MVS update

197

February 2003

In this issue

MVS Update

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

North American office
Xephon
PO Box 350100
Westminster, CO 80035-0100
USA
Telephone: 303 410 9344

Subscriptions and back-issues
A year's subscription to *MVS Update*, comprising twelve monthly issues, costs £340.00 in the UK; $505.00 in the USA and Canada; £346.00 in Europe; £352.00 in Australasia and Japan; and £350.00 elsewhere. In all cases the price includes postage. Individual issues, starting with the January 1999 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.

Editor
Trevor Eddolls

Contributions
When Xephon is given copyright, articles published in *MVS Update* are paid for at the rate of £170 ($260) per 1000 words and £100 ($160) per 100 lines of code for the first 200 lines of original material. The remaining code is paid for at the rate of £50 ($80) per 100 lines. In addition, there is a flat fee of £30 ($50) per article. To find out more about contributing an article, without any obligation, please download a copy of our *Notes for Contributors* from www.xephon. com/nfc.

Edit macro to add a line after the cursor

This edit macro inserts an extra parameter line in an EXEC card in a job (and puts a comma on the preceding line). It is fairly simple, but this allows it to be adapted for many purposes. The line that is added is set in line 2.

To run it, simply put the macro in a library, which is in your SYSEXEC concatenation, type KKSM on the command line (do not press *Enter*), position the cursor on the line after which you want to have the line inserted, and then press *Enter*. The line will be inserted and a comma added to the end of the previous line. If you don't want the comma at the end, edit line 12 in the macro and delete the || ',' part.

There is code in place to deal with single and double quotes and ampersands on the line where you put the cursor – if there are any other special characters it should be fairly easy to change the code!

```
/* rexx */
/****************************************************************/
/* This exec inserts a line (lin2) after the cursor, and puts  */
/* a comma at the end of the preceding line.                    */
/* It can handle single or double quotes and ampersands in the */
/* preceding line.                                              */
/****************************************************************/
'ISREDIT MACRO (XX)'
'ISREDIT (row,col) = CURSOR'
'ISREDIT (STM) = line .zcsr'
stm = substr(stm,1,72)
stm = strip(stm,T,' ') || "," /* This is where we add the comma */
/* Change single, double quotes and ampersands to non disp chars */
stm = translate(stm,'DD'X,"'")
stm = translate(stm,'DF'X,'"')
stm = translate(stm,'64'X,'&')
col = length(stm) + 1
'ISREDIT cursor = (row,col)'
rowd = row
val1 = "'ISREDIT LINE_AFTER .ZCSR="
val2 = stm || '"'
interpret val1 '"'val2 "'"
row = row + 1
/* This is the row we want to insert after the cursor. */
```

```
val1 = "'ISREDIT LINE_AFTER" row "="
lin2 = "//         REGION=32M"
interpret val1 '"'lin2 '"' "'"
'ISREDIT delete' rowd
row = rowd
/* Now change all the non display chars back to single, */
/* double quotes and ampersands. */
col = 1 /* Needed for the following changes to work. */
'ISREDIT cursor = (row,col)'
"ISREDIT change X'DD' X'7D' "
"ISREDIT change X'DF' X'7F' "
"ISREDIT change X'64' X'5Ø' "
col = 1
'ISREDIT cursor = (row,col)'
```

C Leonard
Freelance Consultant (UK)

Monitoring job run status

THE PROBLEM

There is often a need to find out whether a particular job ran at a specified time or not. For example, in our mission-critical production environment, we open ‘P’ class initiators between 6:00pm and 8:00am the next morning for our production jobs to run. During the day time, our application users submit jobs under the assumption that these jobs will run after 6:00pm. What if for some reason the job which closes the ‘P’ class initiator did not run at 08:00am? All the production jobs will start to run as soon as the application users submit the job. Normally, these jobs might be scheduled for submission from an automatic job submission product such as CA-7 or Tivoli OPC/ESA. If these products abend or fail to submit the job, and if you do not have the mechanism to notify the system programmers, this may have a big impact on the real-time production environment. The simple solution might be to give instructions to the operators to monitor this particular job at a given time. However, what happens if the operator fails to notice that a job did not come up on time? If there is an automated mechanism to monitor and notify the systems programmers, or an attention message is sent to the operator console, the necessary action can be taken at the earliest possible opportunity.

A SOLUTION

The following REXX routine, which uses IOF, solves the above problem. This routine reads the input file, CHECK.RUN.JOB, to find out the jobs to be monitored. In this input dataset we can specify how long to monitor for a specific job, and we can specify to monitor only during the weekdays, weekends, or all the time. In the example given below, the PRODINIT job should have run between 07:50am and 08:00am only during the week. If it did not run during the above period, it would send a notification. However, there may be a need to monitor a set of jobs all the time. For this, the start time and end time need to be 00:00. Then, this REXX routine will assume it has to be monitored all the time. The sample input file has a detailed description for every column.

Two kinds of notification can be achieved with this routine if any job did not run during the time specified in the input dataset. One is a highlighted attention console message to the operator. There should be instructions to the operator about whom to contact if a specific job did not run or about what action should be taken. In order to achieve this, our REXX routine will call an Assembler routine (WTOREXX) and pass a message such as 'JOB PRODINIT did not run in system. Please check'. The Assembler routine will place the above message on the operator console with a highlighted attention message to grab the operator's attention.

The other way of notifying staff is to send a pager message to the systems programmer's pager, if your environment has that facility. These instructions are commented out in the REXX routine and that can be modified as per your environment. We have software called TELALERT, which runs in a Unix environment; we pass the pager message with the REXEC (Remote Execute) command.

This REXX routine can be run in batch – sample JCL is also provided. Whenever there is a message or page from this routine, it will record an entry with the date and time in the log dataset. This is just for future reference to find out which JOB or STC failed or was not up at a specific time.

With the help of a JES2 automatic command you can run this REXX routine in batch every 15 or 30 minutes, depending on your needs. A sample JES2 automatic command to run every 15 minutes would be:

```
$TAAØØ1,I=3ØØ,'$VS,''S CHECKRUN'''
```

If you want to run at a specified time, use the following JES2 command. In our example, we set it to run at 08:00am every day:

```
$TAAØØ1,T=Ø8.ØØ,'$VS,''S CHECKRUN'''
```

OPERATIONAL ENVIRONMENT

Use of this program is dependent on the correct customization of IOF, and the Assembler routine must have been compiled and link edited in dataset CHECK.RUN.LOADLIB. The sample JCL procedure has to be placed in any one of the JES2 procedure libraries with a proper user-id, which has a privilege to access the input and log datasets. The IOF minimum release should be 7C.

ASSEMBLER PROGRAM (WTOREXX):

```
         EJECT
TITLE 'WRITE TO CONSOLE FROM REXX PROGRAM'
WTOREXX  CSECT
         USING WTOREXX,R12
         STM   R14,R12,12(R13)
         LR    R12,R15
         ST    R13,SAVEAREA+4
         LA    R1Ø,SAVEAREA
         ST    R1Ø,8(R13)
         LR    R13,R1Ø
         B     MAINLINE
         DC    CL1Ø'WTOREXX'
         DC    CL1Ø'&SYSDATE'
         DC    CL1Ø'&SYSTIME'
MAINLINE DS    ØH
         LR    R2,R1
         L     R3,16(R2)
         L     R8,2Ø(R2)
         L     R8,Ø(R8)
         USING EVALBLOCK,R8
         LR    R2,R3                    ARGLIST
MOVMSG   LA    R6,MSG
         L     R3,4(R2)                 LENGTH
         L     R2,Ø(R2)                 MESSAGE
         LR    R7,R3
         STH   R3,MSGL
         MVCL  R6,R2
         LA    R7,MSGL
         WTO   TEXT=(R7),DESC=1     DESCRIPTOR HIGHLIGHTS ON CONSOLE
         MVC   EVALBLOCK_EVLEN,=F'1'
         MVC   EVALBLOCK_EVDATA,=C'Ø'
*-----------------------------------------------------------------*
RETURN   DS    ØH
         L     R13,SAVEAREA+4    RESTORE R13
         LM    R14,R12,12(R13)   RESTORE R14 TO R12
         XR    R15,R15           ZERO RETURN CODE REG
         BR    R14               RETURN
*-----------------------------------------------------------------*
* EQUATES                                                         *
*-----------------------------------------------------------------*
RØ       EQU   Ø
R1       EQU   1
R2       EQU   2
R3       EQU   3
R4       EQU   4
R5       EQU   5
R6       EQU   6
R7       EQU   7
R8       EQU   8
R9       EQU   9
R1Ø      EQU   1Ø
R11      EQU   11
R12      EQU   12
R13      EQU   13
R14      EQU   14
R15      EQU   15
*-----------------------------------------------------------------*
SAVEAREA DC    18F'Ø' ADDRESSED BY REG 13
         EJECT
*-----------------------------------------------------------------*
MSGL     DS    H
MSG      DS    CL2ØØ
*
*-----------------------------------------------------------------*
*        EVALBLOCK DSECT                                          *
*-----------------------------------------------------------------*
         IRXEVALB
         END   WTOREXX
```

REXX ROUTINE (CHECKRUN) :

```
/* rexx */
parse source . . myname . . . . . myenv .
say myname myenv
/* - - checkrun - Checks whether jobs were run or not.            */
/*                If not, sends highlighted WTO message to        */
/*            the operator console. Or this routine can be        */
/*            modified to send a pager message to the             */
/*            systems programmer.                                 */
/* General Description: This routine is designed to run in a TSO  */
/* batch environment to find out if a particular job was          */
/* run or not. If not, it alerts the IBMSYS on-call person        */
/* One physical sequential 'PUBLIC.RUN.LOG' dataset with LRECL=8Ø */
/* should be created for a log.                                   */
startmsg = 'Job '
endmsg = ' did not run in System. Please check up'
trace all
if myenv ¬= 'IOF' then do
  "IOF *.%"myname
  exit
end
else do
  Address "TSO"
  "alloc fi(run) da('PUBLIC.RUN.JOB') shr reu"
  "execio * diskr run (stem jobs. finis"
  "alloc fi(log) da('PUBLIC.RUN.LOG') mod reu"
  noofjobs = jobs.Ø
  do i = 1 to noofjobs
    ADDRESS IOF
    "H"
    " "
    iscomment = substr(jobs.i,1,1)
    if iscomment = '*' then iterate
    iofjobname = strip(substr(jobs.i,2,8))
    iofweekend = substr(jobs.i,11,3)
    iofstarttime = substr(jobs.i,15,5)
    iofendtime = substr(jobs.i,21,5)
    iofstarthh = substr(iofstarttime,1,2)
    iofstartmm = substr(iofstarttime,4,2)
    iofstarthrs = (iofstarthh * 6Ø) + iofstartmm
    iofendhh = substr(iofendtime,1,2)
    iofendmm = substr(iofendtime,4,2)
    iofendhrs = (iofendhh * 6Ø) + iofendmm
        if (iofweekend = 'SAT') then
        do
          select
            when currentday = 'Monday' then iterate
            when currentday = 'Tuesday' then iterate
            when currentday = 'Wednesday' then iterate
            when currentday = 'Thursday' then iterate
            when currentday = 'Friday' then iterate
            when currentday = 'Sunday' then iterate
            otherwise
               say "Today is Saturday"
          end         /* for Select Command */
        end           /* For IF in Week End checking */
        if (iofweekend = 'SUN') then
        do
          select
            when currentday = 'Monday' then iterate
            when currentday = 'Tuesday' then iterate
            when currentday = 'Wednesday' then iterate
            when currentday = 'Thursday' then iterate
            when currentday = 'Friday' then iterate
            when currentday = 'Saturday' then iterate
            otherwise
               say "Today is Sunday"
          end         /* for Select Command */
        end           /* For IF in Week End checking */
        if (iofweekend = 'DAY') then
        do
          select
             when currentday = 'Saturday' then iterate
             when currentday = 'Sunday' then iterate
             otherwise
                say "Today is Week Day"
          end   /* for Select command  */
        end
        "pre "iofjobname
        "extend on"
 "TSICOPY NAME(JOBNAME ran) TO(REXX) VARNAME(JNAME jran)"
        say jname
        say jran
        if length(janme) <> Ø then
        do
         jhrs = Ø
         if (iofstarthrs <> Ø ) & (iofendhrs <> Ø) then
         do
          jhh = substr(jran,1,2)
          jmm = substr(jran,4,2)
          say 'jhh is :' jhh
          say 'jmm is :' jmm
          jhrs = (jhh * 6Ø) + jmm
         end
        if ( (jhrs <  iofstarthrs) | (jhrs >  iofendhrs) ) then
         do
           msg1 = startmsg||iofjobname||endmsg
           Address "TSO"
/*This same routine can be used to send pager message to the concerned*/
/*systems programmer. Do uncomment the following three lines to send a*/
/*page from the server where you run your paging software. In our case*/
```

```
/*we run the paging software (TELALERT) on one of our Unix systems. We*/
/* use REXEC command to send a page from the mainframe.              */
/*        host = 'HOSTNAME'                                          */
/*       pager = 'PAGERID'                                           */
/*"REXEC -l loginid -p password "HOST" telalertc -g "pager" -m "msg1  */
/*                                                                   */
/*          CALL THE WTOREXX ASSEMBLER ROUTINE TO GET HIGHLIGHTED     */
/*          MESSAGE ON CONSOLE                                       */
            CALL WTOREXX(msg1)           "REXEC -l paging -p uupaging
"HOST" telalertc -g "pager" -m "msg1
           logmsg = date()||' '||time()||' '||msg1
           push logmsg
           "execio 1 diskw log"
         end
        end
        jname = ''
        jran  = ''
    /* end     For If checking on time */
    end    /* for DO loop */
 Address "TSO"
 "execio * diskw log (finis"
 "free file(log)"
 "free file(run)"
 Address IOF
 "exit"
exit
```

SAMPLE INPUT FILE FORMAT (CHECK.RUN.JOB)

```
*  '*' AT FIRST COLUMN IN COMMENT
*  COL.POSTITION
*  2-9   : JOB NAME
*  11-13 : DAY - WEEK DAY, SAT - SATURDAY, SUN - SUNDAY, ALL - ALL DAY
*  15-19 : START RUNNING TIME   NOTE TIME FORMAT ALWAYS : HH:MM (24 HRS)
*        IF IT CAN RUN AT ANY TIME DURING THE DAY, SPECIFY ØØ:ØØ IN BOTH
*        START AND END TIME
*  21-25 : END RUNNING TIME
 PRODINIT DAY Ø7:5Ø Ø8:ØØ
 DEVINIT   DAY Ø9:ØØ 1Ø:ØØ
```

JCL PROC TO RUN THE ABOVE REXX PROGRAM IN BATCH

```
//PROC  CHECKRUN
//STEP1    EXEC PGM=IKJEFTØ1
//STEPLIB    DD DSN=CHECK.RUN.LOADLIB,DISP=SHR
//SYSPRINT   DD SYSOUT=*
//SYSIN      DD DUMMY
//SYSOUT     DD DUMMY
```

```
//SYSTSPRT   DD SYSOUT=*
//ACTIVE     DD DSN=CHECK.RUN.JOB,DISP=SHR
//LOG        DD DSN=CHECK.RUN.LOG,DISP=SHR
//SYSTSIN    DD *
  CHECKRUN
*/
//  PEND
```

Muthukumar Kannaiyan
Systems Programmer (USA)

zSeries File Systems

INTRODUCTION

The zSeries File System (zFS) is a new Unix file system that can be used in addition to the Hierarchical File System. The zFS file system is different from HFS; for example, zFS file systems are created and formatted in a different manner from HFS file systems.

However, the application view of zFS is the same as the application view of HFS. The same APIs and commands are used for zFS as are used for HFS. Once the zFS file system is mounted it is almost indistinguishable from a mounted HFS file system.

zFS is not a replacement for HFS. HFS is still required for your root file system. zFS, whose performance is better than that of HFS, especially for frequently accessed files larger than 8KB, can be used in addition to HFS.

zFS code was not available initially for z/OS 1.2. To enable zFS support, you should install APARs OW50850 (UW82925) and OW51563 (UW83377). The zFS support on OS/390 2.10 and z/OS 1.1 is provided by APAR OW51780.

This article describes this new kind of file system and will present ways in which zFS administration is different from HFS administration.

zFS is installed as part of the z/OS Distributed File Service (DFS) base element. zFS introduces some new terms and file system structures that you should know:

- zFS physical file system.
- zFS file system.
- zFS file system aggregates.

