135 VM

November 1997

update

In this issue

VM Update

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

North American office
Xephon/QNA
1301 West Highway 407, Suite 201-405
Lewisville, TX 75067
USA
Telephone: 940 455 7050

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

Contributions
Articles published in *VM Update* are paid for at the rate of £170 ($250) per 1000 words for original material. To find out more about contributing an article, without any obligation, please contact us at any of the addresses above and we will send you a copy of our *Notes for Contributors*.

Editor
Trevor Eddolls

Disclaimer
Readers are cautioned that, although the information in this journal is presented in good faith, neither Xephon nor the organizations or individuals that supplied information in this journal give any warranty or make any representations as to the accuracy of the material it contains. Neither Xephon nor the contributing organizations or individuals accept any liability of any kind howsoever arising out of the use of such material. Readers should satisfy themselves as to the correctness and relevance to their circumstances of all advice, information, code, JCL, EXECs, and other contents of this journal before making any use of it.

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

VM Update on-line
Code from *VM Update* can be downloaded from our Web site at http://www.xephon.com; you will need the user-id shown on your address label.

Printed in England.

Finding files containing a specified string

The following is a useful utility. It will find which files contain a particular string.

The syntax is:

```
search /<string>/ <filter>
```

where <filter> is the same as for LISTFILE; the filemode can also be specified as DIRS or ALL. If DIRS is used, it scans a user's directories; if ALL, it scans all directories and then all filemodes (but not those directories already scanned).

The separator '/' can be any character.

```
/* */
addr = Address()
address command

Parse source . callType . . . myname .
Upper myName
If callType<>'SUBROUTINE' then do
'PIPE COMMAND QUERY INSTSEG | SPEC W3 | VAR INSTSEG'  /*for future use*/
'SET INSTSEG ON A'
end

Parse arg argg
argg = 'STRIP'(argg,'B')
parse var argg sep 2
interpret "Parse var Argg '"sep"'arg'"sep"' filter"
Upper Filter
Parse var filter ffn fft ffm

If fFm = 'DIRS' | fFm = 'ALL' then do        /*  scan all sfs dirs   */
   'PIPE COMMAND LISTDIR | DROP 1 | STEM DIRS.'
   freemode = modefind()
   do i=1 to dirs.Ø  until result=23         /*  until interrupted   */
     parse var dirs.i mode dir
     If mode = '-' then mode = dir
     Interpret 'call "'myname'" "'sep||arg||sep ffn fft mode'"'
   end i
   If result = 23                            /*  Interrupted ?       */
      Then do
```

```
         rrc=23
         signal exitt
       end
   If fFm = 'DIRS' then signal exitt         /*else (fFm='ALL') go on*/
   'PIPE COMMAND QUERY DISK | DROP 1 | STEM ACCESSED.'
    do i=1 to accessed.Ø
      parse var accessed.i . vdev mode .     /*ignore things like Y/S*/
      if vdev = 'DIR' then iterate i         /*already scanned       */
      interpret 'call "'myname'" "'sep||arg||sep ffn fft ,
                                substr(mode,1,1) '"'
    end i
    signal exitt
end


If index(ffm,'.')<>Ø                         /* single dir           */
   then do
     'PIPE COMMAND LISTDIR' ffm '| DROP 1 | VAR DIR'
       if dir = '' then do; say 'Directory' ffm 'unavailable'
                        exit 16
                        end
      parse var dir dmode dir
      if dmode = '-' then do                  /*get mode for access */
          freemode = modefind()               /*determine available */
          'ACCESS' ffm freemode
            if rc<>Ø then return rc
          dMode = freeMode
          ModeGot = 1                        /*don't forget to release*/
      end
      else ModeGot = Ø

     where  = Filter                         /*will show on screen    */
     filter = ffn fft dmode
   end
   else where = Filter                       /* minidisk              */

'PIPE COMMAND LISTFILE' Filter ' | STEM FID.'
  if rcæ=Ø then do
   say 'No files' where
   If ModeGot = 1 then 'RELEASE' freeMode
   return rc
  end

say 'Searching for' arg 'in' where            /* mention dir if there*/

Target = sep || arg || sep
'EXECSTATE $SEARCH XEDIT'
if rc<>Ø then do
```

```
   queue '/**/"SET CASE M I # SET VARBLANK ON',
   queue '# LOCATE' target '"'
   queue 'If rc=Ø  then do;"MACRO PROFILE"'
   queue 'Mess = "To interrupt, enter STOP."'
   queue '"EXTRACT /EDIRNAME/"'
   queue 'If edirname.1<>"" then mess = mess "Directory:" edirname.1'
   queue '  " COMMAND SET SYNONYM STOP COMMAND QUIT 23',
           '# COMMAND SET MSGM ON # COMMAND MSG" Mess',
          '"# COMMAND CMSG &/'ARG'"'
   queue 'End'
   queue 'else "COMMAND QUIT"'
   'EXECIO' queued() 'DISKW $SEARCH XEDIT A1 1 (FINI'
   'EXECLOAD $SEARCH XEDIT A (PUSH'
end

Signal on Halt
Address 'COMMAND'
Do i=1 to Fid.Ø until rc=23   /* RC=23: user entered STOP in xedit */
  'XEDIT' fid.i '(NOMSG PROFILE $SEARCH'
     /* Note:
        "pipe < | locate" would be faster, but case-sensitive.
        "pipe < | CaseI locate " would be 2-3 times SLOWER.
     */
end i
If rc=23 then do;rrc=23;signal Halt;end
         else           Signal Exitt

Halt:Nop
Ms = 'Interrupted.' i 'of' fid.Ø 'files were processed.'
If Addr = 'XEDIT' then   address 'XEDIT' 'MSG' ms
                  else   say ms
Exitt: Nop
if ModeGot=1 then 'RELEASE' dMode     /* directory was accessed       */
If calltype <> 'SUBROUTINE'          /* not being called recursively?*/
  then do
    address command 'EXECDROP $SEARCH XEDIT'
    'SET INSTSEG' instseg
  end
If rrc=23 then return rrc            /* halted by 'STOP' in xedit    */
```

Vadim Rapp
Independent Consultant (USA)

Backing out using 3480/3490 cassette drives

This system, for managing back-up jobs, was designed in response to the decision not to use operators while running back-up jobs during nights and at weekends. Our installation was at that time equipped with seven 3480 (14 drives), all with Automatic Cartridge Loaders (ACLs). The system allowed us to set up a mixture of back-up jobs (normal mini-disk back-ups, various DDR back-ups, back-up of saved segments, etc), and make these jobs run more or less in parallel. We used VMBACKUP, VMTAPE, and VMSHEDULER from Sterling Software as the backbone for the daily production. This system made it possible to preload all the 14 drives with a near maximum number of cartridges and so speed up production. The system has at its centre a simple file, BACKUP PARMS, that controls the distribution of drives to the individual jobs. This file also allows us to mask out specific drives that are out of service, and also to skip certain or all jobs for specified dates. The BACKUP PARMS file is defined and maintained by the BACKUP EXEC.

The system has been working under VM/ESA 1.1 and 2.1.

SET UP

A user-id called BACKUP should be defined. This user-id will hold all the software, and a 191 mini-disk of three cylinders and a 192 mini-disk of one cylinder should be defined. The memory size of BACKUP needs to be only 6MB and must have a privilege class of B (DETACH). In addition to holding the software, BACKUP is also used for setting up tape drives for certain back-up jobs. This prevents interference from other jobs when the jobs are started at the same time.

Additional user-ids, BACKUP1 to BACKUP*n*, should be defined, each with a 191 disk of one cylinder and privilege class B (DETACH and AUTOLOG). The memory size should be 8MB and a directory statement linking BACKUP's 191 mini-disk as 192 in RR mode should be included.

These user-ids are used while starting back-up jobs. As many user-ids

as needed should be defined. The BACKUP*x* user-ids are active for all the time it takes to perform a DDR back-up and are, therefore, not able to accept any new jobs.

OPERATION

Each day, just before leaving for the day, the operator sets up the 3480s with cartridges in accordance with a pre-defined scheme, and the rest is up to the system.

