18M
//STEPLIB  DD   DSN=load.library,DISP=SHR
//SYSPRINT DD   SYSOUT=*
//CEEDUMP  DD   DSN=le.trace.ceedump.output,DISP=(,CATLG),
//            SPACE=(TRK,(5,2)),
//            DCB=(LRECL=132,RECFM=FB,BLKSIZE=132Ø)
```

The CEEDUMP output dataset can now be processed by the RPTSTGA program.

## RPTSTGA PROGRAM

The following is sample JCL to run the RPTSTGA program:

```
//RPTSTGA  EXEC PGM=RPTSTGA
//STEPLIB  DD   DSN=load.library,DISP=SHR
```

```
//INPUT    DD   DSN=le.trace.ceedump.output,DISP=SHR
//OUTPUT   DD   SYSOUT=*
```

If RPTSTGA is able to pair up all calloc(), malloc(), and realloc() requests with a corresponding free() request and there are no orphaned free() requests (free() trace entries that do not have a prior calloc(), malloc(), or realloc() entry), RPTSTGA will end with a return code of 0 and produce one line of output in the OUTPUT dataset as follows:

```
ALL ALLOCATED STORAGE HAS BEEN RELEASED
```

If RPTSTGA does not find a free() trace entry for all the calloc(), malloc(), or realloc() requests, RPTSTGA will end with a return code of 4 and produce a report in the OUTPUT dataset containing information similar to the following:

```
POTENTIAL MEMORY LEAK DETECTED
ADDR 0CE3E028 LEN(00046400) not released.  Tm: 20.52.03.957020  Dt:
2005.05.09
ADDR 0CE16BE8 LEN(00000252) not released.  Tm: 20.52.03.957125  Dt:
2005.05.09
ADDR 0CE16CF0 LEN(00000252) not released.  Tm: 20.52.03.957167  Dt:
2005.05.09
ADDR 0CE16DF8 LEN(00000252) not released.  Tm: 20.52.03.957206  Dt:
2005.05.09
ADDR 0CE16F00 LEN(00000252) not released.  Tm: 20.52.03.957323  Dt:
2005.05.09
ADDR 0CE17008 LEN(00001024) not released.  Tm: 20.52.03.976742  Dt:
2005.05.09
ADDR 0CE17410 LEN(00000009) not released.  Tm: 20.52.03.980889  Dt:
2005.05.09
ADDR 0CE1BC98 LEN(00000200) not released.  Tm: 20.52.04.835534  Dt:
2005.05.09
ADDR 0CE1C0B0 LEN(00000200) not released.  Tm: 20.52.04.835840  Dt:
2005.05.09
ADDR 0CE1C180 LEN(00000200) not released.  Tm: 20.52.04.836054  Dt:
2005.05.09
ADDR 0CE1C250 LEN(00000200) not released.  Tm: 20.52.04.836192  Dt:
2005.05.09
ADDR 0CE1C340 LEN(00000200) not released.  Tm: 20.52.04.836378  Dt:
2005.05.09
```

Notice that the storage address, storage length, and time stamp of the trace entry are indicated. This allows you to go back into the CEEDUMP output to identify the offending trace entry and then initiate corrective action.

As was mentioned earlier, it is possible to have free() trace entries that don't match up with any prior calloc(), malloc(), or realloc() requests. These can be rogue requests or they can be valid requests to free storage for other C function calls that acquire dynamic storage such as strdup(). If this occurs, RPTSTGA will also end with a return code of 4 and produce an output record in the OUTPUT dataset similar to:

```
Orphaned trace free() entry:  Time 20.52.04.826479 Date 2005.05.09
```

## CONCLUSION

RPTSTGA is a very useful tool for assessing virtual storage usage in application code that has been written in a high-level language. If you will be creating long-running application code written in a high-level language that uses LE to manage dynamic storage, you should consider using a tool like RPTSTGA prior to releasing the application into production use. It will allow you to examine virtual storage usage to determine whether anything unexpected is occurring with dynamic storage acquisition or release.

## RPTSTGA ASSEMBLER

```
*-------------------------------------------------------------------*
*   The RPTSTGA program examines the CEEDUMP output data from high-  *
*   level language programs that uses LE (Language Environment) to    *
*   manage the acquisition and release of dynamic storage.  For       *
*   this example, a C program has been used - two #pragma statements  *
*   are used in the C program source that LE should trace and report  *
*   program dynamic storage usage.  These #pragma statements are:     *
*                                                                     *
*   #pragma  runopts("TRACE(ON,16380K,DUMP,LE=8)")                    *
*   #pragma  runopts("RPTSTG(ON)")                                    *
*                                                                     *
*   With those directives in place, LE will trace and report on all   *
*   calloc(), malloc(), realloc(), and free() function calls that     *
*   are made in the C program.                                        *
*                                                                     *
*   The RPTSTGA program examines the trace output produced in the     *
*   CEEDUMP output to determine whether or not there are any memory   *
*   leak conditions (a calloc(), malloc(), or realloc() function      *
*   call with no corresponding free() function call) in the program   *
*   source.  As well, the RPTSTGA program will also track free()      *
```

```
*   function calls that don't have a prior calloc(), malloc(), or     *
*   realloc() association.                                            *
*                                                                     *
*   Assemble the RPTSTGA source and use the following JCL to create   *
*   the program load module:                                          *
*                                                                     *
*      //IEWL     EXEC  PGM=HEWLHØ96,PARM='XREF,LIST,MAP'            *
*      //SYSPRINT DD    SYSOUT=*                                     *
*      //SYSUT1   DD    UNIT=SYSDA,SPACE=(CYL,(2,1))                 *
*      //OBJECT   DD    DSN=object.code.pds,DISP=SHR                 *
*      //SYSLIB   DD    DSN=SYS1.CSSLIB,DISP=SHR                     *
*      //SYSLMOD  DD    DSN=load.library,DISP=SHR                    *
*      //SYSLIN   DD    *                                            *
*         INCLUDE OBJECT(RPTSTGA)                                   *
*         ENTRY   RPTSTGA                                           *
*         NAME    RPTSTGA(R)                                        *
*                                                                   *
*   The following JCL can be used to produce a storage analysis     *
*   report from an LE storage trace:                                *
*                                                                   *
*   //RPTSTGA  EXEC PGM=RPTSTGA                                     *
*   //STEPLIB  DD   DSN=load.library,DISP=SHR                       *
*   //INPUT    DD   DSN=LE.CEEDUMP.trace.output,DISP=SHR            *
*   //OUTPUT   DD   SYSOUT=*                                        *
*-------------------------------------------------------------------*
RPTSTGA  CSECT
RPTSTGA  AMODE 31
RPTSTGA  RMODE 24                    RMODE=24 BECAUSE DCBs BELOW LINE
         STM   R14,R12,12(R13)       SAVE THE REGISTERS
         LR    R12,R15               COPY BASE REGISTER
         USING RPTSTGA,R12           SET ADDRESSABILITY
         LR    R11,R1                SAVE INCOMING PARM ADDRESS
         LR    R3,R13                SAVE INCOMING SAVEAREA ADDRESS
         STORAGE OBTAIN,LENGTH=WORKLEN,LOC=ANY
         LR    RØ,R1                 COPY THE ADDRESS
         LR    R14,R1                AGAIN
         LR    R2,R1                 AGAIN
         L     R1,=A(WORKLEN)        GET THE LENGTH
         XR    R15,R15               SET FILL BYTE VALUE
         MVCL  RØ,R14                CLEAR THE STORAGE
         ST    R3,4(,R2)             SAVE OLD SAVEAREA ADDRESS
         LR    R13,R2                GET NEW SAVEAREA ADDRESS
         USING WORKAREA,R13          SET ADDRESSABILITY
         LR    R1,R11                COPY PARM ADDRESS
*-------------------------------------------------------------------*
*   Open INPUT and OUTPUT datasets.                                 *
*-------------------------------------------------------------------*
         OPEN  (INPUT,INPUT),MODE=31 OPEN INPUT DATASET
         OPEN  (OUTPUT,OUTPUT),MODE=31 OPEN OUTPUT DATASET
         USING ALCE,R2               SET ADDRESSABILITY TO ALCE
```

```
         LA    R8,INREC              GET INPUT RECORD ADDRESS
         GET   INPUT,(R8)            READ FIRST INPUT RECORD
         CLI   INREC+Ø,C'1'          CARRIAGE CONTROL CHARACTERS?
         BE    EVENTLP               YES - INREC ADDRESS IS FINE
         LA    R8,1(,R8)             OFFSET INPUT BY ONE BYTE
EVENTLP  EQU   *
         GET   INPUT,(R8)            GET INPUT DATA
*--------------------------------------------------------------------*
*   Capture the time and date information if this is a 'Time' record. *
*--------------------------------------------------------------------*
         CLC   INREC+22(4),=C'Time'  AN EVENT 'Time' RECORD?
         BNE   EVENTCHK              NO - CHECK FOR EVENT SPECIFICS
         MVC   TIMESAVE(15),INREC+27 SAVE THE TIME
         MVC   DATESAVE(1Ø),INREC+51 SAVE THE DATE
         B     EVENTLP               GET NEXT RECORD
EVENTCHK EQU   *
*--------------------------------------------------------------------*
*   It's not a 'Time' record.  Determine whether it's an event-      *
*   specific record that we are interested in.                       *
*--------------------------------------------------------------------*
         CLC   INREC+97(8),=C'-->(122)' A calloc()?
         BE    CALLOC                YES - PROCESS calloc()
         CLC   INREC+97(8),=C'-->(123)' A free()?
         BE    FREE                  YES - PROCESS free()
         CLC   INREC+97(8),=C'-->(124)' A malloc()?
         BE    MALLOC                YES - PROCESS malloc()
         CLC   INREC+97(8),=C'-->(125)' A realloc()?
         BE    REALLOC               YES - PROCESS realloc()
         B     EVENTLP               GET NEXT RECORD
CALLOC   EQU   *
*--------------------------------------------------------------------*
*   It's a calloc() event record.  Acquire some storage for an       *
*   allocation entry control block.                                  *
*--------------------------------------------------------------------*
         STORAGE OBTAIN,LENGTH=ALCELN,LOC=ANY
         ST    R1,CURRENT            SAVE ENTRY ADDRESS
         LR    R2,R1                 COPY IT
         MVC   ALCENXT(4),ALLOCPTR   COPY PREVIOUS ENTRY ADDRESS
         MVC   ALCETYPE(8),=C'calloc  ' SET ENTRY TYPE
         XR    R4,R4                 CLEAR R4
         XR    R5,R5                 CLEAR R5
         XR    R6,R6                 CLEAR R6
         XR    R7,R7                 CLEAR R7
         LA    R15,INREC+113         POINT TO COUNT AREA
CALLOC1Ø EQU   *
*--------------------------------------------------------------------*
*   Capture the calloc() count and size information.                 *
*--------------------------------------------------------------------*
         CLI   Ø(R15),C','           COUNT,SIZE SEPARATOR?
         BE    CALLOC2Ø              YES - GO PROCESS SIZE
```

```
         IC    R4,Ø(,R15)            GET DIGIT FROM COUNT AREA
         N     R4,=X'ØØØØØØØF'       TURN OFF ALL BUT LOW ORDER NIBBLE
         MH    R5,=H'1Ø'             MULTIPLY EXISTING VALUE BY 1Ø
         AR    R5,R4                 ADD IN NEW
         LA    R15,1(,R15)           POINT TO NEXT BYTE
         B     CALLOC1Ø              CHECK FOR MORE COUNT DIGITS
CALLOC2Ø EQU   *
         LA    R15,1(,R15)           POINT TO NEXT BYTE
CALLOC25 EQU   *
         CLI   Ø(R15),C')'           COUNT,SIZE TERMINATOR?
         BE    CALLOC3Ø              YES - GOT COUNT AND SIZE
         IC    R6,Ø(,R15)            GET DIGIT FROM SIZE AREA
         N     R6,=X'ØØØØØØØF'       TURN OFF ALL BUT LOW ORDER NIBBLE
         MH    R7,=H'1Ø'             MULTIPLY EXISTING VALUE BY 1Ø
         AR    R7,R6                 ADD IN NEW
         LA    R15,1(,R15)           POINT TO NEXT BYTE
         B     CALLOC25              CHECK FOR MORE SIZE DIGITS
CALLOC3Ø EQU   *
*--------------------------------------------------------------------*
*   Calculate and save the size of the calloc() request (count*size). *
*--------------------------------------------------------------------*
         XR    R4,R4                 CLEAR R4
         XR    R6,R6                 CLEAR R6
         MR    R4,R7                 GET AREA SIZE
         CVD   R5,DBL2               CONVERT TO DECIMAL
         UNPK  DBL1(8),DBL2(8)       UNPACK
         OC    DBL1(8),=8X'FØ'       MAKE SURE IT'S READABLE
         MVC   ALCELEN(8),DBL1       COPY LENGTH
CALLOC4Ø EQU   *
*--------------------------------------------------------------------*
*   Loop until we find the 'Time' record for the calloc() return.    *
*--------------------------------------------------------------------*
         GET   INPUT,(R8)            GET INPUT DATA
         CLC   INREC+22(4),=C'Time'  AN EVENT 'Time' RECORD?
         BNE   CALLOC4Ø              NO - LOOP FOR 'Time' RECORD
         MVC   TIMESAVE(15),INREC+27 SAVE THE TIME
         MVC   DATESAVE(1Ø),INREC+51 SAVE THE DATE
CALLOC5Ø EQU   *
*--------------------------------------------------------------------*
*   Loop until we find the calloc() completion record.               *
*--------------------------------------------------------------------*
         GET   INPUT,(R8)            GET INPUT DATA
         CLC   INREC+97(8),=C'<--(122)' EVENT COMPLETION RECORD?
         BNE   CALLOC5Ø              NO - GET NEXT RECORD
*--------------------------------------------------------------------*
*   Determine whether the request was a success.  If it is, capture  *
*   the storage address in the allocation entry control block.  If it *
*   is not a success (R15=Ø), release the allocation entry control   *
*   block storage and check for more events.                         *
*--------------------------------------------------------------------*
```