ZFS physical file system

zFS is a physical file system (PFS) that is started by Unix System Services during IPL.

A physical file system is the part of the operating system that handles the actual storage and manipulation of data on a storage medium.

In order to start the zFS physical file system, you should:

1 Add SYS1.SIOELMOD to APF list and linklist.

2 Update SYS1.PARMLIB(BPXPRM00):

```
/****************************/
/* ZFS                      */
/****************************/

FILESYSTYPE TYPE(ZFS) ENTRYPOINT(IOEFSCM)
        ASNAME(ZFS)
```

The ASNAME parameter controls the name of the zFS STC.

3 Then add the zFS procedure to SYS1.PROCLIB:

```
//ZFS     PROC
//ZFSGO   EXEC PGM=BPXVCLNY,TIME=NOLIMIT,REGION=ØM
//IOEZPRM DD DISP=SHR,DSN=SYS1.PARMLIB(IOEZFSØØ)
```

The ZFS STC should be associated with a RACF userid with an OMVS segmentuid(0):

```
$HASP1ØØ ZFS      ON STCINRDR
$HASP373 ZFS      STARTED
IEF4Ø3I ZFS - STARTED - TIME=14.26.53
```

```
IOEZØØØ52I zFS kernel: Initializing z/OS    zSeries File System 872
Version Ø1.Ø2.ØØ Service Level OW53952.
Created on Wed Mar 27 17:Ø2:36 EST 2ØØ2.
IOEZØØ178I SYS1.PARMLIB(IOEZFSØØ) is the 873
configuration dataset currently in use.
IOEZØØØ55I zFS kernel: initialization complete.
```

zFS can be stopped by using the P ZFS operator command. When zFS is stopped, you receive the following message:

```
nn BPXFØ32D FILESYSTYPE ZFS TERMINATED. REPLY 'R' WHEN READY TO RESTART.
REPLY 'I' TO IGNORE.
```

zFS file system

zFS does not replace HFS, rather it is complementary to it. HFS is required for z/OS installation and the root file system must be HFS.

Like HFS, zFS is a Unix file system. It contains files and directories that can be accessed with the APIs available for HFS.

Like HFS, zFS can be mounted into the z/OS Unix System Services file hierarchy.

In general, the application view of zFS is the same as the application view of HFS. Once a zFS file system is mounted, it is almost indistinguishable from any other mounted HFS.

The benefits of using zFS are:

- Improved performance
- Improved crash recovery.

ZFS file system aggregates

zFS supports the use of 'aggregates'. A zFS aggregate is an MVS dataset containing one or more zFS file systems. The aggregate is actually a VSAM Linear Dataset (VSAM LDS), which is a container.

Thera are two kinds of zFS aggregate:

- HFS compatibility mode aggregates – this type of aggregate can contain only one zFS file system.

- Multi-file system aggregates – this type of aggregate can contain one or more zFS file systems.

Compatibility mode aggregates

A zFS aggregate that contains exactly one single zFS file system is called a 'compatibility mode aggregate'. This is flagged in the aggregate when it is created.

The name of the file system is the same as the name of the aggregate, which is the same as the VSAM LDS cluster name. The file system quota in a compatibility mode aggregate is set to the size of the aggregate. Compatibility mode aggregates are more like an HFS dataset, except they are VSAM linear datasets instead of HFS datasets.

Creating a zFS compatible aggregate

In order to create a zFS compatible aggregate you should run a two-step batch job. The first step will allocate a VSAM LDS cluster and the second step will use the IOEAGFMT utility to format the zFS with the -compat option.

The IOEAGFMT is a stand-alone utility that does not require the zFS physical file system to be active. The JCL to allocate a compatible aggregate is:

```
//STEPØ1  EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=R
//*
//SYSIN    DD *
  DELETE SYSTEMØ1.TESTAAAA.OMVS.ZFS.COMPAT
  SET MAXCC = Ø
  DEFINE CLUSTER -
         (NAME(SYSTEMØ1.TESTAAAA.OMVS.ZFS.COMPAT) -
          LINEAR CYL(Ø1Ø ØØØ) SHAREOPTIONS(2))
/*
//STEPØ2    EXEC PGM=IOEAGFMT,
// PARM=('-aggregate SYSTEMØ1.TESTAAAA.omvs.zfs.compat -compat')
//SYSPRINT  DD SYSOUT=*
//STDOUT    DD SYSOUT=*
//STDERR    DD SYSOUT=*
//*
```

The IOEAGFMT will then produce the following report:

```
IOEZØØØØ4I Loading dataset 'SYSTEMØ1.TESTAAAA.omvs.zfs.compat'.
IOEZØØØØ5I Dataset 'SYSTEMØ1.TESTAAAA.omvs.zfs.compat' loaded
successfully.
*** Using default initialempty value of 1.
*** Using default number of (8192-byte) blocks: 899
*** Defaulting to 13 log blocks(maximum of 1 concurrent transactions).
Done.  /dev/lfs1/SYSTEMØ1.TESTAAAA.omvs.zfs.compat is now a zFS
aggregate.
IOEZØØØ71I Attaching aggregate SYSTEMØ1.TESTAAAA.OMVS.ZFS.COMPAT to
create hfs-compatible file system
IOEZØØØ74I Creating file system of size 6335K, owner id Ø, group id 1,
permissions x1ED
IOEZØØØ48I Detaching aggregate SYSTEMØ1.TESTAAAA.OMVS.ZFS.COMPAT
IOEZØØØ77I HFS-compatibility aggregate SYSTEMØ1.TESTAAAA.OMVS.ZFS.COMPAT
has been successfully created
```

Mounting a zFS compatible aggregate

Once the zFS compatible aggregate is created, it is not necessary to attach it. A compatibility mode aggregate is more like an HFS and does not require an attach as a separate step. You can mount the corresponding file system with the TSO MOUNT command:

```
mount filesystem('SYSTEMØ1.TESTAAAA.OMVS.ZFS.COMPAT') type(zfs)
      mode(rdwr) mountpoint('/etc/zfs')
```

Multiple file system aggregates

A multiple file system aggregate can contain multiple zFS file systems. This makes it possible to do space sharing between the zFS file systems within the aggregate.

zFS implements space sharing. It means that if multiple file systems are stored in a single aggregate, when files are removed from one of the file systems, freeing DASD space, another file system can use that space when new files are created.

The maximum size of each filesystem in an aggregate is a logical limit, which is determined when the file system is created. This maximum size is called a ‘quota’.

The multiple file system aggregate has its own name. This name is assigned when the aggregate is created. It is always the same as the VSAM LDS cluster name.

Each zFS file system in the aggregate has its own file system name. This name is assigned when the particular file system in the aggregate is created.

Creating a zFS multi-file aggregate

In the same way, in order to create a zFS multi-file aggregate, you should run a two-step batch job.

The first step will allocate a VSAM LDS cluster and the second step will use the IOEAGFMT utility to format the zFS without any option.

The JCL to allocate a multi-file aggregate is:

```
//STEPØ1  EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=R
//*
//SYSIN    DD *
  DELETE SYSTEMØ1.TESTAAAA.OMVS.ZFS.MULTI
  SET MAXCC = Ø
  DEFINE CLUSTER -
         (NAME(SYSTEMØ1.TESTAAAA.OMVS.ZFS.MULTI) -
          LINEAR CYL(Ø1Ø ØØØ) SHAREOPTIONS(2))
/*
//STEPØ2    EXEC PGM=IOEAGFMT,
// PARM=('-aggregate SYSTEMØ1.TESTAAAA.omvs.zfs.multi')
//SYSPRINT  DD SYSOUT=*
//STDOUT    DD SYSOUT=*
//STDERR    DD SYSOUT=*
```

IOEAGFMT output looks like:

```
IOEZØØØØ4I Loading dataset 'SYSTEMØ1.TESTAAAA.omvs.zfs.multi'.
IOEZØØØØ5I Dataset 'SYSTEMØ1.TESTAAAA.omvs.zfs.multi' loaded
successfully.
*** Using default initialempty value of 1.
*** Using default number of (8192-byte) blocks: 899
*** Defaulting to 13 log blocks(maximum of 1 concurrent transactions).
Done.  /dev/lfs1/SYSTEMØ1.TESTAAAA.omvs.zfs.multi is now a zFS
aggregate.
```

Attaching a multi-file aggregate

zFS multi-file aggregates must be attached (opened) by zFS before they can be used. Since the file system is in the aggregate, the aggregate must be attached before the file system can be mounted.

Attach occurs in one of three ways:

- At IPL time, or when zFS is started, by putting the aggregate name in the zFS parameter file, IOEFSPRM:

```
BROWSE    SYS1.PARMLIB(IOEZFSØØ) - Ø1.Ø2       Line ØØØØØØØØ Col ØØ1 Ø8Ø
 Command ===>                                           Scroll ===> PAGE
***************************** Top of Data ******************************
auto_attach ON
user_cache_size(256M)
define_aggr R/W   attach cluster(SYSTEMØ1.TESTAAAA.OMVS.ZFS.MULTI)
**************************** Bottom of Data ****************************
```

- By using the IOEZADM utility:

```
//STEPØ1    EXEC PGM=IOEZADM,
// PARM=('attach -aggregate SYSTEMØ1.TESTAAAA.omvs.zfs.multi')
//SYSPRINT  DD SYSOUT=*
//STDOUT    DD SYSOUT=*
//STDERR    DD SYSOUT=*
```

This results in:

```
IOEZØØ117I Aggregate SYSTEMØ1.TESTAAAA.OMVS.ZFS.MULTI attached
successfully
```

- By issuing the zfsadm attach command from the USS shell:

```
J895288:/: >zfsadm attach -aggregate SYSTEMØ1.TESTAAAA.OMVS.ZFS.MULTI
IOEZØØ117I Aggregate SYSTEMØ1.TESTAAAA.OMVS.ZFS.MULTI attached
successfully
```

Detaching a multi-file aggregate

You can issue the zfsadm detach command from the USS shell:

```
J895288:/: >zfsadm detach -aggregate SYSTEMØ1.TESTAAAA.omvs.zfs.multi
IOEZØØ122I Aggregate SYSTEMØ1.TESTAAAA.OMVS.ZFS.MULTI detached
successfully
```

Defining a zFS file system in a multi-file aggregate

Once a multi-file aggregate has been attached, a zFS file system can be created in the aggregate:

```
//STEPØ1    EXEC PGM=IOEZADM,
// PARM=('create -filesystem FSØ1 -size 1ØØØ -aggregate SYSTEMØ1.baseprx
//             vs.omvs.zfs.multi')
//SYSPRINT  DD SYSOUT=*
//STDOUT    DD SYSOUT=*
//STDERR    DD SYSOUT=*

IOEZØØØ99I File system FSØ1 created successfully
```

The name of the file system is case-sensitive and must be upper-case.

An alternative is to use the zfsadm command from the USS shell to create the zFS file system:

```
J895288:/: >zfsadm create –filesystem FSØ2 -size 5ØØ –aggregate
SYSTEMØ1.TESTAAAA.OMVS.ZFS.MULTI
IOEZØØØ99I File system FSØ2 created successfully
```

Mounting the multi-file aggregate zFS file system

Once a zFS file system is created it can be mounted using the TSO MOUNT command:

```
//STEP1    EXEC PGM=IKJEFTØ1
//SYSPROC  DD DISP=SHR,DSN=SYS1.SBPXEXEC
//SYSTSPRT DD SYSOUT=*
//SYSTSIN  DD *
  oshell mkdir /etc/zfs
  mount filesystem(FSØ2) type(ZFS) +
        mode(rdwr) mountpoint('/etc/zfs')
//*
```

ZFSADM USS SHELL COMMAND/IOEZADM UTILITY

The zfsadm USS shell command and the IOEZADM batch program can be used to manage file systems and aggregates. Both the command and the program require the zFS physical file system to be up and running.

zfsadm USS shell command

The zfsadm command can be run as a shell command from the z/OS Unix System Services (USS) shell:

```
J895288:/: >zfsadm lsfs
IOEZØØ129I Total of 2 file systems found for aggregate
SYSTEMØ1.TESTAAAA.OMVS.ZF
S.MULTI
```

```
FSØ1             RW (Not Mounted)       9 K alloc      9 K quota On-line
FSØ2             RW (Mounted R/W)       9 K alloc      9 K quota On-line
Total file systems on-line 2; total off-line Ø; total busy Ø; total
mounted 1
```

IOEZADM batch utility

A sample of the JCL to run PGM=IOEZADM is supplied in SYS1.SIOESAMP(IOEZADM). This will define a zFS file system in a multi-file aggregate, using the IOEZADM batch utility:

```
//STEPØ1    EXEC PGM=IOEZADM,
// PARM=('create -filesystem FSØ1 -size 1ØØØ -aggregate SYSTEMØ1.baseprx
//               vs.omvs.zfs.multi')
//SYSPRINT  DD SYSOUT=*
//STDOUT    DD SYSOUT=*
//STDERR    DD SYSOUT=*
```

zfsadm/IOEZADM subcommands

The zfsadm command or the IOEZADM program use the same set of subcommands:

- attach – attach an aggregate
- apropos – display first line of help entry
- detach – detach an aggregate
- grow – grow an aggregate
- aggrinfo – obtain information on an attached aggregate
- clone – clone a filesystem
- clonesys – clone multiple filesystems
- create – create a filesystem
- delete – delete a filesystem
- help – get help on commands
- lsaggr – list aggregates
- lsfs – list filesystem information
- lsquota – list filesystem information

- quiesce – quiesce an aggregate
- rename – rename a filesystem
- setquota – set filesystem quota
- unquiesce – unquiesce an aggregate.

MOUNTING ZFS FILE SYSTEMS DURING THE IPL PROCESS

Because a BPXPRM*xx* MOUNT statement cannot be used to mount a zFS file system in the z/OS 1.2 release, a special configuration should be used to mount zFS file systems dynamically during IPL.

zFS file systems mounts can only be specified as USS mounts in */etc/rc*.

zFS physical file system configuration

If you have created and formatted a zFS multi-file system aggregate, you may add an entry in the SYS1.PARMLIB(IEOZFS00) file for the aggregate:

```
auto_attach ON
user_cache_size(256M)
define_aggr R/W   attach cluster(SYSTEMØ1.TESTAAAA.OMVS.ZFS.COMPAT)
define_aggr R/W   attach cluster(SYSTEMØ1.TESTAAAA.OMVS.ZFS.MULTI)
```

This causes the multi-file system aggregate to be attached when zFS is started during the IPL process.

/etc/rc commands

zFS file systems can be automatically mounted at IPL time by specifying them in the */etc/rc* file using an USS shell mount command, */usr/sbin/mount*:

```
/usr/sbin/mount -t ZFS -f SYSTEMØ1.TESTAAAA.OMVS.ZFS.COMPAT /etc/zfs/
compat
/usr/sbin/mount -t ZFS -f FSØ1 /etc/zfs/FSØ1
/usr/sbin/mount -t ZFS -f FSØ2 /etc/zfs/FSØ2
```

Growing a zFS aggregate

If a zFS aggregate becomes full, the administrator can grow the aggregate. He can cause secondary allocations to occur and format them to be part of the aggregate. There is no automatic grow mechanism in zFS. This is accomplished with the zfsadm grow command.

The aggregate's VSAM linear dataset must have a secondary allocation specified and space on the volume(s) must be available.

The size specified on the zfsadm grow command must be larger than the current size of the aggregate:

```
J895288:/: >zfsadm aggrinfo SYSTEMØ1.TESTAAAA.omvs.zfs.multi
SYSTEMØ1.TESTAAAA.OMVS.ZFS.MULTI (R/W MULT): 1159 K free out of total
1296 (136 reserved)
J895288:/: >zfsadm grow     SYSTEMØ1.TESTAAAA.omvs.zfs.multi 15ØØ
IOEZØØ173I Aggregate SYSTEMØ1.TESTAAAA.OMVS.ZFS.MULTI successfully grown
SYSTEMØ1.TESTAAAA.OMVS.ZFS.MULTI (R/W MULT): 18Ø7 K free out of total
1944 (2Ø8 reserved)
```

Aggregates cannot be made smaller.

zFS back-up/restore

A zFS aggregate can be backed up and restored using IDCAMS REPRO or DFSMS DSS.