A job is started, by VMSCHEDULER, by executing the corresponding start-up EXEC (E21RES or MONTHLY) on one of the BACKUP*x* user-ids. The start-up EXEC then performs all the necessary housekeeping operations and calls STADRAIN EXEC to get hold of the specified tape drives. When the drives are reserved, the start-up EXEC starts performing the job, or submits it to VMBACKUP. When completed, it logs off.

CONTENT

The BACKUP system consists of the following files.

EXECs:

- BACKUP EXEC – calls EXECSCRN MODULE. It supplies the user interface for job set up. It creates and maintains the BACKUP PARMS file.
- SKIP EXEC – determines whether a job should not run on the present day. It is called by all job-initializing EXECs.
- STADRAIN EXEC – calls TAPSTART EXEC and TAPDRAIN EXEC. It reserves tape drives for the job. It is called by all job-initializing EXECs.
- TAPSTART EXEC – calls VMTAPE to start a tape drive for the job. It also DETACHes a drive from a user who occupies it, and is therefore preventing a back-up job from running.
- TAPDRAIN EXEC – calls VMTAPE to drain a drive in order to ensure that the tape drive will be selected for this job.

- REMVSKIP EXEC – periodically executed to remove obsolete SKIP records from the BACKUP PARMS file.
- TRANSLOG EXEC – retrieves class U files (status files) from the reader. It renames the file to the name of the job that caused it to be sent. It transfers the renamed file to a specified user-id for archiving.
- BKPMAINT EXEC – sets up the environment for the operator so that he/she will be able to create/update jobs on the BACKUP PARMS file.
- MONTHLY EXEC – sample EXEC for transferring a job to VMBACKUP.
- E21RES EXEC – sample EXEC for starting a DDR back-up for MAINT's 123 disk. Calls VMTTAPE, XDDR MODULE, WRITLINE MODULE, and TRANSLOG EXEC. XDDR and WRITLINE are described in *VM Update* March 1997.
- RESTMDDR EXEC – sample EXEC for restoring previously backed out DDRs, using manually mounted tapes. Calls VMTAPE, XDDR MODULE, and WRITLINE MODULE.
- RESTADDR EXEC – sample EXEC for restoring previously backed out DDRs, using automatic tape mounts. Calls VMTAPE, XDDR MODULE, and WRITLINE MODULE.

MODULEs:

- EXECSCRN – called by BACKUP EXEC (*VM Update* June 1987 – freely available from the Xephon Web site).
- XDDR – called by E21RES EXEC (*VM Update* March 1997)
- WRITLINE – called by E21RES EXEC and XDDR MODULE (*VM Update* March 1997).

Parameter files:

- BACKUP PARMS – file that defines the drives to be used for individual jobs. Defines drives that should be temporarily drained (excluded). It also defines dates when a regular back-up job for some reason should not run.

- BACKUP PANEL – panel definition for BACKUP EXEC (EXECSCRN).

BACKUP EXEC

The environment that BACKUP EXEC uses is established by BKPMAINT EXEC. This EXEC should reside on a disk that is normally available to the operators and others who need to have access. BKPMAINT calls BACKUP EXEC, which brings up the start menu. Figure 1 shows what this screen looks like. On this screen one can define a new back-up job, select already defined jobs for definition, or display status from the BACKUP PARMS file. This is achieved by typing an existing job name and pressing the appropriate PF key.

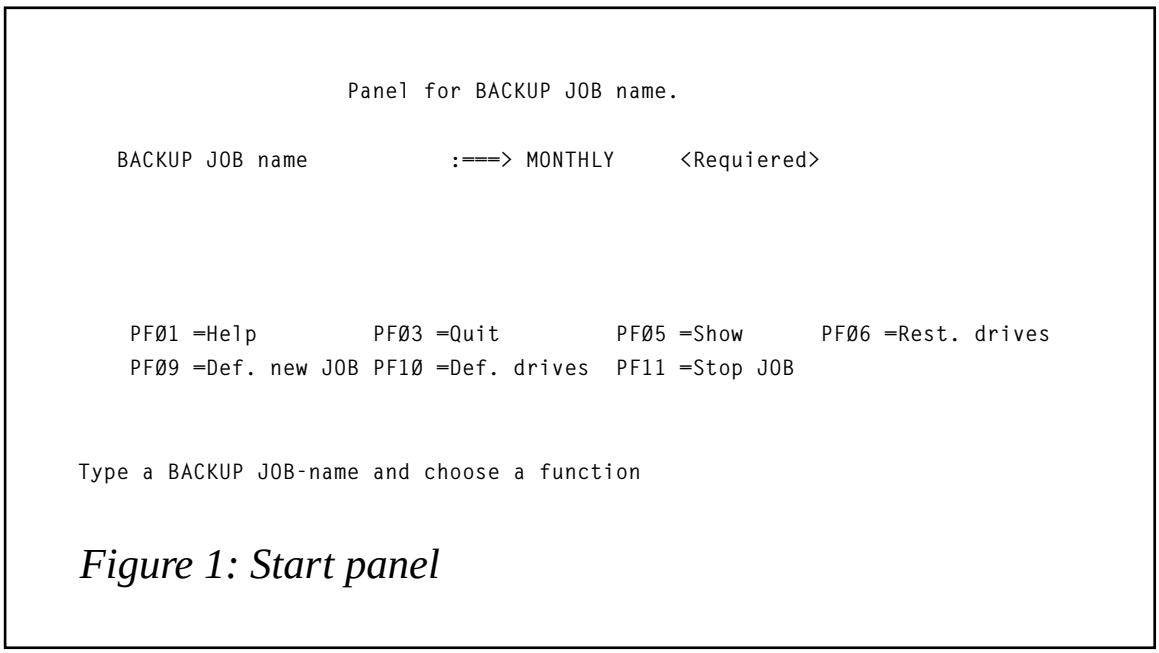

Figure 1: Start panel

Associated with this screen we have the following responses:

1 Wrong response.

2 Missing jobname.

3 JOB ‘jobname’ already defined.

4 JOB ‘jobname’ correctly defined.

5 You can only have two active drivelists for ‘jobname’.

Restore before you define a new list, or remove manually.

6 There is nothing to restore for ‘jobname’.

7 Jobname not correct or jobname not defined in BACKUP PARMS file.

When you hit PF09, you initialize for a new job. There is no screen connected to this function.

When you hit PF10, you are able to define tape drives that are to be connected to the job. Figure 2 shows what this screen looks like with some data entered. If you enter data to a job that already has drives defined, it is considered that this is a temporary definition. The old definition is saved in the file, but now with a ‘*’ in front of the jobname. Later this saved definition can be restored from the start menu (by PF06). Only one saved definition can be maintained. Should

```
        Panel for defining tape drives to a BACKUP JOB.

     START tape     :===> 886     881     882     883    <Max. 4 drives
                    :===> 884     885     886             to each line>
                    :===>
                    :===>

     DRAIN tape     :===> 887     888     889            <Max. 3 drives
                    :===> 88A     88B     88C             to each line>
                    :===> 88D
                    :===>
                    :===>




   PF03 =Main menu                                PF05 =Process
                   PF11 =Panel for stop date


Type addresses for drives to be used in the MONTHLY BACKUP JOB
Remember also to define drives that are not to be used for this JOB
```

Figure 2: Tape drives definition panel

there be a need to make another redefinition, then the first one (with '*') must be removed manually,

Associated with this screen we have the following responses:

1 Wrong response.

2 Two equal addresses in START list. Correct and press PF5.

3 Two equal addresses in DRAIN list. Correct and press PF5.

4 Wrong tape addresses in START list. Correct and press PF5.

5 Wrong tape addresses in DRAIN list. Correct and press PF5.

6 The sum of drives STARTed and drives DRAINed must be 'xx'. Correct and press PF5.

7 The same tape drive is 'STARTed' and 'DRAINed'. Correct and press PF5.

When you choose the function PF05 from the start screen you will be presented with a screen looking like Figure 3. This is a purely informative screen with no data entry possible.