```
         CLC   INREC+110(8),=8C'0'   R15=00000000?
         BE    CALLOC60              YES - ALLOCATE FAILED
         MVC   ALCETIME(15),TIMESAVE COPY TIME TO ALLOC ENTRY
         MVC   ALCEDATE(10),DATESAVE COPY DATE TO ALLOC ENTRY
         MVC   ALCEADDR(8),INREC+110 COPY ADDRESS TO ALLOC ENTRY
         MVC   ALCENXT(4),ALLOCPTR   COPY CHAIN POINTER TO NEXT
         ST    R2,ALLOCPTR           SAVE NEW CHAIN START
         B     EVENTLP               GET NEXT RECORD
CALLOC60 EQU   *
         STORAGE RELEASE,LENGTH=ALCELN,ADDR=(R2)
         B     EVENTLP               GET NEXT RECORD
*-------------------------------------------------------------------*
MALLOC   EQU   *
*-------------------------------------------------------------------*
*   It's a malloc() event record.  Acquire some storage for an      *
*   allocation entry control block.                                 *
*-------------------------------------------------------------------*
         STORAGE OBTAIN,LENGTH=ALCELN,LOC=ANY
         ST    R1,CURRENT            SAVE ENTRY ADDRESS
         LR    R2,R1                 COPY IT
         MVC   ALCENXT(4),ALLOCPTR   COPY PREVIOUS ENTRY ADDRESS
         MVC   ALCETYPE(8),=C'malloc  ' SET ENTRY TYPE
         XR    R6,R6                 CLEAR R6
         XR    R7,R7                 CLEAR R7
         LA    R15,INREC+113         POINT TO SIZE AREA
MALLOC20 EQU   *
*-------------------------------------------------------------------*
*   Capture the malloc() size information.                          *
*-------------------------------------------------------------------*
         CLI   0(R15),C')'           SIZE TERMINATOR?
         BE    MALLOC30              YES - GOT SIZE
         IC    R6,0(,R15)            GET DIGIT FROM SIZE AREA
         N     R6,=X'0000000F'       TURN OFF ALL BUT LOW ORDER NIBBLE
         MH    R7,=H'10'             MULTIPLY EXISTING VALUE BY 10
         AR    R7,R6                 ADD IN NEW
         LA    R15,1(,R15)           POINT TO NEXT BYTE
         B     MALLOC20              CHECK FOR MORE SIZE DIGITS
MALLOC30 EQU   *
*-------------------------------------------------------------------*
*   Save the size of the calloc() request.                          *
*-------------------------------------------------------------------*
         CVD   R7,DBL2               CONVERT TO DECIMAL
         UNPK  DBL1(8),DBL2(8)       UNPACK
         OC    DBL1(8),=8X'F0'       MAKE SURE IT'S READABLE
         MVC   ALCELEN(8),DBL1       COPY LENGTH
MALLOC40 EQU   *
*-------------------------------------------------------------------*
*   Loop until we find the 'Time' record for the malloc() return.   *
*-------------------------------------------------------------------*
         GET   INPUT,(R8)            GET INPUT DATA
```

```
         CLC   INREC+22(4),=C'Time'  AN EVENT 'Time' RECORD?
         BNE   MALLOC4Ø              NO - LOOP FOR 'Time' RECORD
         MVC   TIMESAVE(15),INREC+27 SAVE THE TIME
         MVC   DATESAVE(1Ø),INREC+51 SAVE THE DATE
MALLOC5Ø EQU   *
*-------------------------------------------------------------------*
*   Loop until we find the malloc() completion record.              *
*-------------------------------------------------------------------*
         GET   INPUT,(R8)            GET INPUT DATA
         CLC   INREC+97(8),=C'<--(124)' EVENT COMPLETION RECORD?
         BNE   MALLOC5Ø              NO - GET NEXT RECORD
*-------------------------------------------------------------------*
*   Determine whether the request was a success.  If it is, capture *
*   the storage address in the allocation entry control block.  If it *
*   is not a success (R15=Ø), release the allocation entry control  *
*   block storage and check for more events.                        *
*-------------------------------------------------------------------*
         CLC   INREC+11Ø(8),=8C'Ø'   R15=ØØØØØØØØ?
         BE    MALLOC6Ø              YES - ALLOCATE FAILED
         MVC   ALCETIME(15),TIMESAVE COPY TIME TO ALLOC ENTRY
         MVC   ALCEDATE(1Ø),DATESAVE COPY DATE TO ALLOC ENTRY
         MVC   ALCEADDR(8),INREC+11Ø COPY ADDRESS TO ALLOC ENTRY
         MVC   ALCENXT(4),ALLOCPTR   COPY CHAIN POINTER TO NEXT
         ST    R2,ALLOCPTR           SAVE NEW CHAIN START
         B     EVENTLP               GET NEXT RECORD
MALLOC6Ø EQU   *
         STORAGE RELEASE,LENGTH=ALCELN,ADDR=(R2)
         B     EVENTLP               GET NEXT RECORD
*-------------------------------------------------------------------*
REALLOC  EQU   *
*-------------------------------------------------------------------*
*   It's a realloc() event record.  Find the previous allocation    *
*   entry control block and free it up.                             *
*-------------------------------------------------------------------*
         LA    R15,ALLOCPTR          GET CHAIN PTR UPDATE TARGET ADDR
         L     R2,ALLOCPTR           GET STARTING ENTRY ADDR
REALLC1Ø EQU   *
         LTR   R2,R2                 END OF CHAIN?
         BZ    REALLC6Ø              YES - ORPHANED FREE TRACE ENTRY
         CLC   ALCEADDR(8),INREC+116 ADDRESS MATCH?
         BE    REALLC2Ø              YES - DO POINTER SWITCH
         LR    R15,R2                SAVE PREVIOUS UPDATE TARGET ADDR
         L     R2,ALCENXT            GET ADDR OF NEXT ENTRY
         B     REALLC1Ø              CHECK NEXT ENTRY
REALLC2Ø EQU   *
         MVC   Ø(4,R15),ALCENXT      COPY NEXT FROM CURRENT TO PREV
         STORAGE RELEASE,LENGTH=ALCELN,ADDR=(R2)
         B     REALLC7Ø              PROCESS ALLOC COMPONENT
REALLC6Ø EQU   *
         MVC   OUTREC(8Ø),OUTRECØ3   COPY OUTPUT RECORD MODEL
```

```
         MVC   OUTREC+3Ø(4),=C'Time' MOVE 'Time' INTO MESSAGE
         MVC   OUTREC+35(15),TIMESAVE MOVE TIME INTO MESSAGE
         MVC   OUTREC+51(4),=C'Date' MOVE 'Date' INTO MESSAGE
         MVC   OUTREC+56(1Ø),DATESAVE MOVE DATE INTO MESSAGE
         PUT   OUTPUT,OUTREC         WRITE OUT MESSAGE
         OI    FLAG,FREEBLK          SET UNPAIRED FREE BLOCK FLAG
REALLC7Ø EQU   *
*-------------------------------------------------------------------*
*   Allocate a new allocation entry control block.                  *
*-------------------------------------------------------------------*
         STORAGE OBTAIN,LENGTH=ALCELN,LOC=ANY
         ST    R1,CURRENT            SAVE ENTRY ADDRESS
         LR    R2,R1                 COPY IT
         MVC   ALCENXT(4),ALLOCPTR   COPY PREVIOUS ENTRY ADDRESS
         MVC   ALCETYPE(8),=C'realloc ' SET ENTRY TYPE
         XR    R6,R6                 CLEAR R6
         XR    R7,R7                 CLEAR R7
         LA    R15,INREC+125         POINT TO SIZE AREA
REALLC75 EQU   *
*-------------------------------------------------------------------*
*   Capture the malloc() size information.                          *
*-------------------------------------------------------------------*
         CLI   Ø(R15),C')'           SIZE TERMINATOR?
         BE    REALLC85              YES - GOT SIZE
         CLI   Ø(R15),C']'           END OF INPUT LINE?
         BE    REALLC8Ø              YES - GOT SIZE
         IC    R6,Ø(,R15)            GET DIGIT FROM SIZE AREA
         N     R6,=X'ØØØØØØØF'       TURN OFF ALL BUT LOW ORDER NIBBLE
         MH    R7,=H'1Ø'             MULTIPLY EXISTING VALUE BY 1Ø
         AR    R7,R6                 ADD IN NEW
         LA    R15,1(,R15)           POINT TO NEXT BYTE
         B     REALLC75              CHECK FOR MORE SIZE DIGITS
REALLC8Ø EQU   *
         GET   INPUT,(R8)            GET INPUT DATA
         CLI   INREC+22,C'+'         NEXT DATA LINE?
         BNE   REALLC8Ø              NO - TRY AGAIN
         LA    R15,INREC+97          POINT TO SIZE AREA CONTINUATION
         B     REALLC75              CHECK FOR MORE SIZE DIGITS
REALLC85 EQU   *
*-------------------------------------------------------------------*
*   Save the size of the realloc() request.                         *
*-------------------------------------------------------------------*
         CVD   R7,DBL2               CONVERT TO DECIMAL
         UNPK  DBL1(8),DBL2(8)       UNPACK
         OC    DBL1(8),=8X'FØ'       MAKE SURE IT'S READABLE
         MVC   ALCELEN(8),DBL1       COPY LENGTH
REALLC9Ø EQU   *
*-------------------------------------------------------------------*
*   Loop until we find the 'Time' record for the realloc() return.  *
*-------------------------------------------------------------------*
```

```
         GET   INPUT,(R8)            GET INPUT DATA
         CLC   INREC+22(4),=C'Time'  AN EVENT 'Time' RECORD?
         BNE   REALLC9Ø              NO - LOOP FOR 'Time' RECORD
         MVC   TIMESAVE(15),INREC+27 SAVE THE TIME
         MVC   DATESAVE(1Ø),INREC+51 SAVE THE DATE
REALLC95 EQU   *
*--------------------------------------------------------------------*
*   Loop until we find the realloc() completion record.              *
*--------------------------------------------------------------------*
         GET   INPUT,(R8)            GET INPUT DATA
         CLC   INREC+97(8),=C'<--(125)' EVENT COMPLETION RECORD?
         BNE   REALLC95              NO - GET NEXT RECORD
*--------------------------------------------------------------------*
*   Determine whether the request was a success.  If it is, capture  *
*   the storage address in the allocation entry control block.  If it *
*   is not a success (R15=Ø), release the allocation entry control   *
*   block storage and check for more events.                         *
*--------------------------------------------------------------------*
         CLC   INREC+11Ø(8),=8C'Ø'   R15=ØØØØØØØØ?
         BE    REALLC1ØØ             YES - ALLOCATE FAILED
         MVC   ALCETIME(15),TIMESAVE COPY TIME TO ALLOC ENTRY
         MVC   ALCEDATE(1Ø),DATESAVE COPY DATE TO ALLOC ENTRY
         MVC   ALCEADDR(8),INREC+11Ø COPY ADDRESS TO ALLOC ENTRY
         MVC   ALCENXT(4),ALLOCPTR   COPY CHAIN POINTER TO NEXT
         ST    R2,ALLOCPTR           SAVE NEW CHAIN START
         B     EVENTLP               GET NEXT RECORD
REALLC1ØØ EQU   *
         STORAGE RELEASE,LENGTH=ALCELN,ADDR=(R2)
         B     EVENTLP               GET NEXT RECORD
*--------------------------------------------------------------------*
FREE     EQU   *
*--------------------------------------------------------------------*
*   It's a free() event record.  Attempt to find a matching          *
*   allocation entry control block and free it up.                   *
*--------------------------------------------------------------------*
         LA    R15,ALLOCPTR          GET CHAIN PTR UPDATE TARGET ADDR
         L     R2,ALLOCPTR           GET STARTING ENTRY ADDR
FREE1Ø   EQU   *
         LTR   R2,R2                 END OF CHAIN?
         BZ    FREE6Ø                YES - ORPHANED FREE TRACE ENTRY
         CLC   ALCEADDR(8),INREC+113 ADDRESS MATCH?
         BE    FREE2Ø                YES - DO POINTER SWITCH
         LR    R15,R2                SAVE PREVIOUS UPDATE TARGET ADDR
         L     R2,ALCENXT            GET ADDR OF NEXT ENTRY
         B     FREE1Ø                CHECK NEXT ENTRY
FREE2Ø   EQU   *
*--------------------------------------------------------------------*
*   A matching allocation entry control block has been located.  Free *
*   it up.                                                           *
*--------------------------------------------------------------------*
```

```
         MVC   Ø(4,R15),ALCENXT      COPY NEXT FROM CURRENT TO PREV
         STORAGE RELEASE,LENGTH=ALCELN,ADDR=(R2)
         B     EVENTLP               GET NEXT RECORD
FREE6Ø   EQU   *
*--------------------------------------------------------------------*
*   A free() trace entry has been located for a block of storage that *
*   does not have a corresponding calloc(), malloc(), or realloc()    *
*   trace entry.  Issue an orphaned free() trace entry message.       *
*--------------------------------------------------------------------*
         MVC   OUTREC(8Ø),OUTRECØ3   COPY OUTPUT RECORD MODEL
         MVC   OUTREC+3Ø(4),=C'Time' MOVE 'Time' INTO MESSAGE
         MVC   OUTREC+35(15),TIMESAVE MOVE TIME INTO MESSAGE
         MVC   OUTREC+51(4),=C'Date' MOVE 'Date' INTO MESSAGE
         MVC   OUTREC+56(1Ø),DATESAVE MOVE DATE INTO MESSAGE
         PUT   OUTPUT,OUTREC         WRITE OUT MESSAGE
         OI    FLAG,FREEBLK          SET UNPAIRED FREE BLOCK FLAG
         B     EVENTLP               GET NEXT RECORD
**********************************************************************
ALLDONE  EQU   *
         CLC   ALLOCPTR(4),=F'Ø'     ANY EXISTING STORAGE REFERENCES?
         BNE   MEMLEAK               YES - MAY INDICATE MEMORY LEAK
         PUT   OUTPUT,OUTRECØ1       WRITE MESSAGE
         TM    FLAG,FREEBLK          UNPAIRED FREE BLOCK ENCOUNTERED?
         BO    RETURNØ4              YES - RETURN RC=4
         B     RETURN                WE'RE DONE
**********************************************************************
MEMLEAK  EQU   *
*--------------------------------------------------------------------*
*   At this point, we have determined that storage allocation        *
*   requests in the trace output data don't all have corresponding   *
*   free requests.                                                   *
*                                                                    *
*   Produce an output record for each entry on the chain.            *
*--------------------------------------------------------------------*
         PUT   OUTPUT,OUTRECØ2       WRITE MESSAGE
*--------------------------------------------------------------------*
*   Process the chain from end to start so that the output records   *
*   are produced in timestamp ascending sequence.                    *
*--------------------------------------------------------------------*
LEAK     EQU   *
         L     R2,ALLOCPTR           GET STARTING ENTRY ADDR
         LA    R5,ALLOCPTR           SAVE POINTER ADDR
LEAKLP   EQU   *
         CLC   ALCENXT,=F'Ø'         LAST ENTRY?
         BE    LASTENT               YES - PROCESS LAST ENTRY
         LR    R5,R2                 SAVE PREV ADDR
         L     R2,ALCENXT            POINT TO NEXT ENTRY
         B     LEAKLP                CHECK IT OUT
LASTENT  EQU   *
         XC    Ø(4,R5),Ø(R5)         ZERO ALCENXT PTR IN PREV ENTRY
```