The zFS aggregate must be quiesced before the back-up.

```
//*-------------------------------------------------------------
//* THIS JOB QUIESCES A ZFS AGGREGATE, DUMPS IT, THEN UNQUIESCES IT.
//*-------------------------------------------------------------
//*-------------------------------------------------------------
//* THIS STEP QUIESCES THE AGGREGATE.
//*-------------------------------------------------------------
//QUIESCE EXEC PGM=IOEZADM,REGION=ØM,
// PARM=('quiesce -aggregate SYSTEMØ1.TESTAAAA.omvs.zfs.multi')
//*
//SYSPRINT DD SYSOUT=*
//STDOUT DD SYSOUT=*
//STDERR DD SYSOUT=*
//*
//*-------------------------------------------------------------
```

```
//* THIS STEP DUMPS THE AGGREGATE.
//*-------------------------------------------------------------
//DUMP EXEC PGM=ADRDSSU,REGION=4Ø96K
//SYSPRINT DD SYSOUT=*
//OUT DD DSN=SYSTEMØ1.TESTAAAA.OMVS.ZFS.MULTI.DUMP,
// DISP=(NEW,CATLG,DELETE),SPACE=(CYL,(5,1),RLSE)
//SYSIN DD *
 DUMP DATASET(INCLUDE(SYSTEMØ1.TESTAAAA.OMVS.ZFS.MULTI)) -
 OUTDD(OUT) TOL(ENQF)
/*
/*
//*-------------------------------------------------------------
//* THIS STEP UNQUIESCES THE AGGREGATE.
//*-------------------------------------------------------------
//UNQUIES EXEC PGM=IOEZADM,REGION=ØM,
// PARM=('unquiesce -aggregate SYSTEMØ1.TESTAAAA.omvs.zfs.multi')
//*
//SYSPRINT DD SYSOUT=*
//STDOUT DD SYSOUT=*
//STDERR DD SYSOUT=*
//*
```

zFS cloning file system cloning

zFS cloning is a new function that makes a quick read-only copy of a zFS file system.

File system cloning is the ability to make a relatively quick read-only copy of a zFS file system that resides in the same aggregate as the original file system.

It is relatively quick because it does not copy the data blocks – it copies only the metadata. Metadata is file information like the owner and permission bit settings. This means that it is quick and does not take up too much space.

The metadata in the clone points to the same data blocks as the metadata in the original file system.

The clone file system is given the same name as the original except that .bak is appended to the name.

```
J895288:/: >zfsadm clone -filesystem FSØ1
IOEZØØ225I File system FSØ1 successfully cloned.
J895288:/: >zfsadm lsfs
IOEZØØ129I Total of 3 file systems found for aggregate
SYSTEMØ1.TESTAAAA.OMVS.ZF
S.MULTI
```

```
FSØ1             RW (Mounted R/W)       8 K alloc      9 K quota On-line
FSØ1.bak         BK (Not Mounted)       9 K alloc      9 K quota On-line
FSØ2             RW (Mounted R/W)       9 K alloc      9 K quota On-line
Total file systems on-line 3; total off-line Ø; total busy Ø; total
mounted 2

IOEZØØ129I Total of 1 file systems found for aggregate
SYSTEMØ1.TESTAAAA.OMVS.ZFS.COMPAT
SYSTEMØ1.TESTAAAA.OMVS.ZFS.COMPAT RW (Mounted R/W)       9 K alloc
9 K quota On-line
Total file systems on-line 4; total off-line Ø; total busy Ø; total
mounted 3
```

As soon as the clone operation is complete, the original file system is immediately available for update.

When the original file system is updated, new blocks are allocated for the updates but the original blocks also remain allocated and are still pointed to by the clone metadata.

Mount a zFS file system clone using zfsadm:

```
/usr/sbin/mount -t ZFS -r -f FSØ1.bak /etc/zfs/FSØ1_clone
```

You can also use the MOUNT TSO command to mount the clone file system. In this case, you should use a special syntax using three quotes around the file system name:

```
MOUNT FILESYSTEM('''FSØ1.bak''') MOUNTPOINT('/etc/zfs/FSØ1_clone')
```

To delete the zFS clone file system:

```
J895288:/: >zfsadm delete -filesystem FSØ1.bak
IOEZØØ1Ø5I File system FSØ1.bak deleted successfully
```

BIBLIOGRAPHY

SC24-5989 *Distributed File Service zFS Administration.*

SC24-5917 *DFS/SMB Messages and Codes.*

Systems Programmer (France) © Xephon 2003

Relink load module

PROBLEM ADDRESSED

There are occasions when an existing load module needs to be relinked, for example to assign a new alias. Unfortunately, there are also circumstances in which the usual method of invoking the Binder (or Linkage Editor) and including the existing load module fails – for example when an explicit ENTRY was specified for the original load module. There are also numerous other circumstances when the Binder changes the contents of the load module. This problem can be avoided by invoking the Binder directly by using the Binder Application Programming Interface and specifying that the created work module will only have ACCESS processing intent, that is, no Binder services will be invoked that change the size or structure of the load module.

BINDER APPLICATION PROGRAMMING INTERFACE (API)

The Binder API is a general-use programming interface described in the *DFSMS Program Management* manual. Two interfaces are provided – an Assembler macro interface and a call interface. The name is the same in both cases, IEWBIND.

The program below (RELINK) is written in C and uses the call interface (invoked dynamically). The C language is used for RELINK because of the availability of standard functions for parsing. C (rather than C++) is used because C++ does not support the fetch() function used to dynamically load the IEWBIND program.

CONTROL STATEMENTS

RELINK supports a subset of the control statements available for the Binder:

- INCLUDE – the source and name of the module to be relinked. The INCLUDE statement has the format:

```
INCLUDE ddname(module)
```

where *ddname* specifies the DD name of the library that contains the *module* to be relinked. Restriction: the INCLUDE statement can specify just one name.

- ALIAS (optional) – the new alias name to be assigned to the new relinked module. Restrictions: the ALIAS statement can specify just one name. Alias names are limited to eight characters.
- NAME – the name to be assigned to the new relinked module. An optional (R) can be appended to the module name to specify that any existing module with the same name will be replaced (default: do not replace). The relinked module is written to the library specified by the SYSLMOD DD statement.

Note: most of the restrictions mentioned above apply to the RELINK program implementation and are not IEWBIND restrictions. They have been made to simplify the program.

RELINK has an optional EXEC parameter, /t or /T, which specifies whether a processing trace is to be written to SYSPRINT. The processing trace lists the executed IEWBIND API service, its return code (decimal) and, should it return an error, the associated service reason code (hexadecimal).

Irrespective of the trace setting, an IEWBIND API service that issues a non-zero return code is always logged. Depending on the service concerned, some IEWBIND API service warnings (return code = 4) and errors (return code = 8) do not terminate the processing.

Because the access-only mode restricts the processing that can be performed on a load module, there are no useful Binder options that can be set (this also includes list options). For this reason, RELINK does not allow for any Binder options to be set.

DD STATEMENTS

RELINK requires the following DD statements:

- syslib – input object modules (library name specified in the INCLUDE syslib(module) statement).
- SYSIN – control statements (input): INCLUDE, ALIAS, NAME in Binder format.
- SYSLMOD – relinked load module (output).
- SYSPRINT – list output (both from RELINK and the Binder).
- libddn – library that contains the included modules. By convention, libddn is normally SYSLIB; the actual DD name is specified in the INCLUDE statement (INCLUDE libddn(modname), where libddn identifies the library from which the module with the name modname is to be included).

RETURN CODES

RELINK issues one of the following return codes:

0	OK.
8	SYSIN open error (RECFM=F, LRECL=80) (E002).
12	Processing error; a Binder service function returned an unexpected return code (E007).
16	No (new) NAME specified (E004).
20	Invalid control statement (E003).
24	Operand length error (E005).
28	IEWBIND fetch error (E006).

An explanatory error message is output to SYSPRINT for every non-zero return code.

INFORMATION MESSAGES

The information messages indicate that the associated RELINK process has been performed successfully. Information messages are issued only when the trace flag is set. The messages are:

I001 *function* RC: *rc*

I002 INCLUDE statement processed

I003 ALIAS statement processed

I004 NAME statement processed.

ERROR MESSAGES

E001 *function* RC: *retcode* RSNCODE: *rsncode*

The *function* IEWBIND API issued a non-zero return code *retcode* (decimal) with *rsncode* (hexadecimal) as reason code. The DFSMS Program Management manual explains the reason codes. They are:

- E002 SYSIN open error – DD:SYSIN cannot be opened.
- E003 Invalid control statement – RELINK accepts only the INCLUDE, ALIAS, and NAME control statements.
- E004 No new name specified – no NAME statement specified.
- E005 Length error: *text* – the length of the *text* operand is too long, for example, the maximum length of an alias name is eight characters.
- E006 IEWBIND fetch error – the Binder could not be loaded.
- E007 Maximum permitted service RC: *retcode* exceeded – the maximum permitted return code for an IEWBIND service has been exceeded. RELINK will be aborted.
- E008 RELINK aborted. RC: *rc* – RELINK has encountered a fatal error (return code *rc*).

PROGRAM CODE

```
/* RELINK */
/* Runtime option: /t = trace */
/* DD-statements: */
/*  SYSIN - Control statements (INCLUDE, ALIAS, NAME) */
/*          in Binder format. */
/*          Restrictions: The ALIAS statement can specify just one */
/*                        alias name. Alias names are restricted to */
/*                        8 characters. */
/*                        Names are restricted to 8 characters. */
/*  SYSPRINT - log (also Binder output listing) */
/*  SYSLMOD  - output load module */
/*  syslib   - input object modules (library name specified  */
/*             in the INCLUDE syslib(module) statement  */
/* Return codes: */
/*   Ø - OK */
/*   8 - SYSIN open error (RECFM=F, LRECL=8Ø) */
/*  12 - processing error; a Binder service function returned an
         unexpected return code */
/*  16 - no (new) NAME specified */
/*  2Ø - invalid control statement */
/*  24 - operand length error */
/*  28 - IEWBIND fetch error */
#pragma linkage (OSFUNC,OS)
typedef int OSFUNC();
#include <stdlib.h>
#include <string.h>
#include <stdio.h>
/* definitions */
#define VERSION 4 /* Binder version */
#define NAMELEN 8 /* maximum name length */
#define TRUE    1 /* true */
enum {INCLUDE = 1, ALIAS, NAME} op;
/* function prototypes */
void testrc(char *pfunct, int rc, int rsncode, int maxrc);
void fn_include(char *pdd, char *pmem);
void fn_alias(char *palias);
void fn_name(char *pname, char *popt);
void lengthError(char *pmsg);
void abend(int rc);
int trace = Ø; /* trace-mode = off (FALSE) */
typedef struct {
  short fc;
  short version;
} FC;
typedef struct {
  short len;
  char data[NAMELEN+1];
} VARCHAR;
struct LISTENTRY {
  char entryName[8];
  int  entryLen;
  char *pentry;
};
typedef struct {
  int countList;
  struct LISTENTRY listentry[1Ø];
} LIST;
VARCHAR ddlib = {6,"SYSLIB"};
VARCHAR ddlmod = {7,"SYSLMOD"};
VARCHAR member = {Ø};
VARCHAR aname = {Ø};
VARCHAR mname = {Ø};
VARCHAR noname = {Ø,""};
char repl = 'N';
int main(int argc, char* argv[])
{
  char kywd[256];
  char op1[256] = "";
  char op2[256] = "";
  char op3[256] = "";
  char op4[256] = "";
  char op5[256] = "";
  FILE *fp;
  char rec[256];
  int rc;
  int retcode;
  int rsncode;
  int version = 4;
  char dtoken[8];
  char workmod[8];
  OSFUNC * fptr; /* pointer to dynamically loaded function */
  /* function codes */
  FC startd = {1,VERSION};
  FC createw = {1Ø,VERSION};
  FC include = {4Ø,VERSION};
  FC adda = {3Ø,VERSION};
  FC savew = {8Ø,VERSION};
  FC deletew = {15,VERSION};
  FC endd = {5,VERSION};
  LIST filelist = {1};
  LIST exitlist = {Ø};
  LIST optlist = {Ø};
  VARCHAR parmstr = {Ø};
  /* Set tracing based on runtime option (default: trace off) */
  if (argc > Ø) {
    if (*argv[1] == 't') trace = TRUE;
    if (*argv[1] == 'T') trace = TRUE;
  }
  fp = fopen("DD:SYSIN","rb,recfm=fb,type=record");
  if (fp == Ø) {
    puts("EØØ2 SYSIN open error");
    abend(8);
  }
  for(;;) {
    int op;
    fread(rec,1,256,fp);
    rec[8Ø] = ØxØØ; /* truncate to 8Ø characters */
    if (trace) puts(rec);
    if (feof(fp) != Ø) break; /* EOF */
    /* Scan for keyword and operands. */
    /* Keyword and operands must be alphanumeric */
    /* (without any special characters) */
    sscanf(rec,"%s\
                 %[ABCDEFGHIJKLMNOPQRSTUVWXYZØ123456789]\
                 %c\
                 %[ABCDEFGHIJKLMNOPQRSTUVWXYZØ123456789]\
                 %c %s"\
                 ,kywd,op1,op2,op3,op4,op5);
    op = Ø; /* reset <op> */
    if (!strcmp(kywd,"INCLUDE")) op = INCLUDE;
    if (!strcmp(kywd,"ALIAS")) op = ALIAS;
    if (!strcmp(kywd,"NAME")) op = NAME;
    switch (op) {
      case INCLUDE:
        fn_include(op1,op3);
        break;
      case ALIAS:
        fn_alias(op1);
        break;
      case NAME:
        fn_name(op1,op3);
        break;
      default: /* invalid keyword */
        if (!trace) puts(rec); /* record not previously listed */
        puts("EØØ3 Invalid control statement");
        abend(2Ø);
    }
  }
  /* load the Binder */
  fptr = (OSFUNC *)fetch("IEWBIND");
  if (fptr == NULL)
  {
    puts("EØØ6 IEWBIND fetch error");
    abend(28);
  }
  /* start dialog */
  memcpy(filelist.listentry[Ø].entryName,"PRINT   ",8);
  filelist.listentry[Ø].entryLen = 8;
  filelist.listentry[Ø].pentry = "SYSPRINT";
  rc = (*fptr)(&startd,&retcode,&rsncode,dtoken,&filelist,&exitlist,
               &optlist,&parmstr);
  testrc("STARTD",rc,rsncode,4);
  /* create workmod */
  rc = (*fptr)(&createw,&retcode,&rsncode,dtoken,workmod,"A");
  testrc("CREATEW",rc,rsncode,4);
  /* include module */
  rc = (*fptr)(&include,&retcode,&rsncode,workmod,"NAME",
               &ddlib,&member,NULL,NULL,NULL,NULL,"YES","NO");
  testrc("INCLUDE",rc,rsncode,4);
  /* add alias */
  if (aname.len != Ø) {
    rc = (*fptr)(&adda,&retcode,&rsncode,workmod,&aname,
                 &noname,&noname,"A");
    testrc("ADDA",rc,rsncode,4);
  }
  /* save workmod */
  if (mname.len == Ø) {
    puts("EØØ4 No new name specified");
    abend(16);
  }
  rc = (*fptr)(&savew,&retcode,&rsncode,workmod,
               &ddlmod,&mname,&repl);
  testrc("SAVEW",rc,rsncode,8);
  /* delete workmod */
  rc = (*fptr)(&deletew,&retcode,&rsncode,workmod,"N");
  testrc("DELETEW",rc,rsncode,4);
  /* end dialog */
  rc = (*fptr)(&endd,&retcode,&rsncode,dtoken,"N");
  testrc("ENDD",rc,rsncode,4);
}
/* Test IEWBIND return code */
void testrc(char *pfunct, int rc, int rsncode, int maxrc) {
  char msgid[5];
  if (rc == Ø) {
    if (!trace) return; /* no trace requested */
    strcpy(msgid,"IØØ1");
    printf("%s %s RC: %d\n", msgid, pfunct, rc);
    return;
  }
  strcpy(msgid,"EØØ1");
  printf("%s %s RC: %d RSNCODE: %X\n", msgid, pfunct, rc, rsncode);
  if (rc <= maxrc) return; /* OK: accept error level */
  printf("EØØ7 Maximum permitted service RC: %d exceeded\n", maxrc);
  abend(12);
}
/* Process INCLUDE */
void fn_include(char *pdd, char *pmem) {
  if (trace) {
    puts("IØØ2 INCLUDE statement processed");
    puts(pdd);
    puts(pmem);
  }
  ddlib.len = strlen(pdd);
  if (ddlib.len > NAMELEN) lengthError("INCLUDE library");
  strcpy(ddlib.data,pdd);
  member.len = strlen(pmem);
  if (member.len > NAMELEN) lengthError("INCLUDE member");
  strcpy(member.data,pmem);
}
/* Process ALIAS */
void fn_alias(char *palias) {
  if (trace) {
    puts("IØØ3 ALIAS statement processed");
    puts(palias);
  }
  aname.len = strlen(palias);
  if (aname.len > NAMELEN) lengthError("ALIAS name");
  strcpy(aname.data,palias);
}
/* Process NAME */
void fn_name(char *pname, char *popt) {
  if (trace) {
    puts("IØØ4 NAME statement processed");
    puts(pname);
  }
  mname.len = strlen(pname);
  if (mname.len > NAMELEN) lengthError("NAME member");
  strcpy(mname.data,pname);
  if (*popt == 'R') repl = 'Y';
}
/* Handle length error */
void lengthError(char *pmsg) {
  printf("EØØ5 Length error: %s\n",pmsg);
  abend(24);
}
void abend(int rc) {
  printf("EØØ8 RELINK aborted. RC: %d\n\n", rc);
  exit(rc);
}
```

SAMPLE JCL

```
//         EXEC PGM=RELINK,PARM='/T'
//STEPLIB  DD DSN=loadlib,DISP=SHR
//SYSLIB   DD DSN=oldlib,DISP=SHR
//SYSLMOD  DD DSN=newlib,DISP=SHR
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
 INCLUDE SYSLIB(ASMCAFR)
 ALIAS AA
 NAME AN(R)
```

EXAMPLE OF A TRACE LOG (FOR THE ABOVE)

```
INCLUDE SYSLIB(ASMCAFR)
IØØ2 INCLUDE statement processed
SYSLIB
ASMCAFR
 ALIAS AA
IØØ3 ALIAS statement processed
AA
 NAME AN(R)
```

```
IØØ4 NAME statement processed
AN
 NAME AN(R)

IØØ1 STARTD rc: Ø
IØØ1 CREATEW rc: Ø
IØØ1 INCLUDE rc: Ø
IØØ1 ADDA rc: Ø
```

Systems Programmer
(Germany)   © Xephon 2003

Automating the defrag process and preparing user-friendly reports – part 2

This month we conclude the code for a suite of jobs that will automate the defrag process and create easy-to-read reports.