When you choose function PF11 from the drive definition screen or from the start screen, you enter the screen for defining or dropping stop dates. Figure 4 shows what this screen looks like before any data is entered. You define a stop date by entering the year, month, and day in the appropriate fields. If you want to delete a previously entered stop date you can do that by entering the appropriate date together with the letters 'RE' in the CMNT field.

Associated with this screen we have the following responses:

1 Wrong response.

2 Incorrect value for month number. Correct and press PF5.

3 Day number is in error. Correct and press PF5.

4 Each number must have two digits, eg 05 02 92, but not (5. Correct and press PF5.

5 One of the dates is today's date. Correct and press PF5.

```
                    Display parameters defined for
                                MONTHLY

 START Ø884 Ø886 Ø887
 DRAIN Ø88Ø Ø881 Ø882 Ø883 Ø885 Ø888 Ø889 Ø88A Ø88B Ø88C Ø88D

*START Ø885 Ø886 Ø887
*DRAIN Ø88Ø Ø881 Ø882 Ø883 Ø884 Ø888 Ø889 Ø88A Ø88B Ø88C Ø88D









 Lines marked  *  can be retrieved (reactivated)
 by choosing restore from the Main menu.


                                PFØ3 =Main menu
                                CLEAR =Show more
```

Figure 3: PARMS file status display

Figure 5 shows the BACKUP PARMS file with three jobs defined. Note that there can only be *one* space between the job name and the record identifier (JOB, START, DRAIN and SKIP).

BACKUP EXEC

```
/*                                                                  */
/********************************************************************/
/*                                                                  */
/* BACKUP EXEC maintains back-up jobs by the use of a central file, */
/* BACKUP PARMS. This file contains information pertinent           */
/* to the individual jobs. The information consists of addresses of */
/* tape drives that are to be used by the job as well as tape       */
```

```
          Panel for defining dates when a BACKUP JOB is not to be performed

                                          DAY   MNT  YEAR    CMNT
          Skip JOB (date: dd mm yy   :===>
                                     :===>
                                     :===>
                                     :===>
                                     :===>
                                     :===>
                                     :===>
                                     :===>




         PFØ3 =Main menu                        PFØ5 =Process


    Type date(s) for possible stop(s) of MONTHLY BACKUP

          If you want to remove a date from the list:
          Type the date and also RE in the CMNT field
```

Figure 4: Job skip panel

```
/* addresses that has to be made inaccessible during job start-up.    */
/* The file also contains information on the date(s) when a           */
/* particular job, for any reason, should not be run.                 */
/* BACKUP EXEC uses EXECSCRN as presented in VM Update, Issue 1Ø,     */
/* June 1987                                                          */
/*                                                                    */
/**********************************************************************/
/*                                                                    */
month = '31 28 31 3Ø 31 3Ø 31 31 3Ø 31 3Ø 31'
jobname = ''; name = ''
tape_panel = Ø
date_panel = Ø
parse value copies(' ',2) with c1 c2 c3 c4 c5 c6 c7 c8
/*                                                                    */
/**********************************************************************/
/*                                                                    */
/* Next six lines contain installation dependent variables and must  */
```

```
MONTHLY JOB
WEEKLY JOB
E21RES JOB
*MONTHLY START  0880 0881 0882 0883 0884 0885 0886 0887
MONTHLY START  0881 0882 0884 0885 0886 0887 0888 0889
*MONTHLY DRAIN  0888 0889 088A 088B 088C 088D
MONTHLY DRAIN  0880 0883 088A 088B 088C 088D
WEEKLY START  0887 0888 0889 088A
WEEKLY DRAIN  0880 0881 0882 0883 0884 0885 088B 088C 088D
E21RES START  0880 0881
E21RES DRAIN  0882 0883 0884 0885 0886 0887 0888 0889 088A 088B 088C 08D
WEEKLY SKIP  19970317
WEEKLY SKIP  19970318
```