```
         MVC   OUTREC(8Ø),OUTRECØ4   MOVE IN MESSAGE MODEL
         MVC   OUTREC+5(8),ALCEADDR  COPY IN THE ADDRESS
         MVC   OUTREC+18(8),ALCELEN  COPY IN THE LENGTH
         MVC   OUTREC+47(15),ALCETIME COPY IN THE TIME
         MVC   OUTREC+68(1Ø),ALCEDATE COPY IN THE DATE
         PUT   OUTPUT,OUTREC         WRITE THE MESSAGE
         STORAGE RELEASE,LENGTH=ALCELN,ADDR=(R2)
         CLC   ALLOCPTR(4),=F'Ø'     LAST ENTRY?
         BE    RETURNØ4              YES - WE'RE DONE
         B     LEAK                  TRY ANOTHER TIME
*********************************************************************
RETURN   EQU   *
         CLOSE (INPUT),MODE=31       CLOSE INPUT DATASET
         CLOSE (OUTPUT),MODE=31      CLOSE OUTPUT DATASET
         LR    R1,R13                COPY STORAGE ADDRESS
         L     R3,4(,R1)             GET RETURN SAVEAREA ADDRESS
         STORAGE RELEASE,LENGTH=WORKLEN,ADDR=(R1)
         LR    R13,R3                RESTORE RETURN SAVEAREA ADDRESS
         LM    R14,R12,12(R13)       RESTORE THE REGISTERS
         XR    R15,R15               SET RETURN CODE
         BR    R14                   RETURN
*********************************************************************
RETURNØ4 EQU   *
         CLOSE (INPUT),MODE=31       CLOSE INPUT DATASET
         CLOSE (OUTPUT),MODE=31      CLOSE OUTPUT DATASET
         LR    R1,R13                COPY STORAGE ADDRESS
         L     R3,4(,R1)             GET RETURN SAVEAREA ADDRESS
         STORAGE RELEASE,LENGTH=WORKLEN,ADDR=(R1)
         LR    R13,R3                RESTORE RETURN SAVEAREA ADDRESS
         LM    R14,R12,12(R13)       RESTORE THE REGISTERS
         LA    R15,4                 SET RETURN CODE
         BR    R14                   RETURN
*********************************************************************
INPUT    DCB   MACRF=(GM),DDNAME=INPUT,DSORG=PS,EODAD=ALLDONE
OUTPUT   DCB   MACRF=(PM),DDNAME=OUTPUT,LRECL=8Ø,DSORG=PS
*********************************************************************
OUTRECØ1 DC    CL8Ø'ALL ALLOCATED STORAGE HAS BEEN RELEASED'
OUTRECØ2 DC    CL8Ø'POTENTIAL MEMORY LEAK DETECTED'
OUTRECØ3 DC    CL8Ø'Orphaned trace free() entry:'
OUTRECØ4 DC    C'ADDR XXXXXXXX LEN(xxxxxxxx) not released.  Tm: xx.xx.'
         DC    CL(8Ø-L'OUTRECØ4)'xx.xxxxxx  Dt: xxxx.xx.xx'
*********************************************************************
WORKAREA DSECT
SAVEAREA DS    18F
INPARM   DS    F
ALLOCPTR DS    F
CURRENT  DS    F
DBL1     DS    2D
DBL2     DS    2D
INREC    DS    CL133
```

```
OUTREC   DS    CL8Ø
TIMESAVE DS    CL16
DATESAVE DS    CL1Ø
FLAG     DS    CL1
FREEBLK  EQU   X'8Ø'
WORKLEN  EQU   *-WORKAREA
ALCE     DSECT                     ALLOCATION ENTRY
ALCENXT  DS    F                   ADDR OF NEXT ENTRY
ALCETIME DS    CL16                TRACE ENTRY TIME
ALCEDATE DS    CL1Ø                TRACE ENTRY DATE
ALCEADDR DS    CL8                 TRACE ENTRY STARTING STORAGE ADDR
ALCELEN  DS    CL8                 TRACE ENTRY LENGTH
ALCETYPE DS    CL8                 TRACE ENTRY TYPE c/m/realloc
ALCERSV1 DS    CL2                 RESERVED
ALCELN   EQU   *-ALCE
         $REQU
         END
```

## ALLOCFRE C

```
#define MVS
/*
 * Set up the run-time options to capture storage allocation
 * trace records.
 */
#pragma  runopts("TRACE(ON,16384K,DUMP,LE=8)")
#pragma  runopts("RPTSTG(ON)")
#include <stdio.h>
#include <string.h>
#include <tcperrno.h>
#include <errno.h>
main()
{
  unsigned char* buf = NULL;
  int i = 0;
// Perform a free() that should fail (free() trace record R15 != 0).
  buf = (char *)(123456);
  free(buf);
  buf = NULL;
// Acquire an initial block of storage with malloc().
  buf = (unsigned char*)malloc(128000);
  if (buf != NULL)
  {
    printf("Storage addr is %x\n",buf);
  }
  while (buf != NULL)
  {
// On alternate passes through the loop, issue a malloc() or calloc().
    if (i == 0)
```

```
    {
      buf = (unsigned char*)malloc(128000);
    }
    else
    {
      buf = (unsigned char*)calloc(40,4000);
    }
    if (errno == ENOMEM || errno == 132)
    {
      printf("No memory available\n");
    }
    else if (buf != NULL)
    {
      printf("Storage addr is %x\n",buf);
      if (i == 0)
      {
// On every second pass through the loop, free the acquired storage.
        free(buf);
        i = 1;
      }
      else
      {
        i = 0;
      }
    }
  }
  return(0);
}
```

*Rudy Douglas*
*System Programmer (Canada)*



## User-written ISPF-based VSAM browse utility

Are you tired of seeing the following message saying that you have no VSAM support?

```
 Menu  Options  View  Utilities  Compilers  Help


DSLIST - Data Sets Matching CICSTS22.**.IV$FILE VSAM processing unavail.
Command ===>                                            Scroll ===> PAGE

Command - Enter "/" to select action            Message           Volume
```

```
------------------------------------------------------------------------
B         CICSTS22.BETA.GIIR200.IV$FILE                          *VSAM*
          CICSTS22.BETA.GIIR200.IV$FILE.DATA                     CICSØ1
          CICSTS22.BETA.GIIR200.IV$FILE.INDEX                    CICSØ1
          CICSTS22.B1CICS.GIIR200.IV$FILE                        *VSAM*
          CICSTS22.B1CICS.GIIR200.IV$FILE.DATA                   CICSØ2
          CICSTS22.B1CICS.GIIR200.IV$FILE.INDEX                  CICSØ2
          CICSTS22.DDCICS.GIIR200.IV$FILE                        *VSAM*
          CICSTS22.DDCICS.GIIR200.IV$FILE.DATA                   CICSØ6+
          CICSTS22.DDCICS.GIIR200.IV$FILE.INDEX                  CICSØ6+
          CICSTS22.D2CICS.GIIR200.IV$FILE                        *VSAM*
          CICSTS22.D2CICS.GIIR200.IV$FILE.DATA                   CICSØ3
          CICSTS22.D2CICS.GIIR200.IV$FILE.INDEX                  CICSØ3
          CICSTS22.D7CICS.GIIR200.IV$FILE                        *VSAM*
          CICSTS22.D7CICS.GIIR200.IV$FILE.DATA                   CICSØ3
                                                                      Ø3
  CICSTS22.BETA.GIIR200.IV$FILE is a VSAM data set. An attempt was    M*
  made to invoke a VSAM editor, viewer, or browser, but it is not     Ø4
  allowed due to configuration table settings                         Ø4
                                                                      M*
          CICSTS22.GIIR120.IV$FILE.BACKUP                        *VSAM*
```

Try this user-written ISPF-based VSAM browse utility.

## PROGRAM

```
*====================================================================
*          - BHB - TSO - BROWSE OF VSAM DATASETS.
*====================================================================
VSAMBR   CSECT
VSAMBR   AMODE 31               SET AMODE
         PRINT NOGEN
         SAVE  (14,12)          SAVE REGISTERS
         BALR  12,Ø             DEFINE BASE REGISTER
         USING *,12
         ST    13,SAVE1+4       STORE CALLING ROUTINE SAVE AREA ADDRESS
         LA    15,SAVE1         DEFINE NEW SAVE AREA
         ST    15,8(13)         SAVE NEW SAVE AREA ADDRESS
         LR    13,15            POINT TO NEW SAVE AREA
         ST    12,DDATA         SAVE BASE REGISTER IN DIALOG DATA AREA
VDEFVGET EQU   *
          CALL  ISPLINK,(VDEF,DSNDDNV,DSNDDNL,FORMATL,LENL,LIST),VL
         CALL  ISPLINK,(VGET,DSNDDNV),VL
         MODCB ACB=ACB1,DDNAME=(*,DDNAME)
OPEN     EQU   *
         OPEN  ACB1             OPEN VSAM DATASET
         LTR   15,15            OPEN ERROR?
         BZ    SHOWCB
          SHOWCB ACB=ACB1,AREA=ERROR,LENGTH=4,FIELDS=ERROR
```

```
*********************************************************************
         LA    3,116
         C     3,ERROR          VERIFY NEEDED?
         BE    VERIFY
         LA    3,16Ø
         C     3,ERROR          EMPTY DATASET?
         BE    EMPTY
         B     STMSGOPN
VERIFY   EQU   *
         VERIFY RPL=RPLVER,ACTION=REFRESH
         LA    3,1
         X     3,OPNAGAIN
         ST    3,OPNAGAIN
         BNZ   CLOSE
STMSGOPN EQU   *
         CALL  ISPLINK,(SETMSG,MSGOPN,COND),VL
         B     CLOSE
SHOWCB   EQU   *
         SHOWCB ACB=ACB1,AREA=NLOGR,LENGTH=4,FIELDS=NLOGR
         L     3,NLOGR
         LTR   3,3              EMPTY DATASET?
         BNZ   CALLBRIF
EMPTY    EQU   *
         CALL  ISPLINK,(SETMSG,MSGEMPTY,COND),VL
         B     CLOSE
CALLBRIF EQU   *
         MODCB RPL=RETRVE,ARG=LOWRBA,OPTCD=(ADR,DIR)
         POINT RPL=RETRVE
         MODCB RPL=RETRVE,OPTCD=(SEQ,FWD)
         CALL  ISPLINK,(BRIF,DSNAME,RECFM,RECL,RADDR,,DDATA),VL
CLOSE    EQU   *
         CLOSE ACB1             CLOSE VSAM DATASET
         L     3,OPNAGAIN
         LTR   3,3
         BNZ   OPEN
RTRN     EQU   *
         L     13,SAVE1+4       RESTORE CALLING ROUTINE SAVE AREA
         RETURN (14,12),RC=Ø    END VSAM DATASET BROWSING
*====================================================================
*        READ ROUTINE
*====================================================================
READRTN  EQU   *
         SAVE  (14,12)
         LM    4,7,Ø(1)
         L     12,Ø(7)          ESTABLISH ADDRESSABILITY
         ST    13,RSAVE+4       STORE BRIF ROUTINE SAVE AREA ADDRESS
         LA    15,RSAVE
         ST    15,8(13)         SAVE NEW SAVE AREA ADDRESS
         LR    13,15            POINT TO NEW SAVE AREA
*
```

```
         L     3,FIRSTENT       FIRST ENTRY?
         LTR   3,3
         BNZ   NOTFIRST
         LA    3,1
         ST    3,FIRSTFLG
         ST    3,FIRSTENT
NOTFIRST EQU   *
         LA    8,AREA1
         ST    8,Ø(4)           STORE RECORD ADDRESS BACK TO BRIF
         ST    6,@RRNO          SAVE ADDRESS OF RRNO
         L     6,Ø(6)           LOAD RRNO
         L     11,LASTRRNO
         CR    6,11             COMPARE
         BH    GET1             BRANCH LESS OR EQUAL TO POINT
POINT    EQU   *
         MODCB RPL=RETRVE,ARG=LOWRBA,OPTCD=(ADR,DIR)
         POINT RPL=RETRVE
         MODCB RPL=RETRVE,OPTCD=(SEQ,FWD)
         LA    11,Ø             SET COUNTER = Ø
GET1     GET   RPL=RETRVE       READ RECORD
         SHOWCB RPL=RETRVE,AREA=ALENGTH,LENGTH=4,FIELDS=(RECLEN)
         L     3,ALENGTH
         ST    3,Ø(5)           STORE RECORD LENGTH BACK TO BRIF
         LA    11,1(11)         ADD 1 TO COUNTER
         CR    11,6             LAST RRNO READ
         BNE   GET1             NO, GO TO GET1
GOBACK   ST    6,LASTRRNO       STORE LAST RRNO READ
         LA    1Ø,Ø             SET RETURN CODE = Ø
         B     TOBRIF
EODAD1   EQU   *
         L     3,FIRSTFLG
         LTR   3,3
         BZ    SETRC8
         CALL  ISPLINK,(SETMSG,MSGEMPTY,COND),VL
SETRC8   EQU   *
         ST    11,LASTRRNO      STORE LAST RRNO OF LAST READ RECORD
         L     6,@RRNO          LOAD RRNO ADRESS
         ST    11,Ø(6)          STORE RRNO BACK TO BRIF
         LA    1Ø,8             SET RETURN CODE = 8
         B     TOBRIF           RETURN TO BRIF
TOBRIF   EQU   *
         LA    3,Ø
         ST    3,FIRSTFLG
         L     13,RSAVE+4       RESTORE BRIF ROUTINE SAVE AREA ADDRESS
         L     14,12(13)
         LR    15,1Ø            LOAD RETURN CODE
         LM    Ø,12,2Ø(13)
         BR    14               RETURN TO BRIF
*=================================================================
*        VSAM CONTROL BLOCKS
```