DEFRAGPB JES2 PROCEDURE

```
//HSMDFRGB PROC DSK=,FLAG=
//*-----------------------------------------------------------------*/
//*              AUTOMATED DEFRAG UTILITY (- Batch -)               */
//*                                                                 */
//* Procedure     : DefragPB                                        */
//* Called from   : DefragR2                                        */
//* Function      : Uses Adrdssu program to defragment the disk volume*/
//*                 of which name is retrieved from REXX DefragR2.    */
//*                                                                 */
//* Parameters: Dsk : Disk volume to be defragmented.               */
//*             Flag: When it's the last defragmented volume, Flg=1  */
//*                   Otherwise ..........................  Flg=Ø   */
//*-----------------------------------------------------------------*/
//STEP1    EXEC PGM=ADRDSSU
//IN       DD   VOL=SER=&DSK,UNIT=339Ø,DISP=OLD
//SYSPRINT DD   DSN=&&DFRG,DISP=(MOD,PASS),UNIT=SYSDA,
// SPACE=(CYL,(1,1)),DCB=(LRECL=131,BLKSIZE=135,RECFM=VA)
//SYSIN    DD   DISP=SHR,DSN=EXP.SMS.DEFRAG.SYSIN
//         DD   DISP=SHR,DSN=EXP.SMS.DEFRAG.SYSIN.EXCLUDE
//*-----------------------------------------------------------------*/
//* In this step the REXX DefragR3 is called. This REXX will read and */
//* format the Sysprint output of the disk that is defragmented by the*/
//* previous step (STEP1).                                          */
//*                                                                 */
//* The parameters DSK and FLAG are passed to the REXX DefragR3.      */
//* When the Flag is '1', then this means that the last disk was just */
//* defragmented. So "Defrag Summary Report" dataset will be sent to  */
//* some users via Smtp.                                              */
//*                                                                   */
//* &&DFRG: This is the Sysprint output of the Defrag process. It's   */
//*         used in the REXX DefragR3 by using file name "DENIZ".     */
//*-------------------------------------------------------------------*/
//STEP2    EXEC PGM=IKJEFTØ1,PARM='ISPSTART CMD(DEFRAGR3 &DSK &FLAG)'
//STEPLIB  DD   DISP=SHR,DSN=SYS1.DGTLLIB
//SYSPROC  DD   DISP=SHR,DSN=EXP.SMS.CNTL
//SYSEXEC  DD   DISP=SHR,DSN=SIS.EXEC
//ISPPLIB  DD   DISP=SHR,DSN=ISP.SISPPENU
//ISPSLIB  DD   DISP=SHR,DSN=ISP.SISPSENU
//ISPTLIB  DD   DISP=SHR,DSN=ISP.SISPTENU
//ISPMLIB  DD   DISP=SHR,DSN=ISP.SISPMENU
//         DD   DISP=SHR,DSN=ISP.SISPMENU
//ISPPROF  DD   DISP=(NEW,DELETE,DELETE),DSN=&&PROF,
// DCB=(ISP.SISPTENU),SPACE=(TRK,(1,1,1)),UNIT=SYSDA
//SYSTSPRT DD   SYSOUT=A
//ISPLOG   DD   SYSOUT=(,),DCB=(LRECL=125,BLKSIZE=129,RECFM=VA)
//SYSTSIN  DD   DUMMY
//*
//DENIZ    DD   DSN=&&DFRG,DISP=(OLD,DELETE)
//*
//         PEND
//*-------------------- End of Proc DefragPB -----------------------*/
```

DEFRAGR1 REXX EXEC

```
/* REXX */
/*-------------------------------------------------------------------*/
/*               AUTOMATED DEFRAG UTILITY (- Batch -)                */
/* REXX program : DefragR1                                           */
/* Called from  : JCL DefragJ                                        */
/* Function     : Generate an report which consists of disk volumes  */
/*                that need defragmentation process.                 */
/*                                                                   */
/* Datasets created                                                  */
/* ================                                                  */
/* Exp.Sms.Defrag.Report   : Defrag report                           */
/*                                                                   */
/* Datasets read                                                     */
/* =============                                                     */
/* Exp.Sms.Defrag.Dcollect : Dcollect dataset                        */
/*                                                                   */
/* Parameters passed from JCL DefragJ                                */
/* (Defrag threshold parameters)                                     */
/* ==================================                                */
/* Num1 : Fragmentation index     threshold                          */
/* Num2 : Disk free space percent threshold                          */
/*                                                                   */
/* NOTE: This REXX is adapted from the IBM REXX Acbqadr5, which is   */
/*       located in the Sys1.Dgtclib.                                */
/*-------------------------------------------------------------------*/
Arg Num1 Num2
Address "TSO"
"Prof nopref"
Sysn     = MvsVar('SYMDEF','SYSNAME')
"Newstack" /* Create a new stack */
Address "ISPEXEC"
V1  =  1
V2  =  2
V3  =  3
V4  =  4
V5  =  5
V6  =  6
V7  =  7
V8  =  8
V9  =  9
V1Ø =  1Ø
V11 =  11
V12 =  12
V13 =  13
V14 =  14
Address "TSO"
/*-------------------------------------------------------------------*/
/* Begin processing the DCOLLECT dataset.                            */
/*-------------------------------------------------------------------*/
/*-------------------------------------------------------------------*/
/* Set up constants                                                  */
/*-------------------------------------------------------------------*/
N        = 1                                /* row number            */
Eof      = 'no'
vtoc_enl = c2x('4Ø'x)
vol_priv = c2x('2Ø'x)
vol_publ = c2x('1Ø'x)
vol_stor = c2x('8Ø'x)
vol_shrd = c2x('Ø4'x)
bit7 = '8Ø'x
bit6 = '4Ø'x
bit5 = '2Ø'x
bit4 = '1Ø'x
bit3 = 'Ø8'x
bit2 = 'Ø4'x
bit1 = 'Ø2'x
bitØ = 'Ø1'x
bit76 = 'CØ'x
hexØ = 'ØØ'x
```

```
hex8Ø = '8Ø'x
hexØ3 = 'Ø3'x
hexØ1 = 'Ø1'x
hexfe = 'fe'x
hexff = 'ff'x
Do While(Eof = 'no')
  "Execio 1 Diskr Dcolin"
  IF rc <>  Ø Then
  Do
    Eof = 'yes'
    Leave
  End
  Else Do
  Parse PULL Dcolrec
  /*-----------------------------------------------------------------*/
  /* Check to see if this a V record?                                */
  /*-----------------------------------------------------------------*/
 If ((Substr(Dcolrec,5,1) = 'V') & (Substr(Dcolrec,5,2) <> 'VL')) Then
      Do
      /*-------------------------------------------------------------*/
      /* Parse V record into variables.                              */
      /*-------------------------------------------------------------*/
      Parse VAR Dcolrec 25 volser 31 . 36 spcpcnt 36 .
      Parse VAR Dcolrec 37 spckb 4Ø . 41 spceall 44 . 45 devcap 48 .
      Parse VAR Dcolrec 49 fragindx 52 . 53 lrgext 56 .
      Parse VAR Dcolrec 57 numexts 6Ø .
      Parse VAR Dcolrec 69 Devtype 76.
      Parse VAR Dcolrec 77 devnum 78 . 83 storgrp 91.
      Devtype = STRIP(Devtype,T,' ')
     /*-------------------------------------------------------------*/
     /* Assigning variables that are to be used in the Reports.      */
     /*-------------------------------------------------------------*/
      Spcpcnt   = c2d(Substr(Dcolrec,36,1)) /* Space percentage       */
      Spckb     = c2d(Substr(Dcolrec,37,4)) /* Space in kb.           */
      Spcall    = c2d(Substr(Dcolrec,41,4)) /* Space Allocated        */
      Devcap    = c2d(Substr(Dcolrec,45,4)) /* Device capacity        */
      Fragindx  = c2d(Substr(Dcolrec,49,4)) /* Fragm. index           */
      Lrgext    = c2d(Substr(Dcolrec,53,4)) /* Largest extent         */
      Numfexts  = c2d(Substr(Dcolrec,57,4)) /* Number of free extents */
      Devnum    = c2x(Substr(Dcolrec,77,2)) /* Device number          */
      Use_attr  = Substr(Dcolrec,31,1)      /* Use attribute          */
      Phystat   = Substr(Dcolrec,31,1)      /* Physical Status of vol.*/
      Indx_stat = Substr(Dcolrec,31,1)      /* Index status           */
/*-----------------------------------------------------------------------*/
/* Process Index Status in V record.                                     */
/*-----------------------------------------------------------------------*/
      Select
         When bitand(indx_stat,bit76) = bit76 Then Vtoc_Indx = 'ENABLED'
         Otherwise                                 Vtoc_Indx =
'DISABLED'
```

```
      End
/*-------------------------------------------------------------------*/
/* Process Devtype and Set Devcap, Lrgext, Spcall & Spckb variables.  */
/*-------------------------------------------------------------------*/
      If Devtype = '339Ø' Then devcap = (devcap*1Ø24)%(15*56664)
      If Devtype = '338Ø' Then devcap = (devcap*1Ø24)%(15*47476)
      If Devtype = '339Ø' Then spckb  = (spckb*1Ø24)%(15*56664)
      If Devtype = '338Ø' Then spckb  = (spckb*1Ø24)%(15*47476)
      If Devtype = '339Ø' Then lrgext = (lrgext*1Ø24)%(15*56664)
      If Devtype = '338Ø' Then lrgext = (lrgext*1Ø24)%(15*47476)
      If Devtype = '339Ø' Then spcall = (spcall*1Ø24)%(15*56664)
      If Devtype = '338Ø' Then spcall = (spcall*1Ø24)%(15*47476)
      If datatype(spcall) = 'NUM' Then
        Do
          /*---------------------------------------------------------*/
          /* Note that the module Acbfuto2 is located in Sys1.Dgtllib */
          /* dataset, which is allocated in the procedure DefragP1   */
          /*---------------------------------------------------------*/
           fmt4 = ACBFUTO2(volser)
           ds4devsz = Substr(fmt4,19,4)
           cyls = c2d(Substr(ds4devsz,1,2))-1
           trks_per_cyl = c2d(Substr(ds4devsz,3,2))
           ds4devtk = c2d(Substr(fmt4,23,2))
           Den = Ø
           If (Devtype = '333Ø' | Devtype = '335Ø' | Devtype = '3375' ),
             Then Den = 1
           If Devtype = '338Ø' Then Den = cyls/885
           If Devtype = '339Ø' Then Den = cyls/1113
           If Devtype = '338Ø' Then
             Do
                Select
                   When Den = 1 Then Devtype = Devtype'-D'
                   When Den = 3 Then Devtype = Devtype'-K'
                   Otherwise         Devtype = Devtype'-?'
                 End
             End
           If Devtype = '339Ø' Then
             Do
                Select
                   When Den = 1 Then Devtype = Devtype'-1'
                   When Den = 2 Then Devtype = Devtype'-2'
                   When Den = 3 Then Devtype = Devtype'-3'
                   When Den = 9 Then Devtype = Devtype'-9'
                   Otherwise         Devtype = Devtype'-?'
                 End
             End
       End
/*-------------------------------------------------------------------*/
/* Process Use Attribute in V Record.                                 */
/*-------------------------------------------------------------------*/
```

```
      If bitand(use_attr,bit5)=bit5 Then Attrib = 'PRIVATE'
      If bitand(use_attr,bit4)=bit4 Then Attrib = 'PUBLIC'
      If bitand(use_attr,bit3)=bit3 Then Attrib = 'STORAGE'
/*-------------------------------------------------------------------*/
/* Process Physical Status of Volume                                 */
/*-------------------------------------------------------------------*/
      IF bitand(physstat,hexØ1)=hexØ1 Then Status = 'CONVERT'
      IF bitand(physstat,hexØ3)=hexØ3 Then Status = 'MANAGED'
                           Else
                                 Do
                                   Status = 'NONSMS'; Storgrp = 'N/A'
                                 End
     End
End
/*-------------------------------------------------------------------*/
/* Format and print the record.                                      */
/*-------------------------------------------------------------------*/
   Record  = ' '
   Titlrec = ' '                       /* Title record for the report */
   /*----------------------------------------------------------------*/
   /* Start at 2nd position, 1st ch.is ASCII control for page control.*/
   /*----------------------------------------------------------------*/
   Recloc = 2
   Titloc = 2
 IF ((Substr(Dcolrec,5,1) = 'V') & (Substr(Dcolrec,5,2) <> 'VL')) Then
    Do
  Address "ISPEXEC"
  L1        = 8
  L2        = 1Ø
  L3        = 9
  L4        = 7
  L5        = 9
  L6        = 9
  L7        = 9
  L8        = 1Ø
  L9        = 9
  L1Ø       = 9
  L11       = 8
  L12       = 8
  L13       = 1Ø
  L14       = 11
  Order.V1  = ""volser" VOLSER"
  Order.V2  = ""vtoc_indx" INDXSTAT"
  Order.V3  = ""attrib" USEATTR"
  Order.V4  = ""spcpcnt" %FREE"
  Order.V5  = ""spckb" FREECYL"
  Order.V6  = ""spcall" SPACALL"
  Order.V7  = ""devcap" DEVCAP"
  Order.V8  = ""fragindx" $FRGINDX"
  Order.V9  = ""lrgext" LRGEXT"
  Order.V1Ø = ""numfexts" FREEXTS"
  Order.V11 = ""Devtype" Devtype"
  Order.V12 = ""devnum" DEVNUM"
  Order.V13 = ""storgrp" STORGRP"
  Order.V14 = ""status" DEVSTAT"
   Do J=1 to 14
     Ctr = Value('V'||J)
     Fldlen.Ctr = Value('L'||J)
   End
  Order.Ø   = 14
  Fldlen.Ø  = 14
     Do i=1 to Order.Ø
        Address "TSO";
        If Fldlen.i <> "Fldlen."i"" Then
          Do;
              Titlrec = Insert(Word(Order.I,2),Titlrec,Titloc)
              Titloc  = Titloc + Fldlen.i
          End
     End
     Do i=1 TO Order.Ø
        Address "TSO";
        If Fldlen.i <> "Fldlen."i"" Then
          Do;
            Record = Insert(Word(Order.I,1),Record,Recloc)
            Recloc = Recloc + Fldlen.i
          End
     End
End
      /*-------------------------------------------------------------*/
      /* We are choosing defrag-candidate volumes based on two values:*/
      /* Frag.index( Num1 ) and Free Space percentage ( Num2 ).       */
      /*                                                              */
      /* You can specify a different Fragm. index according to your   */
      /* needs. Note that If you keep it too low, you may have too    */
      /* many disks to defrag and this may affect your system's       */
      /* performance.                                                 */
      /*-------------------------------------------------------------*/
      If (Record <> '' & Fragindx > Num1 & Spcpcnt > Num2) Then
        Do
          Queue ""Record""
          "Execio 1 Diskw Report "
          N = N + 1
        End
END
/*-------------------------------------------------------------------*/
/* Write heading record for Report.                                  */
/*-------------------------------------------------------------------*/
Queue ""Titlrec""
"Execio 1 Diskw Report "
/*-------------------------------------------------------------------*/
```

```
/* Close Report and Dcollect dataset.                              */
/*-----------------------------------------------------------------*/
"Delstack"  /* Remove the stack */
"Execio Ø Diskw Report (Finis"
"Execio Ø Diskr Dcolin  (Finis"
/*-----------------------------------------------------------------*/
/* Sort the Report by fragm.index.                                 */
/*-----------------------------------------------------------------*/
Call Sortfile Exp.Sms.Defrag.Report  '(64,3,BI,D)'
EXIT Ø    /* End of REXX DEFRAGR1   */
/*=================================================================*/
/*=================================================================*/
SORTFILE:
Arg Pop Pip
"Alloc Fi(Sortin)   Da('"Pop"') Shr Reuse"
"Alloc Fi(Sortout)  Da('"Pop"') Shr Reuse"
"Alloc Fi(Sortlib)  Da('Sys1.Sortlib') Shr Reuse"
"Alloc Fi(sortwkØ1) Cylinders Space(5Ø,5) Unit(Sysda) Reuse"
"Alloc Fi(Sysout) Space(9,3) Track Lrecl(8Ø) Recfm(f) Blksize(8Ø) Reuse"
"Alloc Fi(Sysin)  Space(1,1) Track Lrecl(8Ø) Recfm(f) Blksize(8Ø) Reuse"
"Newstack" /* Create a new stack */
Sort_sysin = " SORT     FIELDS="pip
Queue Sort_sysin
"Execio 1 Diskw Sysin (Finis"
"Delstack"  /* Remove the stack */
"Call 'Sys1.Sortlpa(Sort)'"
If Rc <> Ø Then Say 'Sort is NOT successfull. Return Code= ' RC
"Free File(Sortin,Sortout,Sysin,SortwkØ1,Sortlib)"
RETURN   /* End-of-Sortfile */
/*=================================================================*/
/*------------------- End of member DefragR1 ---------------------*/
```