Figure 5: BACKUP PARMS file example

```
/* be changed accordingly.                                              */
/* Pay attention to the two statements just in front of the tap1:       */
/* label and corresponding definition in BACKUP PANEL file              */
/*                                                                      */
/************************************************************************/
/*                                                                      */
fm  = 'M'                                          /* Defined by    */
fm2 = 'N'                                          /* BKPMAINT EXEC */
max_drain_count = 13
max_start_count = 14
numb_drives = 14
drivelist = '0880088108820883088408850886088708880889088A088B088C088D'
parse value copies(' ',2) with d1 d2 d3 d4 d5 d6 d7 d8
parse value copies(' ',2) with m1 m2 m3 m4 m5 m6 m7 m8
parse value copies(' ',2) with y1 y2 y3 y4 y5 y6 y7 y8
parse value copies(' ',2) with msg10   msg11   msg20   msg21   msg30,
                               msg31   msg40
parse value copies(' ',2) with msg4.1  msg4.2  msg4.3  msg4.4  msg4.5 ,
                               msg4.6  msg4.7  msg4.8  msg4.9  msg4.10,
                               msg4.11
/*                                                                    */
/************************************************************************/
/*                                                                      */
/* Display entry panel                                                  */
/*                                                                      */
/************************************************************************/
/*                                                                      */
```

```
job:
jobname = name
if msgsw = Ø then
do
  msg1Ø = ''; msg11 = ''
end
'EXECSCRN BACKUP PANEL *START *END (ALARM NOTRANS NOCLEAR'
if key = 'PFKØ1' then signal help
if key = 'PFKØ3' then exit
if key = 'PFKØ5' then signal display
if key = 'PFKØ6' then
do
  name = jobname
  signal restore
end                  /* end do*/
if key = 'PFKØ9' then signal new_job
if key = 'PFK1Ø' then
do
  date_panel = Ø
  signal tape
end
if key = 'PFK11' then
do
  tape_panel = Ø
  signal skip
end
if key = 'ENTER' then
do
  msgsw = Ø
  signal job
end
msgsw = 1
msg1Ø = 'Wrong response'
msg11 = ''
signal job
/*                                                                  */
/********************************************************************/
/*                                                                  */
/* Define a new job (Selected from entry panel)                     */
/*                                                                  */
/********************************************************************/
/*                                                                  */
new_job:
if jobname = '' then
do
  msgsw = 1
  msg1Ø = 'Missing jobname'
  msg11 = ''
  signal job
end
```

```
'pipe',
'< BACKUP PARMS' fm ,
'| locate /'||jobname 'JOB'||'/',
'| stem result.'
 if result.Ø ¬= Ø then
 do
   msg1Ø = 'JOB' jobname 'already defined'
   msgsw = 1
   signal job
 end
 name = translate(strip(jobname))
 'pipe',
 '< BACKUP PARMS' fm ,
 '| stem parms.'
 line = name 'JOB'
 'pipe',
 'var line',
 '| > BACKUP PARMS' fm 'F 8Ø'
 'pipe',
 'stem parms.',
 '| >> BACKUP PARMS' fm
  msgsw = 1
  msg1Ø = 'JOB' name 'correctly defined'
  msg11 = ''
 signal job
/*                                                                  */
/********************************************************************/
/*                                                                  */
/* Display panel for definition of tape drives to be used by the    */
/* specified job                                                    */
/*                                                                  */
/********************************************************************/
/*                                                                  */
tape:
call check_jobname
'pipe',
'< BACKUP PARMS' fm ,
'| locate /'||'*'||jobname||'/',
'| stem result.'
 if result.Ø ¬= Ø then
do
  msg1Ø = 'You can only have two active drivelists for' jobname'.'
  msg11 = 'Restore before you define a new list, or remove manually.'
  msgsw = 1
  signal job
end                     /* end do */
name = jobname
parse value copies(' ',2) with s.1 s.2 s.3 s.4 s.5 s.6 s.7 s.8 s.9 ,
                               s.1Ø s.11 s.12 s.13 s.14
parse value copies(' ',2) with d.1 d.2 d.3 d.4 d.5 d.6 d.7 d.8 d.9 ,
```

```
                                d.1Ø d.11 d.12 d.13
tape1:
tape_panel = Ø
'EXECSCRN BACKUP PANEL *START1 *END1 (ALARM NOTRANS NOCLEAR'
if key = 'PFKØ3' then
do
  msgsw = Ø
  signal job
end
if key = 'PFKØ5' then
do
  tape_panel = 1
  signal check_tape
end                       /* end do */
if key = 'PFK11' then
do
  tape_panel = 1
  signal skip
end
if key = 'ENTER' then signal tape1
msg2Ø = 'Wrong response'
signal tape1
/*                                                                    */
/**********************************************************************/
/*                                                                    */
/* Display panel for definition of dates when a specified job         */
/* should not run (skipped)                                           */
/*                                                                    */
/**********************************************************************/
/*                                                                    */
skip:
call check_jobname
name = jobname
parse value copies(' ',2) with c1 c2 c3 c4 c5 c6 c7 c8
parse value copies(' ',2) with d1 d2 d3 d4 d5 d6 d7 d8
parse value copies(' ',2) with m1 m2 m3 m4 m5 m6 m7 m8
parse value copies(' ',2) with y1 y2 y3 y4 y5 y6 y7 y8
skip1:
date_panel = Ø
'EXECSCRN BACKUP PANEL *START2 *END2 (ALARM NOTRANS NOCLEAR'
if key = 'PFKØ3' then
do
  msgsw = Ø
  signal job
end
if key = 'PFKØ5' then
do
  date_panel = 1
  signal check_date
end                       /* end do */
```

```
if key = 'ENTER' then signal skip1
msg3Ø = 'Wrong response'
signal skip1
restore:
call check_jobname
'pipe',
'< BACKUP PARMS' fm ,
'| locate /'||'*'||jobname||'/',
'| stem result.'
 if result.Ø = Ø then
do
  msgsw = 1
  msg1Ø = 'There is nothing to restore for' jobname
  msg11 = ''
  signal job
end                     /* end do */
'pipe',
'< BACKUP PARMS' fm ,
'| stem parms.'
'ERASE BACKUP PARMS' fm
do k = 1 to parms.Ø
  if word(parms.k,1) = jobname & (word(parms.k,2) = 'START' |,
                       word(parms.k,2) = 'DRAIN') then parms.k = ''
  if word(parms.k,1) = '*'||jobname then parms.k = strip(parms.k,L,'*'
)
end                      /* end do while queued()  */
'pipe',
'stem parms.',
'| >> BACKUP PARMS' fm
call copy_disk
signal display
/*                                                                  */
/********************************************************************/
/*                                                                  */
/* Show status from BACKUP PARMS file for the specified job         */
/*                                                                  */
/********************************************************************/
/*                                                                  */
display:
call check_jobname
parse value copies(' ',2) with msg4.1 msg4.2 msg4.3 msg4.4 msg4.5 ,
                         msg4.6 msg4.7 msg4.8 msg4.9 msg4.1Ø msg4.11
msg4Ø = jobname
j = 1
'pipe',
'< BACKUP PARMS' fm ,
'| stem parms.'
do k = 1 to parms.Ø
  if word(parms.k,1) = jobname | word(parms.k,1) = '*'||jobname then
  do
```

```
    if word(parms.k,2) ¬= 'JOB' then
    if word(parms.k,1) = jobname then
    do
      if word(parms.k,2) = 'START' then msg4.1 = 'START'
subword(parms.k,3)
      if word(parms.k,2) = 'DRAIN' then msg4.2 = 'DRAIN'
subword(parms.k,3)
      if word(parms.k,2) = 'SKIP' then
      do
        skip.j = subword(parms.k,3)
        j = j + 1
      end
    end
    if word(parms.k,1) = '*'||jobname then
    do
      if word(parms.k,2) = 'START' then msg4.3 = '*'||'START'
subword(parms.k,3)
      if word(parms.k,2) = 'DRAIN' then msg4.4 = '*'||'DRAIN'
subword(parms.k,3)
      if word(parms.k,2) = 'SKIP' then
      do
        skip.j = subword(parms.k,3)
        j = j + 1
      end
    end
  end
end            /* end do while queued()  */
j = j - 1
i = 5
more:
if j = Ø then signal show
sk = strip(skip.j)
msg4.i = 'SKIP' substr(sk,7,2)||'/'||substr(sk,5,2)||'/'||,
                substr(sk,3,2)
j = j - 1
i = i + 1
if i > 11 then
do
  i = 5
  signal show
end
signal more
show:
'EXECSCRN BACKUP PANEL *START3 *END3 (ALARM NOTRANS NOCLEAR'
if key = 'PFKØ3' then
do
  msgsw = Ø
  signal job
end
if key = 'CLEAR' then
```

```
do
  parse value copies(' ',2) with msg4.5 msg4.6 msg4.7 msg4.8 msg4.9 ,
                                 msg4.1Ø msg4.11
  signal more
end
/*                                                                    */
/**********************************************************************/
/*                                                                    */
/* Check whether tape drives are correctly defined.                   */
/*                                                                    */
/**********************************************************************/
/*                                                                    */
check_tape:
msg2Ø = ''
msg21 = ''
if tape_panel = 1 then
do
  call check_jobname
  start = Ø
  drainlist = ''
  startlist = ''
  do i = 1 to max_start_count
     s.i = translate(strip(s.i))
     if s.i ¬= '' & left(s.i,1) ¬= 'Ø' & length(s.i) < 4 then
     do
       s.i = 'Ø's.i
     end
  end
  do i = 1 to max_drain_count
     d.i = translate(strip(d.i))
     if d.i ¬= '' & left(d.i,1) ¬= 'Ø' & length(d.i) < 4 then
     do
       d.i = 'Ø'd.i
     end
  end
  do i = 1 to max_start_count - 1
    do j = i + 1 to max_start_count
      if s.i ¬= '' | s.j ¬= '' then
      if s.i = s.j then
      do
        msg2Ø = 'Two equal addresses in START list.'
        msg21 = 'Correct and press PF5'
        signal tape1
      end
    end
  end
  do i = 1 to max_drain_count - 1
    do j = i + 1 to max_drain_count
      if d.i ¬= '' | d.j ¬= '' then
      if d.i = d.j then
```

```
      do
        msg2Ø = 'Two equal addresses i DRAIN list.'
        msg21 = 'Correct and press PF5.'
        signal tape1
      end
    end
  end