```
*=====================================================================
ACB1     ACB   EXLST=EXLST1,                                          X
               MACRF=(ADR,SEQ,IN)
EXLST1   EXLST EODAD=(EODAD1,A)
RETRVE   RPL   ACB=ACB1,                                             X
               AREA=AREA1,                                           X
               AREALEN=9000,                                         X
               OPTCD=SEQ
RPLVER   RPL   ACB=ACB1,                                             X
               AREA=AREA1,                                           X
               AREALEN=9000,                                         X
               OPTCD=CNV
*=====================================================================
*        WORK AREAS AND CONSTANTS
*=====================================================================
         LTORG
SAVE1    DC    18F'0'           REGISTER SAVE AREA
RSAVE    DC    18F'0'           REGISTER SAVE AREA
*
VDEF     DC    CL8'VDEFINE '
VGET     DC    CL8'VGET    '
DSNDDNV  DC    CL15'(DSVSAM DDVSAM)'
DSNDDNL  DS    0CL52
DSNAME   DS    CL44' '
DDNAME   DS    CL8' '
FORMATL  DS    0CL16
         DC    CL8'CHAR    '
         DC    CL8'CHAR    '
LENL     DS    0F
DSLEN    DC    F'44'
DDLEN    DC    F'8'
LIST     DC    CL8'LIST    '
*
SETMSG   DC    CL8'SETMSG  '
MSGOPN   DC    CL8'VSAM005 '
MSGEMPTY DC    CL8'VSAM004 '
COND     DC    CL8'COND    '
*
BRIF     DC    CL8'BRIF    '
RECFM    DC    CL2'V '
RECL     DC    F'9000'
RADDR    DC    AL4(READRTN)
DDATA    DC    F'0'
*
ALENGTH  DC    F'0'
RRNO     DC    F'0'
@RRNO    DC    F'0'
LASTRRNO DC    F'0'
LOWRBA   DC    F'0'
NLOGR    DC    F'0'
```

```
ERROR    DC    F'0'
OPNAGAIN DC    F'0'
FIRSTENT DC    F'0'
FIRSTFLG DC    F'0'
AREA1    DC    9000F'0'  /* CAN BE ADJUSTED BASED ON RECORD SIZE */
*
         END   VSAMBR
```

## JCL

```
//E411BHBA JOB  (DQMVST,T),BHB.ASSEMBLE.H,MSGCLASS=T,CLASS=A,
//            NOTIFY=E404BHB
/*JOBPARM  ROOM=E404
//* **
//ASMHCLG PROC SYSPRT='*',SYSPUN=Z
//C      EXEC  PGM=IEV90,PARM=(OBJECT),REGION=2048K
//SYSLIB   DD  DSN=SYS1.AMODGEN,DISP=SHR,DCB=BLKSIZE=4080
//         DD  DSN=SYS1.MACLIB,DISP=SHR,DCB=BLKSIZE=4080
//SYSUT1   DD  UNIT=(SYSDA,SEP=SYSLIB),SPACE=(CYL,(10,5)),DSN=&SYSUT1
//SYSPUNCH DD  SYSOUT=&SYSPUN,DCB=(BLKSIZE=800),
//             SPACE=(CYL,(5,5,0))
//SYSPRINT DD SYSOUT=&SYSPRT,DCB=(BLKSIZE=3509),
//             UNIT=(,SEP=(SYSUT1,SYSPUNCH))
//SYSLIN   DD  DISP=(,PASS),UNIT=SYSDA,SPACE=(CYL,(5,5,0)),            *
//             DCB=(BLKSIZE=400),DSN=&&LOADSET
//L    EXEC  PGM=IEWL,PARM='MAP,LET,LIST,NCAL,AC=1',
//             COND=(8,LT,C),REGION=2048K
//SYSLIN  DD   DSN=&&LOADSET,DISP=(OLD,DELETE)
//        DD   DDNAME=SYSIN
//SYSLMOD DD DISP=(,PASS),UNIT=SYSDA,SPACE=(CYL,(2,1,2)),DSN=&GOSET(GO)
//SYSLIB  DD DISP=SHR,DSN=SYS1.ISP.SISPLOAD  <== YOUR ISP LIBRARY
//        DD DISP=SHR,DSN=SYS1.LINKLIB
//SYSUT1  DD   UNIT=SYSDA,SPACE=(CYL,(3,2)),DSN=&SYSUT1
//SYSPRINT DD  SYSOUT=&SYSPRT,DCB=(RECFM=FB,BLKSIZE=3509)
//         PEND
//STEP1    EXEC PROC=ASMHCLG
//C.SYSIN  DD  DISP=SHR,DSN=E411BHB.SPF.SOURCE(VSAMBR) <= SOURCE
//L.SYSLMOD DD DSN=E411BHB.TESTLIB(VSAMBR),DISP=SHR <= LINK LIST LIBRARY
//L.SYSIN  DD  *
 INCLUDE SYSLIB(ISPLINK)
 NAME VSAMBR(R)
/*
```

## CLIST

```
  PROC 0
 /*-----------------------------------------------------------------*
 /*
```

```
 /*    ALLOCATE A VSAM DATASET FOR BROWSE PURPOSE  BHB
 /*
  /*-------------------------------------------------------------------*
    CONTROL MAIN NOFLUSH
START:   +
    ISPEXEC DISPLAY PANEL(BRIFPAN)             /* GET DATASET NAME
    IF &LASTCC GT Ø THEN EXIT
  /*-------------------------------------------------------------------*
 /* CHECK DATASET
  /*-------------------------------------------------------------------*
    SET MSGCONT = &STR(&SYSDSN('&DSVSAM'))  /* DATASET AVAILABLE?
    IF &MSGCONT = OK THEN +
      DO                                    /* YES
       LISTDSI '&DSVSAM' DIRECTORY          /* VSAM?
       IF &SYSDSORG = VS THEN +
       SYSCALL ALLOCDS &DSVSAM              /* YES: ALLOC
       ELSE ISPEXEC SETMSG MSG(VSAMØØØ)               /*  NOPE
      END
    ELSE ISPEXEC SETMSG MSG(VSAMØØ1)       /* NO:  SET ERROR MSG
    GOTO START
  /*-------------------------------------------------------------------*
 /* ALLOCATE DATASET
  /*-------------------------------------------------------------------*
ALLOCDS: PROC 1 DSVSAM
    SET TIME = &STR(&SUBSTR(1:2,&SYSTIME)&SUBSTR(4:5,&SYSTIME)+
               &SUBSTR(7:8,&SYSTIME))      /* CREATE DDNAME
    SET DDVSAM = DD&TIME                   /* FROM HHMMSS
    ALLOC DD(&DDVSAM) DA('&DSVSAM') REU SHR  /* ALLOC DISP=SHR
    IF &LASTCC = Ø THEN +
       DO
        ISPEXEC VPUT (DSVSAM,DDVSAM) SHARED  /* VPUT DSN AND DDN
        ISPEXEC SELECT PGM(BRIFASM)          /* CALL PRM TO BROWSE
        FREE DD(&DDVSAM)                     /* FREE DATASET
       END                                   /* SET ERROR
    ELSE ISPEXEC SETMSG MSG(VSAMØØ2)
    END
EXIT: EXIT
```

## ISPMLIB

```
VSAMØØØ .ALARM=YES
'DATASET SPECIFIED IS NOT A VSAM DATASET.'

VSAMØØ1 .ALARM=YES
'&MSGCONT.'

VSAMØØ2 .ALARM=YES
'&DSVSAM COULD NOT BE ALLOCATED.'
```

```
VSAMØØ3 .ALARM=YES
'INVALID SPECIFICATION IN A VSAM DATASET NAME.'

VSAMØØ4 .ALARM=YES
'EMPTY VSAM DATASET HAS BEEN REQUESTED.'

VSAMØØ5 .ALARM=YES
'OPEN OF DATASET ACB FAILED.'
```

## ISPPLIB

```
)ATTR  DEFAULT(%+_)
)BODY
%--------------------BROWSE VSAM DATASET -----------------------------
%COMMAND ===>_ZCMD                                                    +
%
%                                                    +USERID   - &ZUSER
%                                                    +TIME     - &ZTIME
%                                                    +TERMINAL - &ZTERM
%                                                    +PF KEYS  - &ZKEYS
%                                                    +APPLID   - &ZAPPLID
+VSAM DATASET:
+   PROJECT%===>_PRJV    +
+   GROUP  %===>_LIBV    +
+   TYPE   %===>_TYPV    +
+
+OTHER VSAM DATASET:
+   DATA SET NAME%===>_WDSN
+

)INIT
 &ZCMD = ' '
 IF (&WDSN ¬= ' ')  .CURSOR = WDSN
 .CURSOR = PRJV

)PROC
   VPUT (PRJV,LIBV,TYPV) PROFILE
   IF (&WDSN = ' ')
       &DSVSAM = '&PRJV..&LIBV..&TYPV'
       VER (&DSVSAM,DSNAME)
   ELSE
      &WDSN1 = TRUNC (&WDSN,'(')             /* IF MEMBER SPECIFIED
      IF (&WDSN1 ¬= &WDSN) .MSG=VSAMØØ3      /*  SET ERROR MSG
      ELSE
         &W = TRUNC (&WDSN,1)
         IF (&W = '''')
            &DSVSAM = .TRAIL                 /* IF WITH QUOTE
            &DSVSAM = TRUNC (&DSVSAM,'''')   /* GET RID OF IT
         ELSE                                /* IF NO QUOTE
```

```
           &DSVSAM = 'SYSUID..&WDSN'        /* ADD PREFIX
         VER (&DSVSAM,DSNAME)

)END
```

*Bruce H Bentley*
*Senior Systems Programmer*
*Federated Mutual Insurance (Canada)*


# Moving data between GDGs and HFS directories

As more applications are beginning to exploit USS functionality, I have found it necessary to have a way to move data between GDGs and HFS directories. This is the result of greater exploitation of native USS, and, in our shop, also because of the extensive exploitation of other Unix directories being NFS mounted on the mainframe.

I developed two programs to deal with this situation. The first program is called GDG2HFS. As the name implies, it is responsible for copying or moving all the entries in a GDG to a target directory on the USS side. The second program is called HFS2GDG. As its name implies, it is responsible for copying or moving all files in an HFS directory to new entries in a GDG. Both utilities are intended to be used in batch.

## GDG2HFS JCL

```
//jobcard…
//*********************************************************************
//* EXECUTE GDG2HFS - copy an entire GDG to an HFS directory          *
//*                                                                   *
//* GDG2HFS is the REXX EXEC                                          *
//*                                                                   *
//* First parm is the REXX EXEC name to IKJEFTØ1                      *
//* Second parm is the target directory                               *
//* Third parm is the temporary directory                             *
//* Fourth parm is the rename pattern (suffixed with a seq number)    *
//*                                                                   *
//* SYSEXEC points to the PDS that the REXX EXEC is uploaded into     *
```

```
//* INPUT points to the GDG Base name                                  *
//*                                                                    *
//********************************************************************
//GDG2HFS  EXEC PGM=IKJEFT01,
//         PARM='GDG2HFS /your/dir /your/dir/temp EXAMPLE'
//SYSEXEC  DD   DSN=your.EXEC,DISP=SHR
//SYSTSPRT DD   SYSOUT=*
//SYSTSIN  DD   DUMMY
//INPUT    DD   DSN=your.gdg,DISP=SHR
```

GDG2HFS reads the MVS catalog to acquire the GDG entries and uses OPUT to copy each file to the USS directory. It first copies each file to a temporary directory. This ensures the copy is possible. Then it MVs the file (USS MOVE command) to the target directory. This will find any permission errors you may encounter with the directory.

In the JCL:

1 SYSEXEC points to the PDS where you put GDG2HFS.

2 The INPUT DD in the JCL points to the source GDG to copy/move.

3 The second PARM is the target HFS directory for the files (usually the application directory).

4 The third PARM is a temporary directory to ensure that the copy can be done (usually a personal directory).

5 The fourth PARM is a name pattern to use for the target file on the HFS. This name will be suffixed with a sequential number. The default is TEST.

6 Remember USS is case sensitive, so use CAPS OFF in your ISPF session.

Like any JCL using GDGs, you can control whether the GDG is deleted in the JCL using the DISP keyword. I recommend using a subsequent step with adequate COND or IF statements to ensure that all entries were successfully copied before deleting your original GDG files.

Sample output from GDG2HFS looks like this:

```
YOUR.GDG is a valid GDG Base

YOUR.GDG.G0001V00 --> /your/dir/temp/YOUR.GDG.G0001V00
YOUR.GDG.G0001V00 --> /your/dir/EXAMPLE0001
YOUR.GDG.G0002V00 --> /your/dir/temp/YOUR.GDG.G0002V00
YOUR.GDG.G0002V00 --> /your/dir/EXAMPLE0002
YOUR.GDG.G0003V00 --> /your/dir/temp/YOUR.GDG.G0003V00
YOUR.GDG.G0003V00 --> /your/dir/EXAMPLE0003
YOUR.GDG.G0004V00 --> /your/dir/temp/YOUR.GDG.G0004V00
YOUR.GDG.G0004V00 --> /your/dir/EXAMPLE0004