DEFRAGR2 REXX EXEC

```
/* REXX */
/*-----------------------------------------------------------------*/
/*              AUTOMATED DEFRAG UTILITY (- Batch -)               */
/* REXX program : DefragR2                                         */
/* Called from  : JCL DefragJ                                      */
/* Function     : Builds a dynamic JCL. (Job-name : EXPSMSØ2)      */
/*                Job steps of this dynamic JCL will call DefragPB */
/*                procedure to defragment a single disk volume.    */
/*                                                                 */
/* Datasets read                                                  */
/* =============                                                   */
/* Exp.Sms.Defrag.Report : Defrag report (sorted by frag.index)    */
/*                                                                 */
/* Datasets created                                               */
/* ================                                                */
/* Exp.Sms.Defrag.Summary: Defrag Summary Report to be sent via Smtp. */
/*--------------------------------------------------------------------*/
"Prof Nopref"
Repdset  = 'Exp.Sms.Defrag.Report'
Sumdset  = "Exp.Sms.Defrag.Summary"
Stadset  = "Exp.Sms.Defrag.Stats"
Sysn     = MvsVar('SYMDEF','SYSNAME')                  /* System name */

"Alloc Da("Repdset") Fi(Martapv) Shr Reuse"
"Execio * Diskr Martapv (Stem Record. Finis"
"Free Fi(Martapv)"
/*--------------------------------------------------------------------*/
/* Build Defrag Summary Report dataset.                               */
/*--------------------------------------------------------------------*/
Call Build_Sum
/*--------------------------------------------------------------------*/
/* Build Jcl to do batch defrag for various disks.                    */
/*--------------------------------------------------------------------*/
"Newstack"      /* Create a new stack */
Queue "//EXPSMSØ2 JOB MSGCLASS=Z,MSGLEVEL=(1,1),TIME=144Ø,"
Queue "//             NOTIFY=&SYSUID,CLASS=E"
Queue "//*"
Say " "
Say " "
If Record.Ø=1 Then Exit /* If there is no disk for Defrag, then exit. */
Cnt = Record.Ø-1       /* Number of Defrag-candidate disks           */
Say "The following "Cnt "disks will be defragmented:"
Vol_list = ''
Do i = 1 to Record.Ø-2
  Vol = Substr(Record.i,3,6)
  Vol_list = Vol ||","|| Vol_list
  Say i"-) " Vol
  Queue "//STEP"i "EXEC DEFRAGPB,DSK="Vol",FLAG="Ø
End
/*--------------------------------------------------------------------*/
/* Now, build the job step for the last disk volume & set the FLAG to */
/* '1'. So the REXX DefragR3, that is called from the DefragPB will   */
/* wrap up the "Defrag Summary Report" dataset and send it to some   */
/* users via SMTP.                                                    */
/*--------------------------------------------------------------------*/
P   = Record.Ø-1
Vol = Substr(Record.P,3,6)
  Vol_list = Vol_list || Vol
Say i"-) " Vol
Say " "
Say " "
Queue "//STEP"i  "EXEC DEFRAGPB,DSK="Vol",FLAG="1
/*--------------------------------------------------------------------*/
/* Submit the job EXPSMSØ2.                                           */
/*--------------------------------------------------------------------*/
```

```
Queue "//"
Queue "QQ"
"Submit * End(QQ)"
"Delstack"  /* Remove the stack */
/*------------------------------------------------------------------*/
/* Update "Defrag Statistics" dataset.                              */
/*------------------------------------------------------------------*/
Call Upd_Stats
Exit Ø  /* End-of-main-REXX */
/*==================================================================*/
/*==================================================================*/
BUILD_SUM:
Address Tso
Rs = Sysdsn(Sumdset)
If Rs="OK" Then Delete Sumdset
"Alloc Fi(Mylog) Space(1,1) Tracks Lrecl(1Ø4) Recfm(F,B) Blksize(Ø),
 Reuse Dsorg(Ps) New Catalog Da("Sumdset")"
"Free F(Mylog)"
/*------------------------------------------------------------------*/
/* Write the heading line.                                          */
/*------------------------------------------------------------------*/
Address Tso
"Alloc Da("Sumdset") F(Mylog) Mod"
"Newstack"   /* Create a new stack */
Date_Time = Date('E')||"  |  "||Time('C')
/*------------------------------------------------------------------*/
/* Write legend part of the report.                                 */
/*------------------------------------------------------------------*/
T1 = "VOLUME Largest_Free_Extent (Cyl) - Before Defrag"
T2 = "VOLUME Largest_Free_Extent (Cyl) - After  Defrag"
T3 = "Free Extents                     - Before Defrag"
T4 = "Free Extents                     - After  Defrag"
Queue"     SUMMARY OF DEFRAGMENTATION PROCESS IN THE LPAR -"Sysn"- "
Date_Time
Queue "       "
Queue "Volume    T1        T2        T3        T4    "
Queue "------    ------    ------    ------    ------"
If Record.Ø=1 Then
                Do
            Say   "There is no any candidate disk for Defrag process..."
            Queue "There is no any candidate disk for Defrag process..."
                End
Queue ""              /* Insert a null to indicate line mode is over */
"Execio * Diskw Mylog (Finis"
"Delstack"  /* Remove the stack */
"Free File(Mylog)"
Return /* End BUILD_SUM */
/*==================================================================*/
/*==================================================================*/
UPD_STATS:
```

```
/*-------------------------------------------------------------------*/
/* "Defrag Statistics" dataset record format :                       */
/*  DATE  -  Number of              -  All volsers of                */
/*           Defrag_candidate disks     Defrag_candidate disks       */
/*-------------------------------------------------------------------*/
Address Tso
"Alloc Da("Stadset") F(Stats) Mod"
"Newstack"   /* Create a new stack */
Rec = Date('E')||" - "||Cnt||" "||Vol_list
Queue Rec
Queue ""  /* Insert a null to indicate line mode is over */
"Execio * Diskw Stats (Finis"
"Delstack"  /* Remove the stack */
"Free File(Stats)"
Return /* End UPD_STATS */
/*===================================================================*/
/*-------------------- End of member DefragR2 ---------------------*/
```

DEFRAGR3 REXX EXEC

```
/* REXX */
/*-------------------------------------------------------------------*/
/*               AUTOMATED DEFRAG UTILITY (- Batch -)                */
/* REXX program : DefragR3                                           */
/* Called from  : JES2 procedure DefragPB                            */
/* Function     : This REXX reads the Sysprint output of each Defrag */
/*                process and formats it by using the edit macro     */
/*                DefragM.                                           */
/*                When this REXX is called for the last time,        */
/*                ( in this case Flg variable will be 1.)  "Defrag   */
/*                Summary Report" dataset will be sent to some users */
/*                via SMTP.                                          */
/* Parameters   : Vol : Volume of defragmented disk                  */
/*                Flg : When it's the last defragmented volume, Flg=1 */
/*                      Oterwise  ............................  Flg=Ø */
/* Datasets read                                                     */
/* ==============                                                    */
/* File_name=DENIZ      : Sysprint output of Adrdssu. It's referred   */
/*                        as &&DFRG in the procedure DefragPB.       */
/* Datasets updated                                                  */
/* ================                                                  */
/* Exp.Sms.Defrag.Summary : Defrag Summary Report dataset            */
/* Datasets created                                                  */
/* ================                                                  */
/* SYSTSPRT spool dataset: This dataset will have detailed Defrag    */
/*                          statistics report for each disk.         */
/*-------------------------------------------------------------------*/
Arg Vol Flg
Sumdset  = "Exp.Sms.Defrag.Summary"
```

```
Sysname  = MvsVar('SYMDEF','SYSNAME')   /* System name */
Date_Time   = Date('U')||"  |  "||Time('C')
/*------------------------------------------------------------------*/
/* Legend for "Defrag Summary Report" dset which will be sent via SMTP*/
/*------------------------------------------------------------------*/
T1 = "T1: VOLUME Largest_Free_Extent (Cyl) - Before Defrag"
T2 = "T2: VOLUME Largest_Free_Extent (Cyl) - After  Defrag"
T3 = "T3: Free Extents                     - Before Defrag"
T4 = "T4: Free Extents                     - After  Defrag"
"Prof Nopref"
/*------------------------------------------------------------------*/
/* Format the Defrag Sysprint output by calling the macro DefragM.    */
/*------------------------------------------------------------------*/
Ddname="Deniz"
Address Ispexec
"Lminit Dataid(Did) Ddname("Ddname") Enq(Exclu)"
"Edit Dataid(&Did) Macro(DefragM)"
"Lmfree Dataid(&Did)"
/*------------------------------------------------------------------*/
/* Write the Defrag statistics dataset to spool (SYSTSPRT).          */
/* This will include volume statistics before and after the Defrag.   */
/*------------------------------------------------------------------*/
Call Process1
/*------------------------------------------------------------------*/
/* Update "Defrag Summary Report" dset for the last defragmented disk.*/
/*------------------------------------------------------------------*/
Call Process2
/*------------------------------------------------------------------*/
/* If this is the last disk to be defragmented, the Flag will be 1.   */
/* So we have to wrap up the "Defrag Summary Report" dataset and      */
/* send it via e-mail.                                                */
/* Secondly, we are saving the "Defrag Summary Report" dataset in     */
/* the GDG dataset by submitting JCL EXPSMSØ3 which is built in the   */
/* SendJob procedure.                                                 */
/*------------------------------------------------------------------*/
If Flg="1" Then
  Do
     Call Process3 Sumdset martapv atalayg denizg
     Call SendJob
  End
Exit Ø  /* End-of-main-REXX */
/*==================================================================*/
/*==================================================================*/
PROCESS1:
/*------------------------------------------------------------------*/
/* Read the formatted Sysprint output.                                */
/*------------------------------------------------------------------*/
Address Tso
"Execio * Diskr "Deniz" (Stem Haydar. Finis"
"Free Fi(Deniz)"
```

```
Numb = Haydar.Ø               /* Total line number                    */
/*--------------------------------------------------------------------*/
/* Extract only necessary parts of Defrag Sysprint output.            */
/*--------------------------------------------------------------------*/
Do i = 1 To Numb
   Haydar.i = Substr(Haydar.i,28)
End
/*--------------------------------------------------------------------*/
/* If Defrag ends normal, then after being edited by edit macro,      */
/* Defrag output has to have 14 lines.                                */
/* 1st line will have volser info.                                    */
/*--------------------------------------------------------------------*/
If Numb <> 13 Then
  Do
     Say "Defrag sysprint is not as expected.            "
     Say "Please check out the following Sysprint records."
     Say "================================================"
     Do i = 1 To Numb
       Say Haydar.i
     End
     Return
  End
/*====================================================================*/
/* Write Defrag Statistics.                                           */
/* To be able to observe Defrag effect on the disk volume, disk's     */
/* space-related characteristics (before and after the Defrag ) will  */
/* be written to Systsprt dataset.                                   */
/*====================================================================*/
Say ""
Say ""
Say "                 DEFRAG STATISTICS FOR VOLUME -" Vol "-"
Say ""
Say "                                    BEFORE      AFTER "
Say "                                    ------      ------"
Say Substr(Haydar.1,1,3Ø) Substr(Haydar.1,31,11) Substr(Haydar.9,31,11)
Say Substr(Haydar.2,1,3Ø) Substr(Haydar.2,31,11) Substr(Haydar.1Ø,31,11)
Say Substr(Haydar.3,1,3Ø) Substr(Haydar.3,31,11) Substr(Haydar.11,31,11)
Say Substr(Haydar.4,1,3Ø) Substr(Haydar.4,31,11) Substr(Haydar.12,31,11)
Say Substr(Haydar.5,1,3Ø) Substr(Haydar.5,31,11) Substr(Haydar.13,31,11)
Say ""
Say Substr(Haydar.6,1,3Ø) Substr(Haydar.6,31,11)
Say Substr(Haydar.7,1,3Ø) Substr(Haydar.7,31,11)
Say Substr(Haydar.8,1,3Ø) Substr(Haydar.8,31,11)
Say ""
Say ""
RETURN /* End-of-the-procedure-PROCESS1     */
/*====================================================================*/
/*====================================================================*/
PROCESS2:
Address Tso
```

```
"Alloc Da("Sumdset") F(Mylog) Mod"
"Newstack"   /* Create a new stack */
Lo1=Substr(Haydar.4,31,6)       /* Contiguous space (before) cyl */
Lo1=Strip(Lo1,l,Ø)              /* Omit the leading zeros.       */
Lo2=Substr(Haydar.12,31,6)      /* Contiguous space (after)  cyl */
Lo2=Strip(Lo2,l,Ø)              /* Omit the leading zeros.       */
Lo3=Substr(Haydar.3,36,6)       /* Free extents (before)         */
Lo3=Strip(Lo3,l,Ø)              /* Omit the leading zeros.       */
Lo4=Substr(Haydar.11,36,6)      /* Free extents (after)          */
Lo4=Strip(Lo4,l,Ø)              /* Omit the leading zeros.       */
Log = Vol||"      "||Lo1||"      "||Lo2||"      "||Lo3||"      "||Lo4
Queue Log
If Flg="1" Then /* If it is the last disk for Defrag, then wrap-up    */
                /* "Defrag Summary" dset by adding commentary lines.  */
          Do
           Queue " "
           Queue "NOTE : To get more information for each defragmented
                       disk, check out the EXPSMSØ2 ouput in the spool."
           Queue " "
           Queue "LEGEND:"
           Queue "======="
           Queue T1
           Queue T2
           Queue T3
           Queue T4
           Queue " "
          End
Queue ""  /* Insert a null to indicate line mode is over */
"Execio * Diskw Mylog (Finis"
"Delstack"  /* Remove the stack */
"Free File(Mylog)"
RETURN /* End-of-the-procedure-PROCESS2     */
/*==================================================================*/
/*==================================================================*/
PROCESS3:
/*------------------------------------------------------------------*/
/* Send the "Defrag Summary Report" dataset to some users via Smtp.  */
/*------------------------------------------------------------------*/
Arg Dset Addr1 Addr2 Addr3
Address Tso
"Alloc Fi(Syslbc)   Da('Sys1.Brodcast')         Shr Reuse"
"Alloc Fi(Sysuads)  Da('Sys1.Uads')             Shr Reuse"
"Alloc Fi(Systcpd)  Da('Sis.Tcpip.Data(Tcpip)') Shr Reuse"
/*------------------------------------------------------------------*/
/* TCPIP member:                                                    */
/* This  member will have parameters related to SMTP such as;       */
/* Tcpipjobname, hostname, Domainorigin, Nsinteraddr, Datasetprefix.  */
/*------------------------------------------------------------------*/
Subj = Sysname"-Daily Defragmentation Summary " Date_Time
/*------------------------------------------------------------------*/
```

```
/* Build SMTPNOTE command.                                          */
/*-----------------------------------------------------------------*/
Q1 = " SMTPNOTE "
Q2 = "  TO("Addr1"@gasnaturalsdg.es "
Q3 = "     "Addr2"@gasnaturalsdg.es)"
Q4 = "  CC("Addr3"@gasnaturalsdg.es)"
Q5 = "  SUBJECT("Subj") "
Q6 = "  BATCH "
Q7 = "  DATASET('"Dset"')"
SMTPNOTE_COMMAND = Q1||Q2||Q3||Q4||Q5||Q6||Q7
/*-----------------------------------------------------------------*/
/* Issue the Smtpnote command.                                      */
/*-----------------------------------------------------------------*/
SMTPNOTE_COMMAND
Send_Rc = Rc
If Send_Rc <> Ø Then
                   Do
                     Say "Smtpnote command Return Code = " Send_Rc
                     Exit 99
                   End
"Free Fi(Syslbc)"
"Free Fi(Sysuads)"
"Free Fi(Systcpd)"
RETURN /* End-of-the-procedure-PROCESS3 */
/*=================================================================*/
/*=================================================================*/
SENDJOB:
/*-----------------------------------------------------------------*/
/* Generate a new version in the "GDG Defrag Summary" dataset.     */
/*-----------------------------------------------------------------*/
Address Tso
"Newstack"  /* Create a new stack */
Queue "//EXPSMSØ3 JOB MSGCLASS=Z,MSGLEVEL=(1,1),TIME=144Ø,"
Queue "//             NOTIFY=&SYSUID,CLASS=E"
Queue "//*"
Queue "//STEP2    EXEC PGM=ICEGENER"
Queue "//SYSUT1   DD   DISP=SHR,DSN=EXP.SMS.DEFRAG.SUMMARY"
Queue "//SYSUT2   DD   DISP=(MOD,CATLG),DCB=*.SYSUT1,"
Queue "// DSN=EXP.SMS.DEFRAG.SUMMARY.GDG(+1),"
Queue "// SPACE=(TRK,(1,1)),UNIT=SYSALLDA"
Queue "//SYSPRINT DD   SYSOUT=*"
Queue "//SYSIN    DD   DUMMY"
/*-----------------------------------------------------------------*/
/* Write the last job statements and submit the job.                */
/*-----------------------------------------------------------------*/
Queue "//"
Queue "QQ"
"Submit * End(QQ)"
"Delstack"  /* Remove the stack */
RETURN /* End-of-the-procedure-SENDJOB */
```

```
/*=================================================================*/
/*-------------------- End of member DefragR3 ---------------------*/
```