  do i = 1 to max_start_count
     if s.i ¬= '' then
     do
       if index(drivelist,s.i) = Ø then
       do
         msg2Ø = 'Wrong tape addresses in START list.'
         msg21 = 'Correct and press PF5'
         signal tape1
       end
       else
       do
         startlist = startlist  s.i
         start = start + 1
       end
     end
  end
  drain = Ø
  do i = 1 to max_drain_count
     if d.i ¬= '' then
     do
       if index(drivelist,d.i) = Ø then
       do
         msg2Ø = 'Wrong tape addresses in DRAIN list.'
         msg21 = 'Correct and press PF5'
         signal skip1
       end
       else
       do
         drainlist = drainlist d.i
         drain = drain + 1
       end
     end
  end
  if start + drain = Ø  then signal check_date
  if start + drain ¬= 14 then
  do
    msg2Ø = 'The sum of drives STARTed and drives DRAINed must be'
numb_drives
    msg21 = 'Correct and press PF5'
    signal tape1
  end
  do i = 1 to max_start_count
    if s.i ¬= '' then
```

```
    do j = 1 to max_drain_count
      if d.j ¬= '' then
        if s.i = d.j then
        do
          msg2Ø = 'The same tape drive is "STARTed" and "DRAINed"'
          msg21 = 'Correct and press PF5'
          signal tape1
        end
    end
  end
end
/*                                                                  */
/********************************************************************/
/*                                                                  */
/* Check the input of dates. If OK combine one to form 19yymmdd.    */
/*                                                                  */
/********************************************************************/
/*                                                                  */
check_date:
msg3Ø = ''
msg31 = ''
if date_panel = 1 then
do
  date = date(s)
cØ = c1 c2 c3 c4 c5 c6 c7 c8
dØ = d1 d2 d3 d4 d5 d6 d7 d8
mØ = m1 m2 m3 m4 m5 m6 m7 m8
yØ = y1 y2 y3 y4 y5 y6 y7 y8
parse var cØ  c.1 c.2 c.3 c.4 c.5 c.6 c.7 c.8
parse var dØ  d.1 d.2 d.3 d.4 d.5 d.6 d.7 d.8
parse var mØ  m.1 m.2 m.3 m.4 m.5 m.6 m.7 m.8
parse var yØ  y.1 y.2 y.3 y.4 y.5 y.6 y.7 y.8
  do i = 1 to 8
    if m.i ¬= '' & d.i ¬= '' & y.i ¬= '' then
    do
      if m.i = Ø | m.i > 12 then
      do
        msg3Ø = 'Incorrect value for month number'
        msg31 = 'Correct and press PF5'
        signal skip1
      end
      if y.i = 96 & m.i = 'Ø2' then
        dag = 29
      else
      dag = word(month,m.i)
      if d.i = Ø | d.i > dag then
      do
        msg3Ø = 'Day number is in error'
        msg31 = 'Correct and press PF5'
        signal skip1
```

```
      end
    end
  end
  do i = 1 to 8
    date.i = '19'|| strip(y.i)||strip(m.i)||strip(d.i)
    if date.i ¬= '19' then
    do
      if length(date.i) ¬= 8 then
      do
        msg3Ø = 'Each number must have two digits, ie. Ø5 Ø2 92
(5.feb.92)'
        msg31 = 'Correct and press PF5'
        signal skip1
      end
      if date.i < date then
      do
        msg3Ø = 'One of the dates are the date of today'
        msg31 = 'Correct and press PF5'
        signal skip1
      end
    end
  end
end
/*                                                                  */
/********************************************************************/
/*                                                                  */
/* Drive definitions and skip dates have been given. Put the results */
/* into BACKUP PARMS file. New dates will be added to the file.     */
/* Dates that are passed will be removed by a job that is scheduled  */
/* to run each morning.                                             */
/* There can only be one START record (line) and one DRAIN record   */
/* in BACKUP PARMS file. Extra DRAIN and START records will be      */
/* replaced by an empty record. These records will be removed by    */
/* the same scheduled job.                                          */
/* Old START or DRAIN records are kept but now preceded with a '*',  */
/* ie *MONTHLY DRAIN 884 885 886. The operator can later choose     */
/* to reactivate this old status (choice from Main menu)            */
/*                                                                  */
/********************************************************************/
/*                                                                  */
startline = Ø
drainline = Ø
jobname = translate(jobname)
'pipe',
'< BACKUP PARMS' fm ,
'| stem parms.'
'ERASE BACKUP PARMS' fm
if tape_panel = 1 & date_panel = 1 then
do k = 1 to parms.Ø
```

```
  if subword(parms.k,1,1) = jobname & subword(parms.k,2,1) = 'START'
then
  do
    if startline = 1 then parms.k = ''
    else
    do
      newline = '*'||parms.k
      'pipe',
      'var newline',
      '| >> BACKUP PARMS' fm
      parms.k = jobname 'START' startlist
      startline = 1
    end
  end
  if subword(parms.k,1,1) = jobname & subword(parms.k,2,1) = 'DRAIN'
then
  do
    if drainline = 1 then parms.k = ''
    else
    do
      newline = '*'||parms.k
      'pipe',
      'var newline',
      '| >> BACKUP PARMS' fm
      parms.k = jobname 'DRAIN' drainlist
      drainline = 1
    end
  end
  if subword(parms.k,1,1) = jobname & subword(parms.k,2,1) = 'SKIP' then
  do
    do i = 1 to 8
      if subword(parms.k,3,1) = date.i & translate(c.i) ¬= 'RE' then
      'pipe',
      'var parms.k',
      '| >> BACKUP PARMS' fm
      else
        if date.i ¬= '19' then
          line = jobname 'SKIP' date.i
          'pipe',
          'var line',
          '| >> BACKUP PARMS' fm
    end
  end
  'pipe',
  'var parms.k',
  '| >> BACKUP PARMS' fm
end
/*                                                                    */
/**********************************************************************/
```

```
/*                                                                    */
/* Only tape drive data is given. Put into BACKUP PARMS file.         */
/*                                                                    */
/**********************************************************************/
/*                                                                    */
if tape_panel = 1 & date_panel = Ø then
do
  present = Ø
  do k = 1 to parms.Ø
    if subword(parms.k,1,1) = jobname & subword(parms.k,2,1) = 'START'
then
    do
      present = 1
      if startline = 1 then parms.k = ''
      else
      do
        newline = '*'||parms.k
        'pipe',
        'var newline',
        '| >> BACKUP PARMS' fm
        parms.k = jobname 'START' startlist
        startline = 1
      end
    end
    if subword(parms.k,1,1) = jobname & subword(parms.k,2,1) = 'DRAIN'
then
    do
      present = 1
      if drainline = 1 then parms.k = ''
      else
      do
        newline = '*'||parms.k
        'pipe',
        'var newline',
        '| >> BACKUP PARMS' fm
        parms.k = jobname 'DRAIN' drainlist
        drainline = 1
      end
    end
    'pipe',
    'var parms.k',
    '| >> BACKUP PARMS' fm
  end
  if present = Ø then
  do
    line = jobname 'START' startlist
    'pipe',
    'var line',
    '| >> BACKUP PARMS' fm
```

```
    line = jobname 'DRAIN' drainlist
    'pipe',
    'var line',
    '| >> BACKUP PARMS' fm
  end
end
/*                                                                  */
/********************************************************************/
/*                                                                  */
/* Only skip dates are given. Put these into BACKUP PARMS file.     */
/*                                                                  */
/********************************************************************/
/*                                                                  */
if tape_panel = Ø & date_panel = 1 then
do
  do k = 1 to parms.Ø
    if subword(parms.k,1,1) = jobname & subword(parms.k,2,1) = 'SKIP'
then
    do
      do i = 1 to 8
        if subword(parms.k,3,1) = date.i & translate(c.i) = 'RE' then
        leave
        else
        do
          'pipe',
          'var parms.k',
          '| >> BACKUP PARMS' fm
          leave
        end
      end
    end
    else
    'pipe',
    'var parms.k',
    '| >> BACKUP PARMS' fm
  end
  do i = 1 to 8
    if translate(c.i) ¬= 'RE' then
    if date.i ¬= '19' then
    do
      line = jobname 'SKIP' date.i
      'pipe',
      'var line',
      '| >> BACKUP PARMS' fm
    end
  end
end
call copy_disk
exit
```

```
/*                                                                    */
/**********************************************************************/
/*                                                                    */
/* First, check if jobname has been defined in BACKUP PARMS file.     */
/* The job has to be defined with a "jobname JOB" record. This        */
/* record must be manually defined. There may only be one 'space'     */
/* between the jobname and the parameter 'JOB', ie MONTHLY JOB.       */
/* Only one line for each job.                                        */
/*                                                                    */
/**********************************************************************/
/*                                                                    */
check_jobname:
if jobname = '' then
do
  msgsw = 1
  msg1Ø = 'Missing jobname'
  msg11 = ''
  signal job
end
jobname = translate(strip(jobname))
name = jobname
'pipe',
'< BACKUP PARMS' fm ,
'| locate /'||jobname 'JOB'||'/',
'| stem result.'
 if result.Ø ¬= Ø then
 return
else
do
  msgsw = 1
  msg1Ø = 'Jobname not correct or jobname'
  msg11 = 'not defined in BACKUP PARMS file'
  signal job
end
copy_disk:
'COPY BACKUP PARMS' fm '= PARMSAVE' fm2 '(REPL OLDD'
call display
return
/*                                                                    */
/**********************************************************************/
/*                                                                    */
/* Display information (help)                                         */
/*                                                                    */
/**********************************************************************/
/*                                                                    */
help:
'EXECSCRN BACKUP PANEL *START4 *END4 (ALARM NOTRANS NOCLEAR'
if key = 'PFKØ3' then
do
```

```
  msgsw = Ø
  signal job
end
if key = 'PFKØ8' then
'EXECSCRN BACKUP PANEL *START5 *END5 (ALARM NOTRANS NOCLEAR'
if key = 'PFKØ7' then signal help
if key = 'PFKØ3' then
do
  msgsw = Ø
  signal job
end
```