4 GDG entries moved to /your/dir, RC=0
```

## GDG2HFS REXX EXEC

```
/*********************************************************************/
/*                               REXX                                */
/*********************************************************************/
/* Purpose: Push all GDSs from a GDG to HFS files                    */
/*-------------------------------------------------------------------*/
/* Syntax:  GDG2HFS targdir tempdir pattern                          */
/*-------------------------------------------------------------------*/
/* Parms: targdir    - Target directory                              */
/*        tempdir    - Temporary directory                           */
/*        pattern    - File name pattern for destination directory   */
/*                                                                   */
/* Notes: DDNAME INPUT must be allocated and point to the GDG Base   */
/*********************************************************************/
/*                       Change Log                                  */
/*********************************************************************/
/* Accept arguments and set pattern to default of 'TEST' if missing  */
/*********************************************************************/
parse arg targdir tempdir pattern
if targdir = '' then
    do
     EXITRC = 12
     msg = 'Target directory name is missing'
     signal shutdown
    end
if tempdir = '' then
    do
     EXITRC = 12
     msg = 'Temporary directory name is missing'
     signal shutdown
    end
if pattern = '' then pattern = 'TEST'
/*********************************************************************/
/* Find the INPUT DD for the GDG Base name                           */
/*********************************************************************/
```

```
LRC = listdsi('INPUT' 'FILE')
if LRC <> Ø then
    do
     EXITRC = 12
     msg = 'INPUT DD is missing'
     signal shutdown
    end
/*********************************************************************/
/* Parse the DSN in the DD to see whether it is a GDG                */
/*********************************************************************/
revdsn = reverse(sysdsname)
if pos('V',revdsn) = 3 & pos('G.',revdsn) = 8 then
    do
     parse var revdsn goovoo '.' revdsn
     gdgbase = reverse(revdsn)
     say gdgbase 'is a valid GDG Base'
     say
    end
else
    do
     EXITRC = 12
     msg = 'DSN in INPUT DD does not appear to be a GDG Base'
     signal shutdown
    end
/*********************************************************************/
/* Get DSN details from a LISTCAT                                    */
/*********************************************************************/
call outtrap 'listc.'
"LISTCAT ENT('"gdgbase"')"
EXITRC = RC
if EXITRC <> Ø then
    do
     msg = 'LISTCAT error on' gdgbase
     signal shutdown
    end
/*********************************************************************/
/* Check whether DSN is a GDG Base                                   */
/*********************************************************************/
if word(listc.1,1) <> 'GDG' then
    do
     EXITRC = 12
     msg = 'DSN on INPUT DD is not a GDG base'
     signal shutdown
    end
/*********************************************************************/
/* Load list of GDSs                                                 */
/*********************************************************************/
gdsnum = 1
do i=1 to listc.Ø
    parse var listc.i type . gdsdsn
```

```
    if type = 'NONVSAM' then
       do
        gds.gdsnum = gdsdsn
        gdsnum = gdsnum + 1
       end
    else
       iterate
end
/*********************************************************************/
/* Process all GDSs                                                  */
/*********************************************************************/
do i=1 to gdsnum-1
/*********************************************************************/
/* OPUT each GDS to the same name in the directory in BINARY         */
/*********************************************************************/
    tempname = tempdir'/'gds.i
   "OPUT '"gds.i"' '"tempname"' BINARY"
    EXITRC = RC
    if EXITRC <> 0 then
       do
        msg = 'OPUT error on' gds.i '-->' tempname
        signal shutdown
       end
    else
       do
        say gds.i '-->' tempname
/*********************************************************************/
/* MOVE the file to the target directory and rename using a pattern  */
/*********************************************************************/
        newname = targdir'/'pattern||right(i,4,0)
       "BPXBATCH SH mv" tempname newname
        EXITRC = RC / 256
        if EXITRC <> 0 then
           do
            msg = 'BPXBATCH error during MV' tempname '-->' newname
            signal shutdown
           end
        else
           do
            say gds.i '-->' newname
           end
       end
end
/*********************************************************************/
/* Completion message                                                */
/*********************************************************************/
msg = gdsnum-1 'GDG entries moved to' targdir
/*********************************************************************/
/* Shutdown                                                          */
/*********************************************************************/
```

```
shutdown: say
           say msg', RC='EXITRC
           exit(EXITRC)
```

## HFS2GDG JCL

```
//jobcard…
//*********************************************************************
//* EXECUTE HFS2GDG - Copy each file in an HFS directory to a GDG     *
//*                                                                   *
//* HFS2GDG is the REXX EXEC                                          *
//*                                                                   *
//* First parm is the REXX EXEC name to IKJEFT01                      *
//* Second parm is the target directory                               *
//* Third parm is a pattern to use in the target directory            *
//*       YES to DELETE HFS after COPY, NO to KEEP HFS                *
//* Fourth parm is a pattern to use in the target directory           *
//*       Only supports a trailing '*'                                *
//*                                                                   *
//* SYSEXEC points to the PDS that the REXX EXEC is uploaded into     *
//* MODEL points to the zero generation of the GDG or the GDG base    *
//*********************************************************************
//HFS2GDG  EXEC PGM=IKJEFT01,
//         PARM='HFS2GDG /your/dir YES EXAMPLE*'
//SYSEXEC  DD   DSN=your.EXEC,DISP=SHR
//SYSTSPRT DD   SYSOUT=*
//SYSTSIN  DD   DUMMY
//MODEL    DD   DSN=your.gdg(0),DISP=SHR
```

HFS2GDG first confirms that the MODEL DD exists and points to a valid GDG. It must have the ZERO generation so it knows where to begin the numbering of subsequent additions to the GDG. If the ZERO generation is not provided, it will have problems cataloguing the generated datasets. HFS2GDG reads the source directory for all ‘regular’ files with the named pattern. It then uses OCOPY to copy the file to the next GDG DSN. If the DELETE PARM is set to YES, the original HFS file is RMed (removed).

In the JCL:

1 SYSEXEC points to the PDS where you put HFS2GDG.

2 The MODEL DD in the JCL points to the ZERO generation of the target GDG to receive the files.

3 The second PARM is the source HFS directory to find the files (usually the application directory).

4 The third PARM is the DELETE indicator. If YES, this deletes the source HFS file. If NO, the source HFS file is **not** deleted.

5 The fourth PARM is a name pattern used to find the source file on the HFS directory. This name will be wildcarded so all files beginning with this value will be OCOPYed. Only a trailing '*' is supported.

6 Remember USS is case sensitive, so use CAPS OFF in your ISPF session.

## HFS2GDG OUTPUT

```
OPSROZ.TEST.GDG is a valid GDG Base

OCOPY successful YOUR.GDG.G0005V00 <-- /your/dir/EXAMPLE0001
RM successfully removed /your/dir/EXAMPLE0001
OCOPY successful YOUR.GDG.G0006V00 <-- /your/dir/EXAMPLE0002
RM successfully removed /your/dir/EXAMPLE0002
OCOPY successful YOUR.GDG.G0007V00 <-- /your/dir/EXAMPLE0003
RM successfully removed /your/dir/EXAMPLE0003
OCOPY successful YOUR.GDG.G0008V00 <-- /your/dir/EXAMPLE0004
RM successfully removed /your/dir/EXAMPLE0004

4 HFS files moved from /your/dir to MVS DSN's, RC=0
```

## HFS2GDG REXX EXEC

```
/*********************************************************************/
/*                              REXX                                 */
/*********************************************************************/
/* Purpose: Pull all HFS files from a directory and load a GDG       */
/*-------------------------------------------------------------------*/
/* Syntax:  HFS2GDG targdir delete pattern                           */
/*-------------------------------------------------------------------*/
/* Parms: targdir    - Target directory                              */
/*        delete     - YES to DELETE HFS after COPY, NO to KEEP HFS  */
/*        pattern    - File name pattern in target directory         */
/*                     trailing asterisk only                        */
/*                                                                   */
/* Notes: DDNAME MODEL must be allocated and point to the zero gen   */
/*        or the GDG base. This DSN will be used as the pattern for  */
```

```
/*        all additions to the GDG from this EXEC.                    */
/*********************************************************************/
/*                        Change Log                                 */
/*********************************************************************/
/* Accept arguments                                                  */
/*********************************************************************/
parse arg targdir delete pattern
if targdir = '' then
    do
     EXITRC = 12
     msg = 'Target directory name is missing'
     signal shutdown
    end
/*********************************************************************/
/* Default DELETE to NO                                              */
/*********************************************************************/
if delete = '' then
    delete = 'NO'
/*********************************************************************/
/* Default PATTERN to '*'                                            */
/*********************************************************************/
if pattern = '' then
    pattern = '*'
else
    if pos('*',pattern) = Ø then
       pattern = pattern||'*'
/*********************************************************************/
/* Confirm MODEL DD is allocated                                     */
/*********************************************************************/
LRC = listdsi('MODEL' 'FILE')
if LRC <> Ø then
    do
     EXITRC = 12
     msg = 'MODEL DD is missing'
     signal shutdown
    end
modeldsn = sysdsname
/*********************************************************************/
/* Establish allocation values for the GDG                           */
/*********************************************************************/
lrecl = syslrecl
blksize = sysblksize
spaceunit = sysunits
primary = sysprimary
secondary = sysseconds
unit = sysunit
recfm = ''
temprecfm = sysrecfm
do i=1 to length(temprecfm)
    recfm = recfm||' '||substr(temprecfm,i,1)
```

```
end
/*********************************************************************/
/* Parse the DSN in the DD to see whether it is a GDG                */
/*********************************************************************/
revdsn = reverse(modeldsn)
if pos('V',revdsn) = 3 & pos('G.',revdsn) = 8 then
    do
     parse var revdsn goovoo '.' revdsn
     gdgbase = reverse(revdsn)
     goovoo = reverse(goovoo)
     say gdgbase 'is a valid GDG Base'
     say
    end
else
    do
     EXITRC = 12
     msg = 'DSN in INPUT DD does not appear to be a GDG Base' modeldsn
     signal shutdown
    end
/*********************************************************************/
/* Get DSN details from a LISTCAT                                    */
/*********************************************************************/
call outtrap 'listc.'
address TSO "LISTCAT ENT('"gdgbase"')"
EXITRC = RC
if EXITRC <> Ø then
    do
     msg = 'LISTCAT error on' gdgbase
     signal shutdown
    end
/*********************************************************************/
/* Check whether DSN is a GDG Base                                   */
/*********************************************************************/
if word(listc.1,1) <> 'GDG' then
    do
     EXITRC = 12
     msg = 'DSN on MODEL DD is not from a GDG' gdgbase
     signal shutdown
    end
/*********************************************************************/
/* Establish the SYSCALL environment                                 */
/*********************************************************************/
if syscalls('ON') <> Ø then
    do
     EXITRC = 2Ø
     msg = 'Error establishing the SYSCALL environment'
     signal shutdown
    end
/*********************************************************************/
/* Read the target directory                                         */
```

```
/*********************************************************************/
address SYSCALL "readdir" targdir 'file.' 'stat.'
EXITRC = RETVAL
if EXITRC <> Ø then
    do
     EXITRC = 12
     msg = 'Error reading target directory' targdir,
           'RETVAL='RETVAL 'ERRNO='ERRNO 'ERRNOJR='ERRNOJR
     signal shutdown
    end
/*********************************************************************/
/* Process only regular files                                        */
/*********************************************************************/
hfsnum = Ø
do i=1 to file.Ø
    if stat.i.ST_TYPE = S_ISREG then
       do
/*********************************************************************/
/* Pattern processing                                                */
/*********************************************************************/
        if pattern <> '*' then
           do
            splat = pos('*',pattern) - 1
            if substr(file.i,1,splat) /== substr(pattern,1,splat) then
               iterate
           end
/*********************************************************************/
/* Maintain counter and HFS filename                                 */
/*********************************************************************/
        hfsnum = hfsnum + 1
        hfsfile = targdir'/'file.i
/*********************************************************************/
/* Allocate the PATH                                                 */
/*********************************************************************/
       "ALLOC F(HFSIN) PATH('"hfsfile"') PATHOPTS(ORDONLY)"
        EXITRC = RC
        if EXITRC <> Ø then
           do
            msg = 'ALLOC PATH error on' hfsfile
            signal shutdown
           end
/*********************************************************************/
/* Set the GOOVOO for the next GDG                                   */
/*********************************************************************/
        parse var goovoo 'G' num1 'V' num2
        num1 = right((num1 + 1),4,Ø)
        if num1 = 'ØØØØ' then
           num2 = right((num2 + 1),2,Ø)
        goovoo = 'G'||num1||'V'||num2
        dsn = gdgbase'.'goovoo
```

```
/*********************************************************************/
/* Allocate the +1 GDG                                               */
/*********************************************************************/
       "ALLOC F(GDGOUT) DA('"dsn"') NEW CATALOG UNIT("unit")" spaceunit,
       "LRECL("lrecl") BLKSIZE("blksize") RECFM("recfm")",
       "SPACE("primary secondary")"
        EXITRC = RC
        if EXITRC <> Ø then
           do
            msg = 'ALLOC DSN error on' dsn
            signal shutdown
           end
/*********************************************************************/
/* OCOPY the HFS to the GDG                                          */
/*********************************************************************/
       "OCOPY INDD(HFSIN) OUTDD(GDGOUT) BINARY"
        EXITRC = RC
        if EXITRC <> Ø then
           do
            msg = 'OCOPY error on' hfsfile '-->' dsn
            signal shutdown
           end
        else
           do
           "FREE F(HFSIN)"
           "FREE F(GDGOUT)"
            say 'OCOPY successful' dsn '<--' hfsfile
           end
/*********************************************************************/
/* Determine whether DELETE processing is required                   */
/*********************************************************************/
        if delete = 'YES' then
           do
           "BPXBATCH SH rm" hfsfile
            EXITRC = RC / 256
            if EXITRC <> Ø then
               do
                msg = 'BPXBATCH error during RM' hfsfile
                signal shutdown
               end
            else
               do
                say 'RM successfully removed' hfsfile
               end
           end
       end
end
/*********************************************************************/
/* Completion message                                                */
/*********************************************************************/
```