EXP.SMS.DEFRAG.SYSIN – SYSIN DATASET

```
 DEFRAG DDNAME(IN) MAXMOVE(999999,1) PASSDELAY(999) WAIT(2,2) -
```

EXP.SMS.DEFRAG.SYSIN.EXCLUDE – SYSIN DATASET

```
 EXCLUDE(LIST(                        -
             ENTER.HERE.DATA.SETS    -
             ENTER.HERE.DATA.SETS    -
            ))
 /*--------------------------------------------------------------*/
 /* Specify here fully or partially qualified name of datasets to be */
 /* excluded from the Defrag operation.                          */
 /*--------------------------------------------------------------*/
```

Atalay Gul
Systems Programmer
Azertia SA (Spain)

VSAM dataset administration

The VADM (VSAM ADMinistration) tool supports the following activities for KSDS VSAM datasets:

- Using LISTCAT to generate the source dataset definition into an ISPF file.
- Using the MODEL option to clone the dataset definition to another dataset.
- Using AMS to delete the dataset name.
- Using REPRO to unload and load data from one VSAM dataset to another.
- Using ALTER to change some dataset parameters.

- Using PRINT to browse the dataset in character, hex, or dump format.
- Using LISTCAT to display the basic dataset parameters and statistics.
- Using LISTCAT to show all dataset information.

The tool runs in an online or batch environment.

The example below shows how to find a VSAM dataset in ISPF3.4:

```
DSLIST - Datasets Matching CSPMSL.NAD*                      Row 1 of 12
Command ===>                                             Scroll ===> CSR

Command - Enter "/" to select action            Message          Volume
-------------------------------------------------------------------------
VADM     CSPMSL.NADI                                              *VSAM*
         CSPMSL.NADI.DATA                                         SMS231
         CSPMSL.NADI.INDEX                                        SMS231
         CSPMSL.NADI.T1                                           *VSAM*
         CSPMSL.NADI.T1.DATA                                      SMS231
         CSPMSL.NADI.T1.INDEX                                     SMS231
         CSPMSL.NADI1                                             *VSAM*
         CSPMSL.NADI1.DATA                                        SMS231
         CSPMSL.NADI1.INDEX                                       SMS231
         CSPMSL.NADI3                                             *VSAM*
         CSPMSL.NADI3.DATA                                        SMS231
         CSPMSL.NADI3.INDEX                                       SMS231
```

The VADM procedure shows the following main menu:

```
User Id: SYSADM           VSAM Administration            Ø2/Ø8/Ø9 Ø9:14

Command ===>
_______________________________________________________________

Source VSAM Dataset: CSPMSL.NADI
S
_  Cloning Source VSAM definition to ISPF file
S  Cloning Source VSAM Cluster to Another VSAM Cluster
_  Delete Source VSAM Cluster
_  Repro Source VSAM Cluster to Another VSAM Cluster
_  Alter Source VSAM Cluster
_  Browse Source VSAM Cluster
_  Display Source VSAM Cluster Parameters
_  Listcat Information
```

```
........................................................................

Enter S on choice and press <Enter>
PF3 - End                                                  Jun 2ØØ2,"ZB"
```

The following example shows the parameter entry panel and a JCL skeleton for cloning the source VSAM cluster to another cluster.

```
------------------  Parameter Entry Panel              ----------------
Command ===>

PARAMETER   PARAMETER VALUE                           PROMPT

VSAMI    => CSPMSL.NADI                               Input  VSAM dataset
VSAMO    => CSPMSL.NADI.NEW                           Output VSAM dataset
Volume   => SMS231                                    Volume
Mode     => B                                         B Batch O Online

Enter values |
                                PF3 Return
```

The VADM procedure generates the following Job control:

```
//SYSADMX JOB (ACCT#),'',
//             NOTIFY=SYSADM,REGION=4M,
//             CLASS=A,MSGCLASS=X,MSGLEVEL=(1,1)
//* ******************************************************************
//*
//* Cloning Source VSAM Cluster to Another VSAM Cluster
//* GENERATION DATE AND TIME : 9 Aug 2ØØ2 AT: 9:25am
//*
//*-------------------------------------------------------
//*---- DELETE/DEFINE VSAM CLUSTER -------------------
//DEFINE  EXEC PGM=IDCAMS
//SYSPRINT  DD SYSOUT=*
//SYSIN     DD *
   DELETE CSPMSL.NADI.NEW PURGE CLUSTER
   SET MAXCC = Ø
   DEFINE CLUSTER (NAME(CSPMSL.NADI.NEW)                 -
                   MODEL(CSPMSL.NADI)                -
                   VOL(SMS231))
/*
```

Components of VADM are as follows:

- VADM – the driver procedure
- VADMP1 – main menu – panel

- VADMP2 – parameter entry panel
- VADMP3 – selection result panel
- VADMP4 – online-batch panel
- VADMP5 – ALTER – change parameters panel
- VADMP6 – display statistics panel
- VADMP7 – display LISTCAT panel
- VADMS – JCL skeleton.

VADM

```
/* REXX */
/* trace r */
/* Input  VSAM dataset */
parse upper arg dsn
if dsn='' then do
   say 'Procedure use only in member list environment'
   Exit
end
infile=dsn
ids = '('||dsn||')'
vfile = substr(infile,2,length(infile)-2)
Top:
f1='';f2='';f3='';f3='';f4='';f4='';f5='';f6='';f7='';f8=''
fi=Ø
address ispexec "display panel(vadmp1)"
if rc=8 then Exit
title='Parameter Entry Panel'
if f5='S' | f5='s' then title='Alter Source VSAM Cluster'
if f7='S' | f7='s' then title='Display VSAM Atributes and Statistics'
VSAMI=vfile
/* Cloning Source VSAM Cluster definition to ISPF file  */
if f1='S' | f1='s' then do
   address ispexec "display panel(vadmp4)"
   if rc=8 then Signal Top
   if ans='YES' then do
      x=outtrap('var.')
      Address TSO "listcat,
                   entries('"VSAMi"'),
                   ALL"
      x=outtrap('off')
      Call VSAM_def
   end
   else do
```

```
      titles='Cloning Source VSAM Cluster definition to ISPF file'
      fi=1
      x=outtrap('var.')
      Address TSO "listcat,
                   entries('"VSAMi"'),
                   ALL"
      x=outtrap('off')
      Call VSAM_def
      Call Batch
   end
   Signal Top
end
/* Cloning Source VSAM Cluster to Another VSAM Cluster  */
if f2='S' | f2='s' then do
   titles='Cloning Source VSAM Cluster to Another VSAM Cluster'
   fi=2
   address ispexec "display panel(vadmp2)"
   if rc=8 then Signal Top
   if mode='O' then do
      x=outtrap('var.')
      Address TSO "define cluster
                   ( name('"VSAMO"'),
                     model('"VSAMI"'),
                     volumes("vol") )"
      x=outtrap('off')
      Call Report
   end
   else Call Batch
   Signal Top
end
/* Delete Source VSAM Cluster                          */
if f3='S' | f3='s' then do
   titles='Delete Source VSAM Cluster'
   fi=3
   address ispexec "display panel(vadmp2)"
   if rc=8 then Signal Top
   if mode='O' then do
      x=outtrap('var.')
      Address TSO "delete '"VSAMI"',
                            purge"
      x=outtrap('off')
      Call Report
   end
   else Call Batch
   Signal Top
end
/* Repro Source VSAM Cluster to Another VSAM Cluster    */
if f4='S' | f4='s' then do
   titles='Repro Source VSAM Cluster to Another VSAM Cluster'
   fi=4
   address ispexec "display panel(vadmp2)"
   if rc=8 then Signal Top
   if mode='O' then do
      x=outtrap('var.')
      Address TSO "repro ids('"VSAMI"'),
                         ods('"VSAMO"'),
                         REPLACE"
      x=outtrap('off')
      Call Report
   end
   else Call Batch
   Signal Top
end
/* Alter Source VSAM Cluster                            */
if f5='S' | f5='s' then do
   titles='Alter Source VSAM Cluster'
   fi=5
   x=outtrap('var.')
   Address TSO "listcat,
                entries('"VSAMi"'),
                ALL"
   x=outtrap('off')
   if word(var.1,1)='CLUSTER' | word(var.1,1)='INDEX' then do
      zedsmsg = 'Use DATA entry'
      zedlmsg = 'For Alter commamd use DATA entry not '||word(var.1,1)
      address ispexec "setmsg msg(isrzØØ1)"
      Signal Top
   end
   Call Init
   Up:
   address ispexec "display panel(vadmp5)"
   if rc=8 then Signal Top
   if zcmd='C' | zcmd='c' then do
      Call Copy_value
      zcmd=''
      Signal Up
   end
   Call Alter_Check
   if mode='O' & modi=1 then do
      x=outtrap('var.')
      Address TSO "alter '"VSAMI"' "aitem""
      x=outtrap('off')
      Call Report
   end
   if mode='B' & modi=1 then Call Batch
   modi=Ø
   Signal Top
end
/* Browse Source VSAM Cluster                           */
if f6='S' | f6='s' then do
```

```
   titles='Browse Source VSAM Cluster'
   fi=6
   optionp=''; opt=Ø
   address ispexec "display panel(vadmp7)"
   if rc=8 then Signal Top
   if formtype¬='' then optionp=formtype||' '
   if skip>Ø then optionp=optionp||'skip('||skip||') '
   if count>Ø then optionp=optionp||'count('||count||')'
   optionp=translate(optionp)
   if optionp¬='' then opt=1
   if mode='O' then do
      x=outtrap('var.')
      Address TSO "print,
                   indataset('"VSAMi"'),
                   "optionp""
      x=outtrap('off')
   Call Report
   end
   Else Call Batch
   Signal Top
end
/* Display Source VSAM Cluster Parameters               */
if f7='S' | f7='s' then do
   titles='Display Source VSAM Cluster Parameters'
   fi=7
   x=outtrap('var.')
   Address TSO "listcat,
                entries('"VSAMi"'),
                ALL"
   x=outtrap('off')
   if word(var.1,1)='DATA' | word(var.1,1)='INDEX' then do
      zedsmsg='Use VSAM entry'
      zedlmsg='For Display commamd use VSAM entry not '||word(var.1,1)
      address ispexec "setmsg msg(isrzØØ1)"
      Signal Top
   end
   Call Load
   address ispexec "display panel(vadmp6)"
   Signal Top
end
/* Listcat Information                                  */
if f8='S' | f8='s' then do
   x=outtrap('var.')
   Address TSO "listcat,
                entries('"VSAMi"'),
                ALL"
   x=outtrap('off')
   Call Report
   Signal Top
end
```

```
Exit
Batch:
  date=date()
  time=time(c)
  x=msg("off")
  user=userid()
  tempfile=userid()||'.VADM.TEMP'
  address tso
  "delete '"tempfile"'"
  "free dsname('"tempfile"')"
  "free ddname(ispfile)"
  "free attrlist(formfile)"
  "attrib formfile blksize(8ØØ) lrecl(8Ø) recfm(f b) dsorg(ps)"
  "alloc ddname(ispfile) dsname('"tempfile"')",
         "new using (formfile) unit(339Ø) space(1 1) cylinders"
  address ispexec
  "ftopen"
  "ftincl VADMS"
  "ftclose"
  zedsmsg = "JCL shown"
  zedlmsg = "JCL for VADM shown"
  "setmsg msg(isrzØØ1)"
  "edit dataset('"tempfile"')"
Return
Report:
  address ispexec 'tbcreate "tlist" names(detail)'
  do i=1 to var.Ø
     detail=var.i
     address ispexec 'tbadd "tlist"'
     if length(var.i)>78
     then do
        detail=substr(var.i,79)
        address ispexec 'tbadd "tlist"'
     end
  end
  address ispexec 'tbtop "tlist"'
  address ispexec 'tbdispl "tlist" panel(vadmp3)'
  address ispexec 'tbend "tlist"'
Return
VSAM_def:
  address ispexec 'tbcreate "tlist" names(detail)'
  detail='DEFINE CLUSTER -'
  address ispexec 'tbadd "tlist"'
  detail='         (NAME('||word(var.1,3)||')) -'
  address ispexec 'tbadd "tlist"'
  detail='    DATA -'
  address ispexec 'tbadd "tlist"'
  detail='         (NAME('||word(var.21,3)||') -'
  address ispexec 'tbadd "tlist"'
  detail='         '||,
        word(translate(word(var.42,1),' ','-'),3)||'S('||,
        word(translate(word(var.43,1),' ','-'),3)||' '||,
        word(translate(word(var.44,1),' ','-'),3)||') -'
address ispexec 'tbadd "tlist"'
detail='          KEYS('||,
        word(translate(word(var.31,1),' ','-'),2)||' '||,
        word(translate(word(var.32,1),' ','-'),2)||') -'
address ispexec 'tbadd "tlist"'
detail='          BUFFERSPACE('||,
        word(translate(word(var.31,3),' ','-'),2)||') -'
address ispexec 'tbadd "tlist"'
detail='          FREESPACE('||,
        word(translate(word(var.38,2),' ','-'),3)||' '||,
        word(translate(word(var.39,2),' ','-'),3)||') -'
address ispexec 'tbadd "tlist"'
detail='          RECORDSIZE('||,
        word(translate(word(var.31,2),' ','-'),2)||' '||,
        word(translate(word(var.32,2),' ','-'),2)||') -'
address ispexec 'tbadd "tlist"'
detail='          CONTROLINTERVALSIZE('||,
        word(translate(word(var.31,4),' ','-'),2)||') -'
address ispexec 'tbadd "tlist"'
detail='          SHAREOPTIONS('||,
        substr(var.33,17,1)||' '||substr(var.33,19,1)||') -'
address ispexec 'tbadd "tlist"'
detail='          '||word(var.33,2)||' '||word(var.33,3)||' -'
address ispexec 'tbadd "tlist"'
detail='          '||word(var.33,4)||' '||word(var.33,5)||' -'
address ispexec 'tbadd "tlist"'
detail='          '||word(var.33,6)||' '||word(var.33,7)||' -'
address ispexec 'tbadd "tlist"'
detail='          '||word(var.34,1)||' '||word(var.34,2)||' -'
address ispexec 'tbadd "tlist"'
detail='          '||word(var.34,3)||' -'
address ispexec 'tbadd "tlist"'
detail='          VOLUMES('||,
        word(translate(word(var.46,1),' ','-'),2)||') )-'
address ispexec 'tbadd "tlist"'
detail='    INDEX -'
address ispexec 'tbadd "tlist"'
detail='          (NAME('||word(var.52,3)||') -'
address ispexec 'tbadd "tlist"'
detail='          '||,
        word(translate(word(var.72,1),' ','-'),3)||'S('||,
        word(translate(word(var.73,1),' ','-'),3)||' '||,
        word(translate(word(var.74,1),' ','-'),3)||') -'
address ispexec 'tbadd "tlist"'
detail='          VOLUMES('||,
        word(translate(word(var.76,1),' ','-'),2)||') )-'
address ispexec 'tbadd "tlist"'
```

```
  detail='      CATALOG('||word(var.2,3)||')'
  address ispexec 'tbadd "tlist"'
  if ans='YES' then do
     address ispexec 'tbtop "tlist"'
     address ispexec 'tbdispl "tlist" panel(vadmp3)'
     address ispexec 'tbend "tlist"'
  end
Return
Init:
  avol=word(translate(word(var.26,1),' ','-'),2)
  bufs=word(translate(word(var.11,3),' ','-'),2)
  eras=word(var.13,4)
  ci=word(translate(word(var.18,2),' ','-'),3)
  ca=word(translate(word(var.19,2),' ','-'),3)
  le=word(translate(word(var.11,1),' ','-'),2)
  of=word(translate(word(var.12,1),' ','-'),2)
  recs=word(translate(word(var.11,2),' ','-'),2)||' '||,
       word(translate(word(var.12,2),' ','-'),2)
  rvol=word(translate(word(var.26,1),' ','-'),2)
  shrl=substr(var.13,17,1)||' '||substr(var.13,19,1)
  unik=word(var.13,3)
Return
Load:
  cname=word(var.1,3)
  catname=word(var.2,3)
  creat=word(translate(word(var.4,2),' ','-'),2)
  rac=word(translate(word(var.17,2),' ','-'),2)
  rac=translate(rac,' ','(')
  rac=translate(rac,' ',')')
  keyl=word(translate(word(var.31,1),' ','-'),2)
  avgl=word(translate(word(var.31,2),' ','-'),2)
  maxl=word(translate(word(var.32,2),' ','-'),2)
  bufs=word(translate(word(var.31,3),' ','-'),2)
  cisz=word(translate(word(var.31,4),' ','-'),2)
  cica=word(translate(word(var.32,4),' ','-'),2)
  space=word(translate(word(var.42,1),' ','-'),3)
  pri=word(translate(word(var.43,1),' ','-'),3)
  sec=word(translate(word(var.44,1),' ','-'),3)
  hia=word(translate(word(var.42,2),' ','-'),4)
  hiu=word(translate(word(var.43,2),' ','-'),4)
  keyli=word(translate(word(var.61,1),' ','-'),2)
  avgli=word(translate(word(var.61,2),' ','-'),2)
  maxli=word(translate(word(var.62,2),' ','-'),2)
  bufsi=word(translate(word(var.61,3),' ','-'),2)
  ciszi=word(translate(word(var.61,4),' ','-'),2)
  cicai=word(translate(word(var.62,4),' ','-'),2)
  spacei=word(translate(word(var.72,1),' ','-'),3)
  prii=word(translate(word(var.73,1),' ','-'),3)
  seci=word(translate(word(var.74,1),' ','-'),3)
  hiai=word(translate(word(var.72,2),' ','-'),4)
  hiui=word(translate(word(var.73,2),' ','-'),4)
  rect=word(translate(word(var.36,1),' ','-'),3)
  recd=word(translate(word(var.37,1),' ','-'),3)
  reci=word(translate(word(var.38,1),' ','-'),3)
  recu=word(translate(word(var.39,1),' ','-'),3)
  recr=word(translate(word(var.4Ø,1),' ','-'),3)
Return
Copy_value:
  avol1=avol
  bufs1=bufs
  eras1=eras
  ci1=ci
  ca1=ca
  le1=le
  of1=of
  recs1=recs
  rvol1=rvol
  shrl1=shrl
  unik1=unik
Return
Alter_Check:
  modi=Ø
  aitem=' '
  if avol1¬=' ' then do
     if avol1¬=avol then do
        aitem=aitem||'ADDVOLUMES('||avol1||') '
        modi=1
     end
  end
  if bufs1¬=' ' then do
     if bufs1¬=bufs then do
        aitem=aitem||'BUFFERSPACE('||bufs1||') '
        modi=1
     end
  end
  if eras1¬=' ' & eras1¬=eras then do
     if eras1¬=eras then do
        aitem=aitem||eras1||' '
        modi=1
     end
  end
  if ci1¬=' ' then do
     if ci1¬=ci then do
        aitem=aitem||'FREESPACE('||ci1||' '
        if ca1=' ' then ca1='Ø'
        modi=1
     end
  end
  if ca1¬=' ' then do
     if ca1¬=ca then do
```

```
        aitem=aitem||ca1||') '
        modi=1
     end
  end
  if le1¬=' ' then do
     if le1¬=le then do
        aitem=aitem||'KEYS('||le1||' '
        if of1=' ' then of1='Ø'
        modi=1
     end
  end
  if of1¬=' ' then do
        aitem=aitem||of1||') '
        modi=1
  end
  if recs1¬=' ' then do
     if recs1¬=recs then do
        aitem=aitem||'RECORDSIZE('||recs1||') '
        modi=1
     end
  end
  if rvol1¬=' ' then do
     if rvol1¬=rvol then do
        aitem=aitem||'REMOVEVOLUMES('||rvol1||') '
        modi=1
     end
  end
  if shrl1¬=' ' then do
     if shrl1¬=shrl then do
        aitem=aitem||'SHAREOPTIONS('||shrl1||') '
        modi=1
     end
  end
  if unik1¬=' ' then do
     if unik1¬=unik then do
        aitem=aitem||unik1||' '
        modi=1
     end
  end
Return
```