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

Odd Hatlevold
Senior Systems Programmer
Statoil (Norway)

Finding unique records

GENERAL DESCRIPTION

When a large table is processed with SQL/DS and the DISTINCT option is specified in an SQL statement, then none of the records found nor any information about the size of the search results are passed to the application program before the end of search. Commonly it will take a long time before the user program begins to receive the search results.

To get information during the database search operation, especially when the size of the expected result is unknown, it is convenient to remove the DISTINCT option from the SQL statement. This allows the application program directly to control the size of the extracted SQL/DS output. In addition, the SQL/DS response time will be greatly decreased.

So, the CMS DISTINCT command is useful in the following cases:

- When unique records have to be selected from SQL/DS output and the SELECT statement is specified without the DISTINCT option.

- The CMS file contains duplicate records, which must be eliminated.

CMS DISTINCT is written in Assembler. It runs under CMS with VM/SP Release 5.

MEMORY REQUIREMENTS

For files with fixed-length records, a buffer area of up to 2MB may be allocated to accelerate read/write operations.

The size of CMS DISTINCT is 748 bytes.

CMS DISTINCT USAGE

CMS DISTINCT sequentially reads and compares records in an input file and writes unique records to the output file. The output file has the same filename and filemode as the original file, but the filetype is changed to DISTINCT.

The comparison checks up to 10 fields. This requires the input file to be sorted according to the fields being checked.

The CMS DISTINCT command is invoked as shown:

```
DISTINCT <fn> <ft> <fm> <FB1 FE2> [<FB2 FE2>] ...
```

where: fn is the input file filename, ft is the input file filetype, and fm is the input file filemode. FBn FEn define the *n*th checked field, where FBn and FEn are the starting and the ending columns of a checked field and n is between 1 and 10 inclusive.

Because of CMS conventions, CMS DISTINCT's code is optimized to process files with fixed-length records only. When files with variable-length records are processed, then CMS DISTINCT will work slowly, at about 1/25 of its ordinary speed.

To simplify the Assembler text, only CMS error codes are displayed if CMS DISTINCT terminates abnormally.

The following are possible reasons for error conditions occurring:

- Input file <fn> <ft> <fm> not found.
- Wrong checked field definition or the input field is greater than

255 bytes.

- Not enough virtual memory to allocate I/O buffers.
- Old file <fn> DISTINCT <fm> already exists. It must be renamed or erased.
- No space on disk <fm> to write file <fn> DISTINCT <fm>.

Here are examples of CMS DISTINCT usage:

- File FN FT A is sorted on fields [1 2]:

```
DISTINCT FN FT A 1 2
DISTINCT FN FT A 1 3                invalid
DISTINCT FN FT A 4 5                invalid
```

- File FN FT A is sorted on fields [10 20]:

```
DISTINCT FN FT A 1Ø 15
DISTINCT FN FT A 1Ø 11
DISTINCT FN FT A 11 15              invalid
DISTINCT FN FT A 15 2Ø              invalid
```

- File FN FT A is sorted on fields [10 20 30 40 50 60]:

```
DISTINCT FN FT A 1Ø 11 3Ø 35 5Ø 6Ø
DISTINCT FN FT A 1Ø 2Ø 3Ø 31
DISTINCT FN FT A 1Ø 15
DISTINCT FN FT A 3Ø 31 1Ø 2Ø 5Ø 6Ø  invalid
DISTINCT FN FT A 5Ø 6Ø 1Ø 11        invalid
DISTINCT FN FT A 3Ø 35              invalid
```

Note: examples indicated as INVALID will execute normally, but the results will be incorrect.

INSTALL EXEC

```
/*****************************************************************/
/***                                                  ***         ***/
/*** INSTALL         generate DISTINCT MODULE         ***  DG'97  ***/
/***                                                  ***         ***/
/*****************************************************************/
/***   SIZE ØØØ31  VER 1.Ø MOD ØØ  TIME 17:Ø2:4Ø  DATE 22/Ø8/97   ***/
/*****************************************************************/

  CLRSCRN
  MESSAGE = 'user request'
```

```
  SAY ' --- Start DISTINCT MODULE generation - reply Y or N'
  PULL REPLY
  IF REPLY ¬= 'Y' THEN
  SIGNAL ERROR
  SET CMSTYPE HT
  SIGNAL ON ERROR
  MESSAGE = 'error when assemble' DISTINCT
  ASSEMBLE DISTINCT
  ERASE DISTINCT LISTING A
  MESSAGE = 'error when load' DISTINCT
  LOAD DISTINCT '(' NOMAP NOLIBE
  MESSAGE = 'error when genmod' DISTINCT
  GENMOD
  ERASE DISTINCT TEXT A
  SIGNAL OFF ERROR
  SET CMSTYPE RT
  SAY ' --- DISTINCT MODULE generated successfully'
  EXIT
ERROR:
  SET CMSTYPE RT
  SAY ' --- DISTINCT MODULE not generated due to' MESSAGE
```

DISTINCT ASSEMBLE

```
*********************************************************************
****                                                  ***         ****
**** DISTINCT          CMS distinct function          ***  DG'97  ****
****                                                  ***         ****
*********************************************************************
****   SIZE ØØ153  VER 1.Ø MOD ØØ  TIME 16:38:28  DATE 22/Ø8/97   ****
*********************************************************************
*                                                                   *
DISTINCT CSECT
         USING *,12
         LR    11,14
         MVC   INOUTDCB+8(18),8(1)
         LA    1,32(1)
         LA    1Ø,2Ø
         SR    9,9
NEXT     EQU   *
         CLI   Ø(1),X'FF'
         BE    LEAVE
         SR    15,15
         CLI   1(1),X'4Ø'
         BE    PACK
         CLI   2(1),X'4Ø'
         BE    PACK1
         LA    15,2
```

```
         B     PACK
PACK1    EQU   *
         LA    15,1
PACK     EQU   *
         STC   15,DOPACK+1
         OI    DOPACK+1,X'7Ø'
DOPACK   PACK  DOUBLE(8),Ø(1,1)
         CVB   15,DOUBLE
         STC   15,DATA(9)
         LA    9,1(9)
         LA    1,8(1)
         BCT   1Ø,NEXT
LEAVE    EQU   *
         SRL   9,1
         LTR   9,9
         BZ    RET
         LA    1Ø,INOUTDCB
         USING FSCBD,1Ø
         FSOPEN FSCB=INOUTDCB,ERROR=RET
         L     7,FSCBSIZE
         CLI   36(1),C'F'
         BNE   DOCRAWL
         MVC   ALLOC(4),=A(1Ø24*256)
         B     DOBUFFER
DOCRAWL  EQU   *
         LA    6,7(7)
         SRL   6,3
         SLL   6,1
         ST    6,ALLOC
DOBUFFER EQU   *
         STH   7,LRECL
         SR    Ø,Ø
         L     1,ALLOC
         SLL   1,2
         DR    Ø,7
         ST    1,FSCBANIT
         MH    1,LRECL
         ST    1,FSCBSIZE
         L     Ø,ALLOC
         DMSFREE DWORDS=(Ø),TYPE=USER,ERR=RET
         LR    7,1
         LA    15,DATA
         LA    14,GENAREA
         SR    Ø,Ø
         SR    1,1
DOGEN    EQU   *
         MVI   Ø(14),X'D5'
         IC    Ø,Ø(15)
         IC    1,1(15)
```

```
         SR    1,Ø
         BM    RET
         BCTR  Ø,Ø
         STC   1,1(14)
         STC   Ø,3(14)
         STC   Ø,5(14)
         MVI   2(14),X'4Ø'
         MVI   4(14),X'5Ø'
         MVC   6(4,14),BNE
         LA    15,2(15)
         LA    14,1Ø(14)
         BCT   9,DOGEN
         MVC   Ø(4,14),B
         LA    9,INITDONE
         LA    8,READ
         LR    6,7
         A     6,FSCBSIZE
         SR    2,2
         LA    3,WRITE
         MVC   FT(8),FSCBFT
READ     EQU   *
         MVC   FSCBFT(8),FT
         ST    7,FSCBBUFF
         FSREAD FSCB=INOUTDCB
         LR    5,7
         LTR   15,15
         BZ    DOBRANCH
         LTR   2,2
         BZ    FREEMAIN
         LA    3,FREEMAIN
         B     CLOSE
DOBRANCH EQU   *
         BR    9
WRITEBUF EQU   *
         AH    4,LRECL
         C     2,FSCBANIT
         BNE   WRITE
CLOSE    EQU   *
         MVC   FSCBFT,=CL8'DISTINCT'
         ST    2,FSCBANIT
         MH    2,LRECL
         ST    2,FSCBSIZE
         ST    6,FSCBBUFF
         FSWRITE FSCB=INOUTDCB,ERROR=RET
INITDONE EQU   *
         LA    9,GENAREA
         SR    2,2
         LR    4,6
         BR    3
```

```
WRITE    EQU   *
         LH    1,LRECL
         LR    15,1
         LR    Ø,4
         LR    14,5
         MVCL  Ø,14
         LA    2,1(2)
COMPARE  EQU   *
         AH    5,LRECL
         LR    15,5
         SR    15,7
         C     15,FSCBNORD
         BE    READ
GENAREA  DC    5ØX'Ø7ØØ'
FREEMAIN EQU   *
         L     Ø,ALLOC
         LR    1,7
         DMSFRET DWORDS=(Ø),LOC=(1)
RET      EQU   *
         BR    11
DOUBLE   DS    D
INOUTDCB FSCB  FORM=E
BNE      BNE   WRITEBUF
B        B     COMPARE
ALLOC    DS    F
LRECL    DS    H
FT       DS    CL8
DATA     DS    2ØC
         LTORG
         FSCBD
         END   DISTINCT
```