```
if delete = 'YES' then
    context = 'moved'
else
    context = 'copied'
msg = hfsnum 'HFS files' context 'from' targdir 'to MVS DSN''s'
/*********************************************************************/
/* Shutdown                                                          */
/*********************************************************************/
shutdown: call syscalls('OFF')
          say
          say msg', RC='EXITRC
          exit(EXITRC)
```

*Robert Zenuk*
*Systems Programmer (USA)*


## REXX commands and the USS environment

The USS (Unix System Services) is the Unix environment available with the z/OS operating system. z/OS Unix System Services (z/OS Unix) gives the z/OS operating system an open standards interface. It consists of three features:

1 A shell and utilities that can be used to enter shell commands, write shell scripts, and work with the filesystem.

2 A debugger that an application programmer can use to debug a z/OS Unix System Services application program written using the C language.

3 An API (Application Program Interface) that can be used to manage and control the USS environment.

This document describes how to use a REXX program to communicate with the USS through the API.

To establish a connection between the programs (REXX, COBOL, Assembler, C, etc) and z/OS Unix we use the syscall function. This is illustrated in Figure 1.

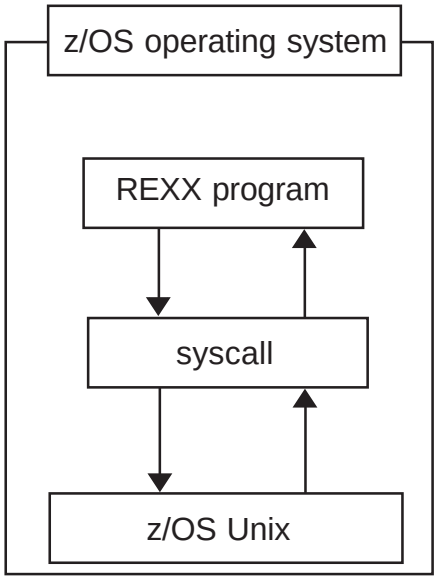


*Figure 1: Connections*

## TYPES OF API FUNCTION

The general types of API function are:

- Application management (process/thread)
- File and directory management
- IPC management (InterProcess Communication)
- Security management
- Network and socket management
- System management.

The following are examples of some API functions (the name is followed by the function's scope):

- access – invokes the access callable service to determine whether the caller can access a file.
- chdir – invokes the chdir callable service to change the working directory.

- fstat – invokes the fstat callable service to get status information about a file that is identified by its file descriptor.
- getcwd – invokes the getcwd callable service to get the pathname of the working directory.
- getegid – invokes the getegid callable service to get the effective group ID (GID) of the calling process.
- getmntent – invokes the w_getmntent callable service to get information about the mounted filesystems, or a specific mounted filesystem, and returns the information formatted in a stem.
- getpsent – invokes the w_getpsent callable service to provide data describing the status of all the processes to which you are authorized. The data is formatted in a stem. The PS_ variables, or their numeric equivalents, are used to access the fields.
- readdir – invokes the opendir, readdir, and closedir callable services to read multiple name entries from a directory and format the information in a stem.
- sleep – invokes the sleep callable service to suspend running of the calling thread (process) until either the number of seconds specified by a number has elapsed, or a signal is delivered to the calling thread to invoke a signal-catching function or end the thread.

## CALLING THE APIs

The user that starts the REXX program must have an Open Edition Segment (OMVS). The OMVS segment is created, in the case of the RACF security system, with the following commands:

```
ALTUSER userid  OMVS(UID(1111) GID(2222) HOME('/u/userid') PROGRAM('/
bin/sh'))
```

*1111* is the ID assigned to the user, and *2222* is the ID assigned to the group.

The rules for calling the API's function from REXX are:

- Initialize the environment.
- Include the address syscall statement.
- Include the API name in quotes.

Below are some examples of REXX API usage.

### XUSERS

This example shows how to retrieve user profile information for OMVS users:

```
/*** REXX ****/
/*************************************************************/
/*  This program is a REXX UNIX API call                     */
/*  The API name is : getpwent                               */
/*  Target:  Retrieve user profile information for OMVS users */
/*************************************************************/
 IF SYSCALLS('ON')>3 THEN
    DO
       SAY 'Unable to establish the SYSCALL environment'
       SAY 'Return code was 'RC
       RETURN
    END
say "User profile information for OMVS users"
say " "
say "Today is " date() " " time()
say " "
 ADDRESS SYSCALL
DO FOREVER
    "getpwent pw."      /* Store the output to the STEM variable pw.x */
    IF retval=Ø ! retval=-1 THEN ,
       LEAVE
 say     'Name='left(pw.PW_NAME,1Ø),
         'Uid='left(pw.PW_UID,5) ,
         'Gid='left(pw.PW_GID,4) ,
         'Dir='left(pw.PW_DIR,25) ,
         'Shell='left(pw.PW_SHELL,8)
 end
 exit Ø
```

Remarks:

- SYSCALLS('ON') – environment initialization.

- ADDRESS SYSCALL – the syscall statement.
- "getpwent pw." – API name in quotes.
- pw. – this is the REXX stem variable to store the command output.
- pw.0 – contains the number of variables returned. You can use the predefined variables beginning with PW_ to access the values:
  - PW_DIR – the initial working directory
  - PW_GID – the group ID
  - PW_NAME – the TSO/E user ID
  - PW_SHELL – the name of the initial user program.
  - PW_UID – the user ID (UID) as defined to RACF.

## XUNAME

This example shows how to retrieve information about the system you are running on:

```
/*** REXX ****/
/*************************************************************/
/*  This program is a REXX UNIX API call                     */
/*  The API name is : uname                                  */
/*  Target: Retrieve info about the system you are running on */
/*************************************************************/
 IF SYSCALLS('ON')>3 THEN
    DO
    SAY 'Unable to establish the SYSCALL environment'
    SAY 'Return code was 'RC
    RETURN
    END
say "System info"
say " "
say "Today is " date() " " time()
say " "
 ADDRESS SYSCALL
    "uname us."      /* Store the output to the STEM variable us.x */
 if retval = -1 then do
                        "strerror" errno errnojr 'erd.'
                        say "Error  =" erd.se_errno
```

```
                        say "Reason =" erd.se_reason
                        say "Module =" erd.se_modid
                        exit
                       end
/* The name of the hardware type on which the system is running */
say    'Hardware type : 'left(us.u_machine,10)
/* The name of this node within the communication network */
say    'System Id     : 'left(us.u_nodename,10)
/* The name of Operating system (Z/OS) */
say    'OS Type       : 'left(us.u_sysname,10)
/* The current release level of this Operating System */
say    'OS Release    : 'left(us.u_release,10)
/* The current Version level of this Operating System */
say    'OS Version    : 'left(us.u_version,10)
exit 0
```

Remarks:

- *retval* is the special variable containing the command return code. A value equal to -1 means that the command is in error.
- The API strerror retrieves text for error codes and reason codes.
- *erd.* is the REXX stem variable to store the command output.
- *us.0* contains the number of variables returned. To access the information, you specify the predefined variables beginning with U_ that derive the appropriate numeric value:
  - U_MACHINE – the name of the hardware type on which the system is running.
  - U_NODENAME – the name of this node within the communication network.
  - U_RELEASE – the current release level of this implementation.
  - U_SYSNAME – the name of this implementation of the operating system (z/OS).
  - U_VERSION – the current version level of this release.

## XPROCESS

This example shows how to retrieve data describing the status of all processes:

```
/* REXX */
/*****************************************************************/
/*  This program is a REXX UNIX API call                         */
/*  The API name are: getpsent and getpwuid                      */
/*  Target: Retrieve data describing the status of all process   */
/*                                                               */
/*  Remark:                                                      */
/*  To display all process data, the user that gives the command*/
/*  must have uid = 0                                            */
/*  If the user has a uid ne 0 the command shows the data only   */
/*  for the user giving the command                              */
/*****************************************************************/
 IF SYSCALLS('ON')>3 THEN
    DO
    SAY 'Unable to establish the SYSCALL environment'
    SAY 'Return code was 'RC
    RETURN
    END
say "Process Information"
say " "
say "Today is " date() " " time()
say " "
 say left('Name',10) left('Pid',10) ,
     left('PPid',10) left('USR_CPUT',10) left('Racf_id',10) 'Command'
 say left('----',10) left('---',10) ,
     left('----',10) left('(Secs)',10) left('-------',10) '-------'
 ADDRESS SYSCALL
    "seteuid 0"  /* The user that give the commands
                    must have uid = 0)  */
    "getpsent ps."      /* store the output to the stem variable ps.x */
    if retval = -1 then do
                         "strerror" errno errnojr 'erd.'
                         say "Error  =" erd.se_errno
                         say "Reason =" erd.se_reason
                         say "Module =" erd.se_modid
                         exit
                        end
 do i=1 to ps.0
   "getpwuid "ps.i.ps_euid" pw."  /* store the output to the
                                     stem variable pw.x */
    name= pw.pw_name       /* The TSO/E user ID                    */
    namr= pw.pw_uid        /* The user ID (UID) as defined to RACF */
    say left(name,10),
    left(ps.i.ps_pid,10),  /* Process ID        */
    left(ps.i.ps_ppid,10), /* Parent Process ID */
```

```
    left(ps.i.PS_USERTIME,1Ø),  /* User CPU times in 1/1ØØths of sec */
    left(namr,1Ø),
    ps.i.ps_cmd            /* Command           */
end
exit Ø
```

Remarks:

- "seteuid 0" – the user giving the commands must have uid = 0.

  Users authorized to perform special functions are defined as having appropriate privileges, and they are called superusers. Appropriate privileges also belong to users with:

  – a user ID of zero.

  – RACF-supported user privileges trusted and privileged, regardless of their user ID.

  A user can switch to superuser authority (with an effective UID of 0) if the user is inserted into the ACL (Access Control List) of the BPX.SUPERUSER FACILITY class profile within RACF.

  The RACF command is:

  ```
  permit BPX.SUPERUSER id(user/group) access(READ) class(FACILITY)
  ```

  The "seteuid 0" command can be used to switch to superuser authority.

  To display all process data, the user giving the command must have uid = 0. If the user has a uid that is not equal to 0, the command shows only the process for that user.

- ps.0 contains the number of processes for which information is returned. ps.1 to ps.*n* (where *n* is the number of entries returned) each contain process information for the *n*th process. You can use the predefined variables that begin with PS_ to access the information. For example, ps.1.ps_pid is the process ID for the first process returned:

- PS_CMD – command
- PS_CONTTY – controlling tty
- PS_EGID – effective group ID
- PS_EUID – effective user ID
- PS_FGPID – foreground process group ID
- PS_MAXVNODES – maximum number of vnode tokens allowed
- PS_PATH – pathname
- PS_PGPID – process group ID
- PS_PID – process ID
- PS_PPID – parent process ID
- PS_RGID – real group ID
- PS_RUID – real user ID
- PS_SERVERFLAGS – server flags
- PS_SERVERNAME – server name supplied on registration
- PS_SERVERTYPE – server type (File=1; Lock=2)
- PS_SGID – saved set group ID
- PS_SID – session ID (leader)
- PS_SIZE – total size
- PS_STARTTIME – starting time, in POSIX format (seconds since the Epoch,00:00:00 on 1 January 1970)
- PS_STAT – process status
- PS_STATE – process state. This value can be expressed as one of the following:
    - PS_CHILD – waiting for a childprocess

- PS_FORK – fork() a new process
- PS_FREEZE – QUIESCEFREEZE
- PS_MSGRCV – IPC MSGRCV WAIT
- PS_MSGSND – IPC MSGSND WAIT
- PS_PAUSE – MVSPAUSE
- PS_QUIESCE – quiesce termination wait
- PS_RUN – running, not in kernel wait
- PS_SEMWT – IPC SEMOP WAIT
- PS_SLEEP – sleep() issued
- PS_WAITC – communication kernel wait
- PS_WAITF – filesystem kernel wait
- PS_WAITO – other kernel wait
- PS_ZOMBIE – process cancelled
- PS_ZOMBIE2 – process terminated yet still the session or process group leader.

– PS_SUID – saved set user ID

– PS_SYSTIME – system CPU time, a value of the type clock_t, which needs to be divided by sysconf(_SC_CLK_TK) to convert it to seconds. For z/OS Unix, this value is expressed in hundredths of a second.

– PS_USERTIME – user CPU time, a value of the type clock_t, which needs to be divided by sysconf(_SC_CLK_TK) to convert it to seconds. For z/OS Unix, this value is expressed in hundredths of a second.

– PS_VNODECOUNT – current number of vnode tokens.

## XMOUNT

This example shows how to get information about the mounted filesystems:

```
/*** REXX ****/
/*************************************************************/
/*  This program is a REXX UNIX API call                     */
/*  The API name are : getmntent and statfs                  */
/*  Target: get information about the mounted filesystems,   */
/*          or a specific mounted filesystem, and return     */
/*          the information formatted in a stem.             */
/*************************************************************/
 IF SYSCALLS('ON')>3 THEN
    DO
    SAY 'Unable to establish the SYSCALL environment'
    SAY 'Return code was 'RC
    RETURN
    END
say "Mounted Filesystem"
say " "
say "Today is " date() " " time()
say " "
ADDRESS SYSCALL
    "getmntent mounts." /* get information about the mounted
                           filesystem and store the output to
                           STEM variable mounts.x */
    if retval = -1 then do
                         "strerror" errno errnojr 'erd.'
                         say "Error  =" erd.se_errno
                         say "Reason =" erd.se_reason
                         say "Module =" erd.se_modid
                         exit
                        end
 do i = 1 to mounts.Ø
  /* The name of the HFS data set containing the file system */
  say i") Dataset-name   " left(mounts.MNTE_FSNAME.i,3Ø)
  /* The mountpoint pathname */
  say  "    Mount Point    " left(mounts.MNTE_PATH.i,3Ø)
  /* The name of the owning system */
  say  "    Sysname        " left(mounts.MNTE_sysname.i,3Ø)
  mnt = strip(mounts.MNTE_FSNAME.i)
   "statfs ("mnt") st."  /* Obtain status information about a
                            specified file system */
/* Filesystem Block size */
   say "    Blockzise      " left(st.STFS_BLOCKSIZE,6) ,
           "(Bytes)"
/* Filesystem Total space in block-size units */
   say "    Total space    " left(st.STFS_TOTAL,6) ,
       "(Blksize Unit)"
```