VADMP1

```
)attr default(%+_)
   [ type (output) intens(low)  color(green) caps(off)
   # type (output) intens(low)  color(white) caps(off)
   _ type (input)  intens(low)  color(yellow) caps(off) pad('_')
   + type (text)   intens(low)  color(green)
   ] type (text)   intens(low)  color(yellow)
   ~ type (text)   intens(high) color(turquoise)
   { type (text)   intens(high) color(blue)
   @ type (text)   intens(high) color(red)   caps(off) hilite(reverse)
)body window(78,23) expand ($$)
]
 User Id: &zuser        + @ VSAM Administration +            ~ &zdate
&ztime
+
%Command ===>_zcmd                                                      +
+
{Source VSAM Dataset:#vfile                                            +
+S+
_z+[field1                                                             +
_z+[field2                                                             +
_z+[field3                                                             +
_z+[field4                                                             +
_z+[field5                                                             +
_z+[field6                                                             +
_z+[field7                                                             +
_z+[field8                                                             +
+
]........................................................................
+
#msg                                                                    +
]PF3 - End +                                                  ~Jun
2002,"ZB"
)init
  .ZVARS = '(f1 f2 f3 f4 f5 f6 f7 f8)'
  &field1 = 'Cloning Source VSAM definition to ISPF file'
  &field2 = 'Cloning Source VSAM Cluster to Another VSAM Cluster'
  &field3 = 'Delete Source VSAM Cluster'
  &field4 = 'Repro Source VSAM Cluster to Another VSAM Cluster'
  &field5 = 'Alter Source VSAM Cluster'
  &field6 = 'Browse Source VSAM Cluster'
  &field7 = 'Display Source VSAM Cluster Parameters'
  &field8 = 'Listcat Information'
  &msg = 'Enter S on choice and press <Enter>'
  IF (&kurs = F1,FIELD1)
      .attr (field1) = 'color (yellow) caps(on)'
  IF (&kurs = F2,FIELD2)
      .attr (field2) = 'color (yellow) caps(on)'
  IF (&kurs = F3,FIELD3)
      .attr (field3) = 'color (yellow) caps(on)'
  IF (&kurs = F4,FIELD4)
      .attr (field4) = 'color (yellow) caps(on)'
  IF (&kurs = F5,FIELD5)
      .attr (field5) = 'color (yellow) caps(on)'
  IF (&kurs = F6,FIELD6)
      .attr (field6) = 'color (yellow) caps(on)'
  IF (&kurs = F7,FIELD7)
```

```
      .attr (field7) = 'color (yellow) caps(on)'
  IF (&kurs = F8,FIELD8)
      .attr (field8) = 'color (yellow) caps(on)'
)proc
  &kurs = .CURSOR
  if (.pfkey = pfØ3) &pf3 = exit
)end
```

VADMP2

```
)Attr Default(%+_)
   | type(text)   intens(high) caps(on ) color(yellow)
   $ type(output) intens(high) caps(off) color(yellow)
   ] type(output) intens(high) caps(off) color(green) hilite(reverse)
   { type(output) intens(high) caps(off) color(blue)
   ? type(text)   intens(high) caps(on ) color(green) hilite(reverse)
   # type(text)   intens(high) caps(off) hilite(reverse)
   @ type(output) intens(high) caps(off) color(yellow) hilite(reverse)
   [ type( input) intens(high) caps(on ) color(green) pad(_)
)Body  Expand(//)
|-/-/- @title                                  +|-/-/-

%Command ===>_zcmd
+
+
#PARAMETER  #PARAMETER VALUE                     #PROMPT            +
+
+VSAMI    =>$vsami                              +Input  VSAM dataset
+VSAMO    =>[vsamo                              +Output VSAM dataset
+Volume   =>[vol   +                            +Volume
+Mode     =>[z+                                 |B+Batch|O+Online
+
$msg                                                                +
                               # PF3 Return +
)Init
  .ZVARS = '(mode)'
  if (&vsamo ¬= ' ')
      .attr (vsamo) = 'pad(nulls)'
  if (&vol ¬= ' ')
      .attr (vol) = 'pad(nulls)'
  if (&mode ¬= ' ')
      .attr (mode) = 'pad(nulls)'
  &msg='Enter values |'
)Reinit
)Proc
  if (&fi=2 | &fi=4)
  VER(&VSAMO,NONBLANK)
  if (&fi=2)
```

```
     VER(&Vol,NONBLANK)
  if (&fi=2 | &fi=3 | &fi=4)
     VER(&Mode,NONBLANK)
  if (&fi=2 | &fi=3 | &fi=4)
     VER(&Mode LIST B,O)
  VPUT (VSAMO Vol Mode) PROFILE
)End
```

VADMP3

```
)Attr Default(%+_)
   ( type(text  ) intens(high) hilite(reverse)
   ] type(text  ) intens(high) hilite(reverse) color(white)
   / type(text  ) intens(high) hilite(reverse) color(yellow)
   [ type(text  ) intens(high) hilite(reverse) color(green)
   + type(text  ) intens(low )
   _ type( input) intens(high) caps(on ) just(left )
   ^ type(output) intens(low ) caps(off) just(asis )
   ~ type(output) intens(low ) caps(off) just(asis ) JUST(RIGHT)
)Body
/ Message Panel +
+
+Command ===>_zcmd                                    +Scroll ===>_amt +
+
+Press[Enter+to have this service continue.
+
]Message                                                              +
+
)Model
^z                                                                    +
)Init
  .ZVARS = '(detail)'
  &amt = PAGE
)Reinit
)Proc
)End
```

VADMP4

```
)ATTR DEFAULT(%+_)
   _ TYPE(INPUT)  COLOR(RED)       HILITE(USCORE)  INTENS(HIGH)
   } TYPE(OUTPUT) COLOR(WHITE) CAPS(OFF) HILITE(REVERSE) INTENS(HIGH)
   ] TYPE(TEXT)   COLOR(TURQUOISE)                 INTENS(HIGH)
   | TYPE(TEXT)   COLOR(WHITE)     HILITE(REVERSE) INTENS(HIGH)
   # TYPE(TEXT)   COLOR(YELLOW)                    INTENS(HIGH)
   ) TYPE(TEXT)   COLOR(GREEN)                     INTENS(HIGH)
   @ TYPE(TEXT)   COLOR(RED)                       INTENS(HIGH)
)BODY WINDOW(35,8)
```

```
}FUN
| +                              |
| #     Online Execution         |
| +                              |
| )        @==>+_ANS+            |
| +                              |
| ]Enter:Continue         PF3:End|
|
)INIT
  &FUN  = '__________ Confirm Panel _________'
)PROC
  VER(&ans,NONBLANK)
  VER(&and LIST YES,NO)
  IF (.PFKEY = PFØ3) &PF3 = EXIT
)END
```

VADMP5

```
)Attr Default(\+_)
   | type(text)   intens(high) caps(on ) color(yellow)
   $ type(output) intens(high) caps(off) color(white)
   ] type(output) intens(high) caps(off) color(green) hilite(reverse)
   { type(output) intens(high) caps(off) color(blue)
   ? type(text)   intens(high) caps(on ) color(green) hilite(reverse)
   # type(text)   intens(high) caps(off) hilite(reverse)
   @ type(output) intens(high) caps(off) color(yellow) hilite(reverse)
   [ type( input) intens(high) caps(on ) color(green) pad(_)
)Body  Expand(//)
|-/-/- @title                                  +|-/-/-
\Command ===>_zcmd                                                 +
+
+Type|C+in command line to copy old value to new value.
+
#PARAMETER     #OLD VALUE          #NEW VALUE       #PROMPT           +
+
+VSAMI       =>$vsami                               +Input  VSAM dataset
+Addvolumes  =>$avol  +            [avol1 +         +Add new volume
+Bufferspace =>$bufs   +           [bufs1  +        +Bufferspace
+Erase       =>$eras   +           [eras1  +        |E+Erase|N+Noerase
+Freespace   =>$ci + $ca +         [ci1+ [ca1+      +%CI %CA
+Keys        =>$le + $of +         [le1+ [of1+      +Keylen rkp
+Recordsize  =>$recs     +         [recs1    +      +Record size
+Remvolumes  =>$rvol  +            [rvol1 +         +Remove volume
+Shareoption =>$shrl  +            [shrl1 +         +Share level options
+Uniquekey   =>$unik       +       [unik1      +
|U+Uniquekey|N+Nonuniquekey
+Mode        =>[z+                                  |B+Batch|O+Online
+
$msg                                                               +
                              # PF3 Return +
)Init
  .ZVARS = '(mode)'
  if (&avol1 ¬= ' ')
      .attr (avol1) = 'pad(nulls)'
  if (&bufs1 ¬= ' ')
      .attr (bufs1) = 'pad(nulls)'
  if (&eras1 ¬= ' ')
      .attr (eras1) = 'pad(nulls)'
  if (&ci1 ¬= ' ')
      .attr (ci1) = 'pad(nulls)'
  if (&ca1 ¬= ' ')
      .attr (ca1) = 'pad(nulls)'
  if (&le1 ¬= ' ')
      .attr (le1) = 'pad(nulls)'
  if (&of1 ¬= ' ')
      .attr (of1) = 'pad(nulls)'
  if (&recs1 ¬= ' ')
      .attr (recs1) = 'pad(nulls)'
  if (&rvol1 ¬= ' ')
      .attr (rvol1) = 'pad(nulls)'
  if (&shrl1 ¬= ' ')
      .attr (shrl1) = 'pad(nulls)'
  if (&unik1 ¬= ' ')
      .attr (unik1) = 'pad(nulls)'
  if (&mode ¬= ' ')
      .attr (mode) = 'pad(nulls)'
  &msg='Enter new values |'
)Reinit
)Proc
  VER(&bufs1,NUM)
  &eras1 = TRANS(TRUNC(&eras1,1) E,ERASE N,NOERASE)
  VER(&eras1 LIST ERASE,NOERASE)
  &unik1 = TRANS(TRUNC(&unik1,1) U,UNIQUEKEY N,NONUNIQUEKEY)
  VER(&unik1 LIST UNIQUEKEY,NONUNIQUEKEY)
  VER(&ca1,NUM)
  VER(&ci1,NUM)
  VER(&le1,NUM)
  VER(&of1,NUM)
  VER(&Mode LIST B,O)
  if (&avol1=&avol)
     &zedsmsg='The same name'
     &zedlmsg='The same name not allowed'
     .msg = isrzØØ1
  if (&bufs1=&bufs)
     &zedsmsg='The same name'
     &zedlmsg='The same name not allowed'
     .msg = isrzØØ1
  if (&eras1=&eras)
     &zedsmsg='The same name'
     &zedlmsg='The same name not allowed'
     .msg = isrzØØ1
  if (&recs1=&recs)
     &zedsmsg='The same name'
     &zedlmsg='The same name not allowed'
     .msg = isrzØØ1
  if (&rvol1=&rvol)
     &zedsmsg='The same name'
     &zedlmsg='The same name not allowed'
     .msg = isrzØØ1
  if (&shrl1=&shrl)
     &zedsmsg='The same name'
     &zedlmsg='The same name not allowed'
     .msg = isrzØØ1
  if (&unik1=&unik)
     &zedsmsg='The same name'
     &zedlmsg='The same name not allowed'
     .msg = isrzØØ1
  VPUT (VSAMO Vol Mode) PROFILE
)End
```