GETTING READY

The INSTALL EXEC should be used to generate the CMS command DISTINCT. Do not forget that CMS DISTINCT correctly processes only those files that are sorted in the same order as the checked fields.

Dobrin Goranov
Information Service Co (Bulgaria)

Measuring COBOL pictures

If you are a COBOL programmer or have to deal with COBOL programs, you probably sometimes have to know the length in bytes of a given File Description, or to know the length of Working Storage or the length of a COMMAREA, in case you also deal with CICS.

To automate the process of measuring how many bytes a set of PICTURE clauses represents, I developed a REXX macro editor to get the job done.

While editing your program or your copybook, just invoke the PIC utility at the editor command prompt, optionally with the arguments of the first and last line to measure.

This utility accepts any valid PIC statement, including editing pictures (ZZ.Z9 style), with the exception of the SYNC and OCCURS DEPENDING ON clauses.

Let's see with an example how it works. Assume you are editing the following file (line numbers at left):

```
000001 01  ANYTHING-AT-ALL.
000002     02 NAME             PIC X(30).
000003     02 ADDRESS          PIC X(75).
000004     02 FILLER REDEFINES ADDRESS.
000005        04  AD-STREET    PIC X(30).
000006        04  AD-NUMBER    PIC X(5).
000007        04  AD-CODE      PIC X(8).
000008        04  AD-CITY      PIC X(12).
000009        04  AD-COUNTRY   PIC X(20).
000010     02 MONTHS OCCURS 12.
000011        04   M-DATE      PIC 9(8).
000012        04   M-VALUE     PIC S9(7)V99 COMP-3.
000013        04   M-CODE   COMP
000014                 PIC 9(4).
```

If you simply type 'PIC' at the editor prompt, you will get the full length answer, ie 285 bytes.

If you ask, for example:

```
PIC 10 13
```

the answer will be 180 bytes (15 bytes under item 'MONTHS' times

12 OCCURS).

However, if you ask:

```
PIC 4 9
```

the answer will be zero, because the first line contains the word ‘REDEFINES’ and all the requested lines are under its influence. PIC always skips redefined areas. If you want to measure the redefined area, just avoid the REDEFINES keyword. In this example, you could ask:

```
PIC 5 9
```

and you would get the answer 75 bytes.

Also note the following – the last picture clause in the example is split across two lines. If you request:

```
PIC 13 13
```

the answer will be zero, because the clause is incomplete.

Instead, you should request:

```
PIC 13 14
```

to get 2 bytes as the answer. This is because a PIC clause is only considered to be fully read when the ending dot ‘.’ is encountered.

This utility expects your file to be a standard COBOL file, that is, comments as asterisks on column 7 (these lines will be ignored) and a useful area between columns 8 and 72. The text may be upper or lowercase. PIC does not check for valid syntax, so, if you do have syntax errors, you may get incorrect results.

```
/*** VM XEDIT MACRO ************************************************/
/*                                                                  */
/* PIC - COBOL PICTURE length calculation                           */
/*                                                                  */
/* Format: PIC  <initial_line>  <final_line>                        */
/*                                                                  */
/* Valid clauses: COMP, COMP-1, COMP-2, COMP-3, COMP-4, BINARY,     */
/*  PACKED-DECIMAL, REDEFINES, OCCURS, SIGN SEPARATE, INDEX, POINTER*/
/*                                                                  */
/* Invalid clauses: OCCURS DEPENDING ON,  SYNCHRONIZED              */
/*                                                                  */
```

```
/*********************************************************************/
arg  inicio fim .                          /* get arguments          */
"extract/size/line/"                       /* get filesize & line    */
start_line = line.1                        /* keep curline number    */
if inicio = "" then inicio = 1             /* check arguments        */
if fim = "" | fim = "*" then fim = size.1
if ¬(datatype(inicio,"W")) then do
    say " Invalid value " inicio
    exit
end
if ¬(datatype(fim,"W")) then do
    say " Invalid value " fim
    exit
end
focc  = 0
ok    = 0
nred  = 0
nsign = 0
total = 0
li    = ""
do k = inicio to fim                       /* from first to last line*/
   ":"k                                    /* set line k as current  */
   "extract/curline/"                      /* get line contents      */
   lin = left(curline.3,72)                /* only COBOL useful area */
   lin = translate(lin)                    /* uppercase line         */
   if substr(lin,7,1)="*" then iterate k   /* skip comment line      */
   li = li strip(lin)                      /* join lines until end of*/
   if right(li,1)¬= "." then iterate k     /* COBOL sentence         */
   li = strip(li,,".")                     /* now, discard dot       */
   picword=""                              /* reset flags            */
   comp   = 9
   redef  = 0
   occurs = 0
   sign   = 0
   do x = 1 to words(li)                   /* analyse each word      */
      wor = word(li,x)                     /* of the line            */
      select
        when x = 1 then nivel = wor
        when wor = "OCCURS" then do
          occurs = word(li,x+1)
          if occurs = "DEPENDING" then signal erro_depending
        end
        when wor = "REDEFINES"        then redef = 1
        when wor = "SIGN"             then sign=sign+1
        when wor = "SEPARATE"         then sign=sign+1
        when left(wor,3) = "PIC"      then picword = word(li,x+1)
        when wor = "BINARY"           then comp = 0
        when wor = "COMP"             then comp = 0
        when wor = "COMPUTATIONAL"    then comp = 0
```

```
         when wor = "COMP-4"           then comp = Ø
         when wor = "COMPUTATIONAL-4"  then comp = Ø
         when wor = "COMP-3"           then comp = 3
         when wor = "COMPUTATIONAL-3"  then comp = 3
         when wor = "PACKED-DECIMAL"   then comp = 3
         when wor = "COMP-2"           then picword="9(8)"
         when wor = "COMPUTATIONAL-2"  then picword="9(8)"
         when wor = "COMP-1"           then picword="9(4)"
         when wor = "COMPUTATIONAL-1"  then picword="9(4)"
         when wor = "INDEX"            then picword="9(4)"
         when wor = "POINTER"          then picword="9(4)"
         when left(wor,4) = "SYNC"     then signal erro_sync
         otherwise nop
      end
   end x
   if nsign = Ø then do
      if sign = 2 then nsign = nivel
   end
   else do
      if nsign ¬< nivel then nsign = Ø
   end
   if nred = Ø then do
      if redef= 1 then nred = nivel
   end
   else do
      if nred ¬< nivel then do
         if redef = Ø then nred = Ø
         else nred = nivel
      end
   end
   if ok > Ø then do f = 1 to ok
      if nivel¬> nivoc.f then do
         if occurs > Ø then do
            factor.f = occurs
            occurs = Ø
         end
         else ok = ok - 1
      end
   end
   if occurs > Ø then do
      ok = ok + 1
      factor.ok = occurs
      nivoc.ok  = nivel
   end
   if nred = Ø & picword ¬="" then call picture
li = ""
end k
say "==>>>> TOTAL from line "inicio" to line "fim ":"  total
":" start_line                          /* reset current line     */
```

```
exit
/*********************************************************************/
picture:
 pi = strip(picword,,".")
 l  = length(pi)
 le = l
 do i = 1 to le
    s = substr(pi,i,1)
    if i=1 & nsign>Ø & s="S" &,          /* numeric display with S */
       comp=9 then l = l + 1             /* and sign active, l+1   */
    if s="V"|s="S"|s="P" then l=l-1      /* no-length characters  */
    if s="(" then pa = i+1               /* pa=first char inside() */
    if s=")" then do
       ab = i - pa                       /*ab=num.of chars inside()*/
       vab = substr(pi,pa,ab)            /* vab=value inside ()    */
       l = l + vab - ab - 3              /* l=final pic length     */
       end
 end
 if comp = 3 then  l = l%2 + 1           /* USAGE IS clause, if any*/
 if comp = 2 then  l = 8
 if comp = 1 then  l = 4
 if comp = Ø then do
    if l < 5          then l = 2
    if l > 4 & l < 1Ø then l = 4
    if l > 9          then l = 8
 end
 if ok > Ø then do f = 1 to ok
    l = l * factor.f
 end
 total = total + l
return
/*********************************************************************/
erro_sync:
 say " Clause SYNCHRONIZED is not supported"
 exit
erro_depending:
 say " Clause OCCURS DEPENDING ON is not supported"
 exit
```