```
/* Filesystem Allocated space in block-size units */
   say "    Alloc space    " left(st.STFS_INUSE,6)   ,
        "(Blksize Unit)"
/* Filesystem Space available in block-size units */
   say "    Space avail    " left(st.STFS_AVAIL,6) ,
       "(Blksize Unit)"
/* Filesystem % Utilization */
   say "    % Utilization  " ((st.STFS_INUSE/st.STFS_TOTAL)*100) %1"%"
  say "-------------------------------------------------------"
 end
 exit 0
```

## XPPRIM

To simplify the use of the programs, I have created an ISPF interface. This is the ISPF panel in which the information is shown:

```
)attr default(%+_)
~ type(text) intens(high) caps(off) just(asis ) color(turq)
hilite(reverse)
% type(text) intens(high) color(white)
+ type(text) color(blue)
! type(text) color(red)
@ type(output)
)Body Expand(\\)
%-\-\- ~UNIX System Services Display%-\-\-
%
%
!Selection ===>_ZCMD +
%
%
%
+1% Mount .....+Get information about the mounted file systems
+2% Process ...+Retrieve data describing the status of all process
+3% Users .....+Retrieve user profile information for OMVS users
+4% Sysinfo ...+Retrieve info about the system you are running on
+X% Exit
+
+
+Userid: &ZUSER  +  Date: &ZDATE  +   Time: &ZTIME +
+
+
+
)PROC
&zsel = trans(&ZCMD
              1,'CMD(XMOUNT)'
              2,'CMD(XPROCESS)'
```

```
              3,'CMD(XUSERS)'
              4,'CMD(XUNAME)'
              X,'EXIT'
              *,'?'
              )
)end
```

## XSTART

This is the start-up program.

```
/* REXX */
dsn = "userid.lib.exec"
address TSO "ALTLIB ACTIVATE APPLICATION(CLIST) DA("dsn")"
address ISPEXEC "LIBDEF ISPPLIB DATASET ID("dsn") STACK"
address ISPEXEC "LIBDEF ISPMLIB DATASET ID("dsn") STACK"
address ISPEXEC "SELECT PANEL(XPPRIM) NEWAPPL(ISR) PASSLIB"
address ISPEXEC "LIBDEF ISPMLIB"
address ISPEXEC "LIBDEF ISPPLIB"
address TSO "ALTLIB DEACTIVATE APPLICATION(CLIST) "
exit Ø
```

## INSTALLATION

Allocate the userid.lib.exec library with these characteristics:

```
Organization . . . . . : PO
Record format . . . . : FB
Record length . . . . . : 8Ø
Block size . . . . . .: 2792Ø
1st extent tracks . . . : 3
Secondary tracks . .: 15
Data set name type : PDS
```

Copy XUSERS, XMOUNT, XPROCESS, XUNAME, XPPRIM, and XSTART to the userid.lib.exec library.

To get the appropriate authorization, add the OMVS segment:

```
ALTUSER userid OMVS(UID(1111) GID(2222) HOME('/u/userid') PROGRAM('/bin/
sh'))
```

Get authorization for the user/group to switch to superuser:

```
permit BPX.SUPERUSER id(user/group) access(READ) class(FACILITY).
```

The "seteuid 0" command can be used to switch to superuser authority.

## To start-up, from the TSO Option 6 type:

```
ex 'userid.lib.exec(xstart)' ex
```

## This selection menu is shown:

```
--------------------  UNIX System Services Display --------------------

Selection ===>

1  Mount ..... Get information about the mounted file systems
2  Process ... Retrieve data describing the status of all process
3  Users .....  Retrieve user profile information for OMVS users
4  Sysinfo ... Retrieve info about the system you are running on
X  Exit

Userid: userid     Date: 05/04/12    Time: 14:59
```

## Selecting Option 1 gives this report:

```
 Mounted Filesystem

 Today is  12 Apr 2005   15:06:35

 1) Dataset-name    OMVS.DEVEL.SVILPLEX.ILMUC
      Mount Point     /DEVEL/etc/licmgmt/certs
Sysname         DEVEL
Blockzise       4096   (Bytes)
Total space     180    (Blksize Unit)
Alloc space     5      (Blksize Unit)
Space avail     158    (Blksize Unit)
% Utilization   2%
-----------------------------------------------------
2) Dataset-name    OMVS.PROD.PRODPLEX.ILMSPDM
     Mount Point     /PROD/etc/licmgmt/ilmadb
     Sysname         PROD
     Blockzise       4096   (Bytes)
     Total space     180    (Blksize Unit)
     Alloc space     5      (Blksize Unit)
     Space avail     158    (Blksize Unit)
     % Utilization   2%
-----------------------------------------------------
```

## Selecting Option 2 gives this report:

```
Process Information

Today is  12 Apr 2005   15:08:50

Name      Pid      PPid      USR_CPUT      Racf_id      Command
----      ---      ----      (Secs)      -------      -------
```

```
ADSM000          1      0          382          0      BPXPINPR
ADSM000   33619971      1       163814          0      EZBTCPIP
CXS00000  16842756      1         8085      70175      EZACIC03
ADSM000      65541      1       163800          0      EZBTTSSL
ADSM000      65542      1       163827          0      EZBTMCTL
ADSM000      65543      1       163829          0      EZACFALG
ADSM000      65544      1       163829          0      EZASASUB
AFO00000  83951626      1       520870      70183      KOGTCSVR
RMF          65547      1      2168749      70182      ERB3GMFC
```

## Selecting Option 3 gives this report:

```
User profile information for OMVS users

 Today is  12 Apr 2005   15:12:58

Name=ADS0000   Uid=0      Gid=6     Dir=/usr/lpp/adsm      Shell=/bin/sh
Name=AFO0000   Uid=70183  Gid=1001  Dir=                   Shell=
Name=AOP0000   Uid=657    Gid=6     Dir=/usr/lpp/Printsrv/bin  Shell=/
bin/sh
Name=AUTO0001  Uid=0      Gid=23    Dir=/u/autodis1        Shell=/bin/sh
Name=AUTOM00 1 Uid=99991  Gid=23    Dir=/                  Shell=/bin/sh
Name=AUTOM002  Uid=99992  Gid=23    Dir=/                  Shell=/bin/sh
Name=AUTOM003  Uid=99993  Gid=23    Dir=/                  Shell=/bin/sh
```

## Selecting Option 4 gives this report:

```
System info

 Today is  12 Apr 2005   15:18:15


Hardware type      2084
System Id          PROD
OS Type            OS/390
OS Release         14.00
OS Version         03
```

*Magni Mauro*
*Systems Engineer (Italy)*

# Retrieving the creation date of a catalogued dataset

This tool is designed to retrieve the creation date of a catalogued dataset.

## PURPOSE

As most of us know, in the mainframe world, there are too many applications that require production support on a 24x7 basis. At times it may be necessary to get old versions of datasets and analyse the root cause of the issue that had occurred at a later date.

Most of us are also aware of the fact that DASD datasets are regularly migrated on mainframe systems to save DASD space or ensure effective utilization of DASD volumes. Some datasets will be on tape as well. Generally, application programmers restore migrated datasets to live DASD to see the creation date and other attributes. They may also use a TMS (tape management system) to view tape dataset attributes, like creation date. If there are too many datasets to be restored or used in TMS, it will definitely be a tedious and time consuming process and the application might miss the SLA (Service Level Agreement) too.

With this tool, it has become really easy to locate a dataset that was created on a specific date. In other words, the tool is written to get the creation date of any catalogued dataset and GDG base.

## APPLICATION

The tool can be used with the following types of dataset:

1 VSAM (all types, such as KSDS, RRDS, ESDS).

2 Non-VSAM (PS/PDS/PDSE).

3 GDG bases.

## SOURCE CODE

```
/*rexx*/
"ispexec control errors return"
Parse arg ds
ds = strip(ds,B,"'")
x = Outtrap(dtl.)
"listcat entries('"ds"') ALL"
savrc = rc
x = outtrap(off)
if savrc = 0 then
  do
     juldt = reverse(substr(reverse(strip(dtl.4)),1,8))
     juldt = substr(juldt,3,2)||substr(juldt,6,3)
     tmp = DATE('U',juldt,'J')||' <> '||juldt
     tmp = overlay(tmp,ds,46)
     say tmp
  end
else
  do
     zedsmsg = 'RC='||savrc||'. <F1>'
     zedlmsg = 'LISTCAT function failed for this dataset'
     "ispexec setmsg msg(ISRZ001)"
  end
Return 0
```

I call this member CD. You can save it with any name and execute it with the name you chose when you saved this source.

## TO EXECUTE

You can execute this REXX in two different ways.

First, from any command panel, issue the following command:

```
TSO CD 'MYID.REXX.SOURCE'
```

Upon successful execution of the command, it will throw a message containing the dataset name, creation date of the dataset (in MM/DD/YY (US standard) <> YYDDD (Julian) format). It's necessary to supply the fully-qualified dataset name.

Quotes can optionally be entered.

The ISPF command panel looks like this:

```
   Menu  Utilities  Compilers  Options  Status  Help

                                ISPF PRIMARY OPTION MENU
  Option ===> TSO CD MYID.REXX.SOURCE

 Ø  Settings      Terminal and user parameters
 1  View          Display source data or listings
.....................................................................................
.....................................................................................
.....................................................................................
.....................................................................................
.....................................................................................



  MYID.REXX.SOURCE                                  Ø9/23/Ø4 <> Ø4267
 ***
```

The second option is to issue the **CD** command from the DSLIST panel (ie use a line command against the dataset name that you want to select):

```
   Menu  Options  View  Utilities

 DSLIST - Data Sets Matching MYID.REXX.S*                Row 1 of 8
  Command ===>                                           Scroll ===> CSR

 Command - Enter "/" to select action             Message         Volume
  ------------------------------------------------------------------------
          MYID.REXX.SOURCE1                                       ABCØØ1
          MYID.REXX.SOURCE2                                       ABCØØ2
          MYID.REXX.SOURCE3                                       ABCØØ3
CD        MYID.REXX.SOURCE4                    CD     RC=Ø        MIGRAT2
  =       MYID.REXX.SOURCE5                    CD     RC=Ø        ABCØØ5
   =      MYID.REXX.SOURCE6                    CD     RC=Ø        ??????
          MYID.REXX.SOURCE7                                       ABCØØ7
          MYID.REXX.SOURCE8                                       ABCØØ8
 ************************* End of Data Set list ************************



MYID.REXX.SOURCE4                                 Ø9/16/Ø4 <> Ø426Ø
MYID.REXX.SOURCE5                                 1Ø/Ø6/Ø4 <> Ø428Ø
MYID.REXX.SOURCE6                                 Ø4/Ø7/Ø3 <> Ø3Ø97
***
```

After typing **CD** and pressing *Enter*, messages similar to those shown above will be seen for every dataset that you selected using the command.

## ERROR MESSAGES

When an error occurs (say, by using an invalid dataset name) the return code will be displayed as an error message in the top right-hand corner.

Pressing F1 (F1 must be mapped to HELP in the keylist panel) will give you the long error message. Alternatively, type HELP and press *Enter* in the command line as soon as you see the short message in the top right-hand corner of the screen to get the detailed message (see the example below).

If you select a dataset that does not exist in the system you will see:

```
                            Data Set List Utility
RC=4. <F1>
  Option ===>   TSO CD 'THIS.DATASET.DOESNOT.EXIST'

     blank Display data set list               P Print data set list
         V Display VTOC information           PV Print VTOC information

 Enter one or both of the parameters below:
    Dsname Level . . .
    Volume serial  . .

 Data set list options
    Initial View . . . 1  1. Volume     Enter "/" to select option
                          2. Space      /  Confirm Data Set Delete
                          3. Attrib     /  Confirm Member Delete
                          4. Total      /  Include Additional Qualifiers

When the data set list is displayed, enter either:
  "/" on the data set list command field for the command prompt pop-up,
  an ISPF line                                              REXX exec, or
  "=" to execut   LISTCAT function failed for this dataset
```

In this second example, a non-standard dataset name (node length greater than eight characters) is used:

```
                     Data Set List Utility                   RC=12. <F1>
  Option ===>   TSO CD 'I.AMGIVING.A.UNREALISTIC.DATASETNAME'
```

```
     blank Display data set list                P Print data set list
         V Display VTOC information            PV Print VTOC information

 Enter one or both of the parameters below:
     Dsname Level . . .
    Volume serial  . .