VADMP6

```
)Attr Default(\+_)
   | type(text)   intens(high) caps(on ) color(yellow)
   $ type(output) intens(high) caps(off) color(white)
   ] type(output) intens(high) caps(off) color(green) hilite(reverse)
   { type(output) intens(high) caps(off) color(blue)
   } type(output) intens(high) caps(off) color(red)
   ? type(text)   intens(high) caps(on ) color(green) hilite(reverse)
   # type(text)   intens(high) caps(off) hilite(reverse)
   @ type(output) intens(high) caps(off) color(yellow) hilite(reverse)
   [ type( input) intens(high) caps(on ) color(green) pad(_)
)Body  Expand(//)
|-/-/- @title                                  +|-/-/-
\Command ===>_zcmd                                                 +
+
#CLUSTER  +                                                        +
+Name    : $cname                                   +
+In-Cat  : $catname                         +
+Creation: $creat     +
+Racf    : $rac+
#DATA     +              #INDEX    +              #STATISTICS    +
+Keylen  : $keyl  +   +Keylen  : $keyli +   +Rec-total    : $rect      +
+Avgrecl : $avgl  +   +Avgrecl : $avgli +   +Rec-deleted  : $recd      +
+Maxrecl : $maxl  +   +Maxrecl : $maxli +   +Rec-inserted : $reci      +
+Bufspace: $bufs  +   +Bufspace: $bufsi +   +Rec-updated  : $recu      +
+Cisize  : $cisz  +   +Cisize  : $ciszi +   +Rec-retrieved: $recr      +
+CI CA   : $cica  +       +CI CA   : $cicai +
+Space   : $space    +    +Space   : $spacei   +
+Pri     : $pri   +       +Pri     : $prii  +
+Sec     : $sec   +       +Sec     : $seci  +
+Hi-a-rba: $hia       +  +Hi-a-rba: $hiai       +
+Hi-u-rba: $hiu       +  +Hi-u-rba: $hiui       +
+
}msg                      ? PF3 Return +
)Init
  &msg='Enter any key |'
)Reinit
)Proc
)End
```

VADMP7

```
)Attr Default(%+_)
   | type(text)   intens(high) caps(on ) color(yellow)
   $ type(output) intens(high) caps(off) color(yellow)
   ] type(output) intens(high) caps(off) color(green) hilite(reverse)
   { type(output) intens(high) caps(off) color(blue)
   ? type(text)   intens(high) caps(on ) color(green) hilite(reverse)
   # type(text)   intens(high) caps(off) hilite(reverse)
   @ type(output) intens(high) caps(off) color(yellow) hilite(reverse)
   [ type( input) intens(high) caps(on ) color(green) pad(_)
)Body  Expand(//)
|-/-/- @title                                  +|-/-/-
%Command ===>_zcmd                                                  +
+
#PARAMETER  #PARAMETER VALUE                          #PROMPT         +
+
+Format   =>[formtype +                             +Character, Dump, Hex
+Skip     =>[skip     +                             +Skip number
+Count    =>[count    +                             +Count number
+Mode     =>[z+                                     |B+Batch|O+Online
+
$msg                                                                +
                               # PF3 Return +
)Init
  .ZVARS = '(mode)'
  if (&formtype ¬= ' ')
      .attr (formtype) = 'pad(nulls)'
  if (&skip ¬= ' ')
      .attr (skip) = 'pad(nulls)'
  if (&count ¬= ' ')
      .attr (count) = 'pad(nulls)'
  &msg='Enter values |'
)Reinit
)Proc
  &formtype= TRANS(TRUNC(&formtype,1) C,CHARACTER H,HEX D,DUMP)
  VER(&formtype LIST CHARACTER,HEX,DUMP)
  VER(&skip,NUM)
  VER(&count,NUM)
  VER(&mode LIST B,O)
  VPUT (formtype skip count mode) PROFILE
)End
```

VADMS

```
)TBA 72
)CM ------------------------------------------------------------------
)CM Skeleton to generate VADM utility                               --
)CM ------------------------------------------------------------------
//&user.X JOB (ACCT#),'&option',
//             NOTIFY=&user,REGION=4M,
//             CLASS=A,MSGCLASS=X,MSGLEVEL=(1,1)
//* ******************************************************************
//*
//* &titles
//* GENERATION DATE AND TIME : &date AT: &time
//*
//*-----------------------------------------------------
)SEL &fi = 1
//*---- DEFINE VSAM CLUSTER ---------------------------
//DEFINE  EXEC PGM=IDCAMS
//SYSPRINT  DD SYSOUT=*
//SYSIN     DD *
)DOT "TLIST"
    &detail
)ENDDOT
)ENDSEL
)SEL &fi = 2
//*---- DELETE/DEFINE VSAM CLUSTER --------------------
//DEFINE  EXEC PGM=IDCAMS
//SYSPRINT  DD SYSOUT=*
//SYSIN     DD *
   DELETE &vsamo PURGE CLUSTER
   SET MAXCC = Ø
   DEFINE CLUSTER (NAME(&vsamo)                -
                   MODEL(&vsami)               -
                   VOL(&vol))
)ENDSEL
)SEL &fi = 3
//*---- DELETE VSAM CLUSTER ---------------------------
//DEFINE  EXEC PGM=IDCAMS
//SYSPRINT  DD SYSOUT=*
//SYSIN     DD *
   DELETE &vsami PURGE CLUSTER
)ENDSEL
```

```
)SEL &fi = 4
//*---- REPRO FROM VSAM TO VSAM ----------------------
//DEFINE  EXEC PGM=IDCAMS
//SYSPRINT  DD SYSOUT=*
//SYSIN     DD *
   REPRO INDATASET(&vsami) -
         OUTDATASET(&vsamO) -
         REPLACE
)ENDSEL
)SEL &fi = 5
//*---- ALTER VSAM DATASET ---------------------------
//DEFINE  EXEC PGM=IDCAMS
//SYSPRINT  DD SYSOUT=*
//SYSIN     DD *
   ALTER &vsami -
  &aitem
)ENDSEL
)SEL &fi = 6
//*---- PRINT VSAM DATASET ---------------------------
//DEFINE  EXEC PGM=IDCAMS
//SYSPRINT  DD SYSOUT=*
//SYSIN     DD *
)SEL &opt = Ø
   PRINT INDATASET(&vsami)
)ENDSEL
)SEL &opt = 1
   PRINT INDATASET(&vsami) -
         &optionp
)ENDSEL
)ENDSEL
/*
```

Bernard Zver
DBA
Informatika (Slovenia)

Matching a filename against a pattern – revisited

The program PATTERN, published in *Matching a filename against a pattern* (*MVS Update*, Issue 191, August 2002), contains a bug. Under certain circumstances, valid matches will fail. Below is a new version of the program.

```
/* REXX MVS *=======================================================*/
/* PATTERN - This program matches a filename to a generic pattern.  */
/* It should be used as a subroutine. The entry parameter is a      */
/* string containg two words, the filename to check and the pattern,*/
/* separated by one or more spaces. Their order is irrelevant.      */
/* Filename is a fully-qualified dataset name                       */
/* Pattern follows the ISPF style convention:                       */
/*    *   - any number of characters                                */
/*    %   - one character                                           */
/*   .*.  - one qualifier                                           */
/*   .**. - any number of qualifiers (including none)               */
/* RC: Ø if there is a match, -1 otherwise.                         */
/* If this program is called as a subroutine, then RC is returned,  */
/* otherwise RC is "said".                                          */
/*==================================================================*/
arg arg1 arg2 .
if pos("*",arg1) > Ø | pos("%",arg1) > Ø then do
   file = arg2
   pattern = arg1
end
else do
   file = arg1
   pattern = arg2
end
any = Ø
returncode = Ø
do alpha = Ø
   parse var pattern p1 "." pattern
   if p1 = "" then p1 = ";"
   if p1 = "**" then do
      if pattern = "" then do
         leave alpha
      end
      else do
         any = 1
         iterate alpha
      end
   end
   if right(p1,1) = "*" & pattern = "" & any = Ø then do
      any = 2
   end
   do beta = Ø
      parse var file f1 "." file
      if f1 = "" then f1 = ";"
      if f1 = ";" & p1 = ";" then leave alpha
      call check_qualifier p1 f1
      returncode = result
      if any = Ø then do
         if returncode = Ø then do
            iterate alpha
         end
         else do
            leave alpha
         end
      end
      if any = 1 then do
         if returncode = 0 then do
            if right(p1,1) = "*" & pattern = "" then do
               leave alpha
            end
            else do
               any = 0
            end
            iterate alpha
         end
         else do
            if f1 = ";" then do
               leave alpha
            end
            else do
               iterate beta
            end
         end
      end
      if any = 2 then do
         leave alpha
      end
   end
end
parse source . calltype .
if calltype = "COMMAND" then say returncode
else return returncode
exit
/*===============================================================*/
check_qualifier: procedure
arg str1 str2
if datatype(left(str2,1),"W") then return -1
if str2 = ";" then return -1
if str1 = ";" then return -1
if str1 = "*" then return  0
call check_qualifier1 str1 str2
if result = -1 then do
   str1a = ""
   do f = length(str1) to 1 by -1
      str1a = str1a||substr(str1,f,1)
   end
   str2a = ""
   do f = length(str2) to 1 by -1
      str2a = str2a||substr(str2,f,1)
   end
   call check_qualifier1 str1a str2a
end
return result
/*=================================================================*/
check_qualifier1: procedure
arg str1 str2
do forever
   p = pos("*%",str1)
   if p = Ø then leave
   str1 = overlay("%*",str1,p)
end
do forever
   p = pos("%*%",str1)
   if p = Ø then leave
   str1 = overlay("%%*",str1,p)
end
do forever
   p = pos("**",str1)
   if p = Ø then leave
   str1 = delstr(str1,p,1)
end
do length(str2)
   if right(str2,1) = right(str1,1) then do
      str2 = substr(str2,1,length(str2)-1)
      str1 = substr(str1,1,length(str1)-1)
   end
end
if str2 = str1 then return Ø
if str2 = "" & str1 = "*" then return Ø
if str2 = "" & str1 <> "" then return -1
v = Ø
prv = ""
str3 = ""
no_pos = Ø
do k = 1 to length(str1)
  select
    when substr(str1,k,1) = '%' then do
       do k1 = k + 1 to length(str1)
          if substr(str1,k1,1) <> '%' then leave k1
       end
       v = v + 1
       if no_pos = 1 then do
          str3 = str3"?var."v
       end
       else do
          str3 = str3"?="k"?var."v
       end
       if substr(str1,k1,1) <> '*' then do
          if no_pos = 1 then do
             str3 = str3"?"
          end
          else do
             str3 = str3"?="k1"?"
          end
          len_eq.v = k1 - k
          len_ge.v = Ø
          k = k1 - 1
       end
       else do
          no_pos = 1
          len_ge.v = k1 - k
          len_eq.v = Ø
          k = k1
       end
    end
    when substr(str1,k,1) = '*' then do
       no_pos = 1
       v = v + 1
       str3 = str3"?var."v"?"
       len_eq.v = Ø
       len_ge.v = Ø
    end
    otherwise do
       str3 = str3"?'"
       do k1 = k to length(str1)
          if substr(str1,k1,1) = '%' |,
             substr(str1,k1,1) = '*' then leave k1
          str3 = str3||substr(str1,k1,1)
       end
       str3 = str3"'"
       k = k1 - 1
    end
  end
end
str3 = space(str3,Ø)
str3 = translate(str3," ","?")
interpret "parse var" str2 str3
ww = words(str3)
do w = 1 to ww
   if left(word(str3,w),1)="=" then,
   str3 = delword(str3,w,1)
end
str3 = space(str3,Ø)
interpret "string = value("str3")"
if string <> str2 then do
   answer = -1
end
else do
   answer = Ø
   do v1 = 1 to v
```

```
      if len_ge.v1 > Ø then do
         if length(var.v1) <  len_ge.v1 then answer = -1
      end
      if len_eq.v1 > Ø then do
         if length(var.v1) <> len_eq.v1 then answer = -1
      end
   end
end
return answer
```

Systems Programmer (Portugal) © Xephon 2003

MVS news

IBM has announced Output Manager for z/OS V1.1 (OM), which captures enterprise data from existing z/OS applications, distributes it on-line, makes outputs available to business decision makers, and tracks data distribution and usage for auditing and planning.

Transforming data into customized accessible formats, the software lets users specify report attributes, collect report data, split system output into individual reports, and perform both on-line and batch printing jobs for efficient printing and delivery to the end users of report data.

It runs on any supported release of z/OS or OS/390 Version 2.8 and later.

For further information contact your local IBM representative.
URL: http://www.ibm.com/software/ad.

* * *

IBM has announced Session Manager for z/OS V1.1, a session manager for VTAM and TCP/IP that provides access to multiple OS/390 or z/OS systems from a single 3270 terminal.

It provides a password-protected single menu from which users can access all applications running on any z/OS or OS/390 machine in the network. It also provides logoff procedures, security checking, and audit logging and centralized administration, operations, and monitoring.

Beyond basic session management, it provides facilities for viewing the screens of remote users, compressing data streams between terminals and applications, and building new applications by integrating existing applications.

For further information contact your local IBM representative.
URL: http://www.ibm.com.

* * *

IBM has launched its Application Workload Modeler R1 for z/OS, which provides the ability to model, measure, and analyse the performance of networks and applications in a client/server, multiprotocol, multiplatform environment.

It's designed to help sites plan for the roll-out of additional software or function more accurately and determine where upgrades may be required in the network and systems. It can be used in two modes. Client/Server mode benchmarks allow sites to evaluate the performance of a communications stack (such as TCP/IP and SNA) and network by modelling the behaviour of an application and generating the associated network traffic. In this case, the actual application need not be installed. It can model both the client and server.

Secondly, the Application Client mode benchmarks allow users to measure the overall end-to-end performance of a well-known TCP/IP application server. This is done by using Application Workload Modeler as a client to an existing application server, such as a TN3270 or FTP server. It will generate real traffic, as specified by the user, to the application server.

Services in support of Application Workload Modeler R1 are available from IBM Global Services and AIM zSeries Software Services.

For further information contact your local IBM representative.
URL: http://www.ibm.com/software/network/awm.