 Data set list options
    Initial View . . . 1  1. Volume     Enter "/" to select option
                          2. Space      /  Confirm Data Set Delete
                          3. Attrib     /  Confirm Member Delete
                          4. Total      /  Include Additional Qualifiers

When the data set list is displayed, enter either:
  "/" on the data set list command field for the command prompt pop-up,
  an ISPF line                                                REXX exec, or
  "=" to execut   LISTCAT function failed for this dataset
```

### RETURN CODES

The return codes used are:

- 0 – successful.
- 4 – dataset does not exist in the catalog.
- 12 – invalid dataset name supplied.

*Suresh Kumar Murugesan*
*Systems Analyst*
*Cognizant Technology Solutions (USA)*


## Enclave resource accounting

### INTRODUCTION

Introduced with MVS/ESA and extended in OS/390, an enclave is a construct that represents a business transaction, or a unit of work. In fact, enclaves are a new method of managing business transactions that give you more effective control

over non-traditional workloads. Although it is not necessary to understand the implementation of enclaves, it is profitable to understand why they were invented and which products are using them, because several products on z/OS are already using enclaves, and you can expect more to use them in the near future. If you use one of these products, you need to manage enclaves as part of your performance policy. To get the most out of your z/OS investment, you should know what enclaves are and how to use them effectively.

Briefly stated, an enclave is an anchor for accumulating the resources consumed by a transaction, regardless of where it is executing. This means that enclaves provide a way to account for the resources used by a business transaction, even when those resources are consumed by multiple work units, even across multiple address spaces or even across multiple systems in a parallel sysplex. The intention is to have the enclave map closely to the end user's view of a transaction. In other words, the transaction represented by an enclave consists of pieces that can span many server address spaces and simultaneously share these address spaces with other transactions.

From a performance management point of view, we know that z/OS Workload Manager (WLM) uses a goal-oriented approach to managing workloads running in the sysplex. You define performance goals – response times, for example – for each type of work. Such goals apply to interactive workloads whose transactions, as reported to z/OS, closely match the end-user's view of transactions. Since goals are known and resource consumption can be monitored, WLM can control the transaction directly rather than indirectly through the associated address spaces. Redefining the definition of an enclave, one may say that an enclave represents a transaction that can span multiple dispatchable units of work (SRBs and TCBs) in one or more address spaces that are reported and managed as a single unit. One of the most novel aspects of enclave transactions is that z/OS can manage them independently, even though several may be executing concurrently in the same address space.

Before enclaves existed, WLM could manage transactions only by adjusting the resources of the transactions' server address spaces. Even if interactive transactions (such as CICS transactions) are known to WLM, resources are distributed indirectly by allocating resources to the address spaces serving the transactions. For example, each address space is assigned one dispatching priority and one I/O priority for all work running there. These priorities are determined by WLM, based on the work with the most stringent goals. If a transaction does not use enclaves then that transaction can be managed only on an address space basis. This means the transaction is tied to the address space processing the transaction. This method of transaction management works well to the extent that transactions with different goals or importance can be segregated into different address spaces. If transactions with multiple goals and importance are using the same server address spaces concurrently, WLM is limited in how precisely it can allocate resources to the work that needs them. Since many new types of work – mostly relating to client/server applications – tended to span multiple address spaces, a more precise method of transaction management became necessary to handle the complex internal flows of some of the new z/OS transaction managers and in fact enclaves are now used to manage these client/server transactional units.

Although enclaves have some characteristics that are similar to other types of transaction there are some important features that distinguish them:

- Enclaves allow a transaction to be managed separately from the server address space(s) processing the work (separation from home address space). In this case, a transaction's dispatchable units running in an enclave are still 'part' of the server address space, but those dispatchable units are managed separately from that address space. This means that the transaction's dispatchable units can be given different CPU and I/O priorities from the server address spaces in which they run and be managed to achieve different goals.

- Enclaves also allow a transaction to be managed separately from other work running in the same address space – this allows different types of work running in the same server address space to be assigned different WLM goals or different performance groups.
- In addition, enclaves allow a transaction to run against other work running in other address spaces such as traditional workloads like TSO, batch, CICS, etc. With enclaves you can now assign WLM goals and importance such that these same transactions, now running in enclaves, can compete for resources on a transactional basis.
- Finally, one of enclaves' objectives is to enable the system to monitor enclave activity and to allow users to understand the service requirements of their distributed workloads.

To sum up: the more precise transaction management possible with enclaves allows diverse workloads to be safely consolidated onto a single z/OS system or sysplex. This allows you to better deploy and monitor your business applications and meet the objectives defined in your service level agreement.

Enclaves can be one of the following types:

- Independent enclaves, which are used to represent new transactions that have not yet been associated with an address in the sysplex. The address space used to issue the enclave creation macro (IWMECREA) becomes the owner of the enclave. An independent enclave represents a complete SRM transaction. Its performance goal is assigned based on the service class to which it is classified when the enclave is created. Each independent enclave starts in period 1 of its service class and switches periods based on the service consumed by the dispatchable units belonging to the enclave.
- A dependent enclave, which represents the continuation of an existing address space transaction under a new set

of dispatchable units. Its performance goal is inherited from the existing address space transaction based on the service class and period being used to manage the address space at the instant the dependent enclave is created. CPU service consumed by a dependent enclave is treated as if it were consumed by the address space transaction, and can cause the address space along with the dependent enclave to switch into later periods.

In order to improve the transaction's overall response time, some work managers will split large transactions across multiple systems in a parallel sysplex. These work managers can use multisystem enclaves to provide consistent management and reporting for these types of transaction. A multisystem enclave begins as either an independent or dependent enclave on a single system. This enclave is called the 'original' enclave. If the work manager decides to involve other systems in the processing of the work unit, it issues a WLM service to 'export' the enclave (IWMEXPT) to other systems in the parallel sysplex.

Each work manager in the supporting address spaces on other systems can now issue WLM's service (IWMIMPT) to 'import' the enclave onto its system. This new, supporting enclave is called a 'foreign' enclave. The original enclave and the foreign enclaves are all referred to as one unit called a multisystem enclave.

## ONLINE MONITORING ENCLAVES

Just as with address space-oriented transactions, such as batch or TSO, enclave transactions consume processor and I/O resources. Since there are no limits to the complexity of an enclave transaction, it is mandatory to be able to monitor enclaves and their processor consumption online in order to support problem diagnosis and workload characterization. This becomes even more important as the number of subsystems exploiting enclaves grows and as use of these applications increases.

When it comes to online monitoring of enclaves there are currently two options available within the standard IBM's toolkit. The first one is SDSF's ENC display, while the second is RMF's Monitor III Enclave report.

The SDSF ENC (enclave) display shows the following enclave characteristics: the subsystem environment; total CPU time; the enclave type: IND (independent) or DEP (dependent); the owning address space number; the service class assigned; the scope of the enclave: either LOCAL (single-system) or MULTISYS (the enclave has an export token and so is multisystem-capable).

The detailed information command (I) displays: the enclave classification criterion, which is the transaction name (ie the transaction program name for the request); transaction class (class name within the subsystem); subsystem collection name as well as the process name associated with the request. Multisystem enclaves are displayed as multiple rows. When you act against any of these rows, SDSF issues the WLM service against the original enclave.

The new RMF Monitor III Enclave report provides information about enclaves to support you in high-level performance problem determination and performance management for thread-based resource consumption and delays. The report shows you the CPU time consumed by an enclave during the sampled range and the total CPU consumption since the enclave was created. This allows you to identify business transactions that are high CPU consumers. It also enables you to distinguish transactions currently consuming significant CPU (possibly looping) from those whose life-time CPU usage is high (candidates for performance analysis). Resource usage and delay percentages show you the progress of a single enclave and indicate possible resource constraints. The report can be tailored to list only those enclaves belonging to the same subsystem, that have the same business goals or that are owned by the same address space. Transaction attributes can be selected to further limit the transactions

being analysed. This allows you to drill down to a single user ID or a specific database request.

There is, however, an interesting phenomenon for those of us who use RMF Monitor III to review enclave activity. The *EAppl%* column shows the percentage of CPU used by each enclave in the displayed interval. We use an interval of 100 seconds on our system, so the percentage CPU is also conveniently the number of CPU seconds. The *TCPU* column shows the total CPU used by each enclave since the enclave first started. When you have long-running enclaves, they can span several reporting intervals, though the enclave name will change, since it is an artificial name. However, when you see an enclave with TCPU significantly greater than the CPU for the interval, then, logically the enclave must have been active for at least one previous interval. One can find, for example, an enclave with a total of 50 seconds of CPU time, and just 2 seconds consumed in this interval. This means that at the end of the previous interval, the enclave should have had a total of 48 seconds of CPU time. If one looks closely, though, one may not find the same enclave reported for several intervals prior to the last one – well, maybe it did not do anything in those missing intervals. However, when one does find the same enclave, say 10 intervals previously, the total CPU time is usually much less than is expected, say 20 seconds. So, sometime between the two available displays, another 12 seconds is not being reported. After some investigation it was found that the RMF monitor will display data only for the enclaves that had a status of ACTIVE at the LAST sampling in the interval. Search for APAR OA09061, which addresses the problem. In addition, take a look at APAR OA05961, which describes the situation you may have seen already: when you change the service class of an enclave (eg via SDSF), the enclave disappears from the RMF Monitor III enclave report, although it is still active on the system, until it is reset back to the original service class.

## COLLECTING ENCLAVE RESOURCE DATA AND REPORTING

The accounting for resources consumed by an enclave depends on whether it is an independent, dependent, or foreign enclave. The SMF type 30 record provides resource consumption data at the address space level. The Processor Accounting Section, Performance Section, and I/O activity section in z/OS contain information on pre-emptible-class SRB and enclave usage on behalf of the address space. No equivalent of the SMF type 30 record is made for enclaves; all recording of enclave service is done via the owning address space.

The following existing fields in the SMF type 30 record have been modified:

- *Smf30Cpt* used to contain TCB CPU time. It now contains the sum of TCB, enclave SRB, pre-emptible SRB, and client SRB CPU time. To compute TCB time subtract Smf30Enc and Smf30Asr.
- *Smf30Csu* contains CPU service. CPU service now contains TCB service as it used to and new components for pre-emptible SRB and client SRB CPU service. There is no corresponding Smf30 field containing only the new components that can be subtracted to isolate TCB service. Service consumed by enclaves created by the address space is not included.
- *Smf30Srv* contains total service, the sum of CPU, SRB, I/O, and MSO service. Total service now contains its old value and new components for pre-emptible SRB and client SRB CPU service. There is no corresponding Smf30 field containing only the new components that can be subtracted to isolate TCB service. Service consumed by enclaves created by the address space is not included.

Smf30Asr **(**pre-emptible SRB and client SRB CPU time) is a new field in the SMF type 30 record that is related to client SRBs and pre-emptible SRBs. It is non-zero if pre-emptible

SRBs run in this address space or if this address space was a client on whose behalf client SRBs were run.

The following are new fields in the SMF type 30 record that are related to enclaves. These fields contain the sums for all completed and existing enclaves created by an address space. They will be non-zero only for address spaces that own enclaves.

- Smf30Eta – enclave transaction active time.
- Smf30Esu – enclave CPU service.
- Smf30Etc – enclave transaction count.
- Smf30Enc – enclave total CPU time.
- Smf30eic – DASD I/O connect time for an independent enclave.
- Smf30eid – DASD I/O disconnect time for an independent enclave.
- Smf30eiw – DASD I/O disconnect time + CU queue time for independent enclave.
- Smf30eis – DASD I/O start subchannel for an independent enclave.

As already said, a dependent enclave is a logical continuation of the transaction already active in a client's address space. Therefore, CPU and MSO service for a dependent enclave is included in the SMF 30 record of the owning address space. MSO service for the enclave is calculated based on the frame count of the owning address space, not on frame usage in the address space(s) where the enclave is executing.

For an independent enclave, CPU service is included in the SMF 30 record of the owning address space. MSO service is not calculated for an independent enclave.

For both dependent and independent enclaves, IOC service is included in the SMF 30 records associated with the address

space where the enclave work is executing. SRB service for enclaves is always zero.

For a foreign enclave, CPU time is included in the SMF 30 record of the owning address space (owner) on the originating system. It is reported separately from local CPU time. Since CPU time used by foreign enclaves is included in the owner's SMF 30 record, it is not included in the SMF 30 records on the other systems where it actually executed. In order for those other systems to have some record of the CPU time used by foreign enclaves, an SMF 97 record is written for each SMF global recording interval. This SMF 97 record identifies the CPU time used by foreign enclaves during that interval, broken down by originating system.

An installation can review the originating system's SMF 30 records to identify the specific jobs that consumed the CPU time in the foreign enclaves.

Note that because data is collected asynchronously for the SMF 30 records, and because SMF intervals can vary from system to system, it may not be possible to exactly match SMF 30 times with SMF 97 times from one global interval to another.

A detailed description of layout of SMF type 30 and 97 records can be obtained from the *MVS System Management Facilities (SMF*) – SA22-7630-03 manual.

*Editor's note: a description of the program and the code will be published next month.*

*Mile Pekic*
*Systems Programmer (Serbia and Montenegro)*


Please note that the correct contact address for Xephon Inc is PO Box 550547, Dallas, TX 75355, USA. The phone number is (214) 340 5690, the fax number is (214) 341 7081, and the e-mail address to use is info@xephon.com.

Cybermation has announced Version 4.4 of ESP Espresso, its distributed enterprise job scheduling product.

The latest version is meant to provide more efficient job scheduling for enterprise applications, enhancing their ability to deal with critical data that must be tightly controlled.

The product provides a single enterprise-wide job scheduling product that integrates with Cybermation Service Manager, a real-time visualization reporting tool.

For further information contact:
URL: www.cybermation.com/newsroom/press/2005/pr03.html.

* * *

DataMirror has announced Version 4.7 of Transformation Server for z/OS, a real-time, bi-directional, mainframe data integration tool. The product allows organizations to synchronize and integrate mainframe data with information systems and business applications running on other platforms.

The product captures changes to data in a databases as they happen, flowing the data in real-time to other platforms or data stores. The new version has a static SQL mode of operation. Users with high-volume integration requirements should benefit most from the resulting improvements in product performance and throughput predictability.

For further information contact:
URL: www.datamirror.com/products/tserver/default.aspx.

* * *

Relational Architects International has announced Version 8.1 of Smart/RESTART.

The product is designed to makes it possible to restart-enable z/OS batch applications, often without changing the source code. It eliminates the need to back out and redo successfully processed work.

Failing batch applications can resume from a checkpoint rather than needing to be rerun from the beginning. Jobs can be resubmitted for restart without JCL changes.

For further information contact:
URL: www.relarc.com/smartrestart.

* * *

Softek has announced Versions 2.3 of Disaster Recovery Manager and Replicator. The new releases are designed to offer flexible, heterogeneous, and cost-effective data protection and recovery.

Softek DR Manager identifies critical files, including profiling interrelationships between files and applications, minimizing exposure caused by data corruption, and isolating critical batch operations for faster recovery.

Softek Replicator protects enterprise data through asynchronous host-based replication. Enhancements to Replicator for Windows, they claim, make it easier to install and dynamically configure while applications are running.

For further information contact:
URL: www.softek.com/en/press/20050525-001.html.

* * *