Luis Paulo Figueiredo Sousa Ribeiro
Systems Programmer
Edinfor (Portugal)

Dynamic menus system for CMS – part 2

This month we continue the code for the on-line administration utilities that go with the dynamic menus system for CMS.

TAFUME.REP

```
*
*     DYNAMIC MENUS SYSTEM MESSAGE REPOSITORY
*
* AFTER EVERY CHANGE IN THIS FILE, YOU MUST :
*
* 'GENMSG TAFUME REPOS A TAF'
* 'SET LANGUAGE (ADD TAF USER'
& 3
* DYNAMIC MENUS MESSAGES
00000101I FUNCTION SUCCESSFULLY COMPLETED
00010101E &1 KEY IS NOT OPERATIONAL
00020101E CANNOT SCROLL TO THE RIGHT
00030101E CANNOT SCROLL TO THE LEFT
00040101E ENTER AN OPTION IN ANY OF THE MENU BOXES
00050101E OPTION &2 IS INVALID IN &1
00060101E THIS OPTION IS NOT OPERATIONAL
00070101E &1 FUNCTION &3 FAILED WITH RETURN CODE &2
00080101E UNKNOWN TYPE IN OPTION &1
00090101E MAKEXEC FAILED TO GENERATE &1 .RC= &2
00100101I TOP OF LIST
00110101I BOTTOM OF LIST
* LOGO / IDENTIFICATION MESSAGES
00120101E MENU NAME WAS NOT SUPPLIED
00130101E MENU PASSWORD WAS NOT SUPPLIED
00140101E NEW PASSWORD IS SAME AS CURRENT
00150101E WHAT ? SERVER RESPONSE IS INVALID
00160101E REQUESTED MENU (&1) UNKNOWN
00170101E SIGNON PASSWORD IS INVALID
00180101E ADD USER PASSWORD IS INVALID
00190101I PASSWORD SUCCESSFULLY CHANGED
00200101I MENU &1 SUCCESSFULLY ADDED
00210101E MENU &1 IS NOT REGISTERED
* ONL ADMINISTRATION - HELP
00400101E ENTER A VALID OPTION OR PRESS A PF KEY.
* ONL ADMINISTRATION - AUTHORIZATIONS
00500101E USER NAME/MENU NAME IS NULL
00510101E THERE IS NO AUTHORIZATION
00520101I AUTHORIZATION FOR &1 / &2 REMOVED
00530101I AUTHORIZATION FOR &1 / &2 UPDATED
```

```
00540101I AUTHORIZATION FOR &1 / &2 ADDED
00550101E MORE THAN 11 LINES FOR THIS MENU
00560101W MENU &1 IS DEFINED, BUT IS EMPTY.
* ONL ADMINISTRATION - MENUS
00600101E &1 COMMAND UNRECOGNIZED
00610101E UPDATE FLAG WAS NOT SET TO "Y"
00620101I LINE &2 HAS BEEN DELETED FROM MENU &1
00630101I LINE SUCCESSFULLY ADDED TO MENU &1
00640101I LINE &2 HAS BEEN CHANGED IN MENU &1
00650101E MENU &1 HAVE MAX NUMBER OF LINES (18)
00660101E INVALID MENU LINE TYPE : &1
00670101E &1 DOESN'T EXIST. PRESS PF09 TO DEFINE.
00680101E MENU DESCRIPTION MUST BE SUPPLIED.
00690101I MENU &1 SUCCESSFULLY ADDED .
00700101W WARNING ! MENU &1 ALREADY EXIST.
00710101E MENU NAME MUST BE SUPPLIED.
00720101I MENU &1 SUCCESSFULLY DELETED.
* ONL ADMINISTRATION - PROCEDURES
00800101E PROCEDURE NAME IS NULL.
00810101E PROCEDURE &1 DOES NOT EXIST.
00820101E EXECUTABLE PROGRAM NAME NULL.
00830101E MINIDISK ADDRESS IS NULL.
00840101E LINK MODE IS INVALID.
00850101E ACCESS MODE MUST BE SUPLIED.
00860101I PROCEDURE &1 SUCCESSFULLY UPDATED.
00870101I PROCEDURE &1 SUCCESSFULLY DELETED.
```

VMU.SRL

```
A:OAMENU.EXC   TEXT~OAMENU EXEC A
A:HMENU.EXC   TEXT~HMENU EXEC A
A:HMENU.DAT   TEXT~HMENU DATA A
A:XLINES.EXC   TEXT~XLINES EXEC A
A:ZLINES.EXC   TEXT~ZLINES EXEC A
A:XSTATMS.EXC   TEXT~XSTATMS EXEC A
A:XAUTH.EXC   TEXT~XAUTH EXEC A
A:XPASW.EXC   TEXT~XPASW EXEC A
A:XPASW.REX   TEXT~XPASW REXX A
A:TAFUME.REP   TEXT~TAFUME REPOS A
A:XLINES.MEN   TEXT~MENU XLINES A
A:XAUTH.MEN   TEXT~MENU XAUTH A
A:XSTATMS.MEN   TEXT~MENU XSTATMS A
A:XPASW.LOG   TEXT~LOG1 XPASW A
A:READ.ME     TEXT~READ ME C
```

XAUTH EXEC

```
 /* XAUTH EXEC */
```

```
/* AIDKEY EQUATES */
$F1='PF01';$F2='PF02';$F3='PF03';$F4='PF04';$F5='PF05';$F6='PF06'
$F7='PF07';$F8='PF08';$F9='PF09';$7A='PF10';$7B='PF11';$7C='PF12'
$C1='PF13';$C2='PF14';$C3='PF15';$C4='PF16';$C5='PF17';$C6='PF18'
$C7='PF19';$C8='PF20';$C9='PF21';$4A='PF22';$4B='PF23';$4C='PF24'
$01='PA1';$6E='PA2';$7D='ENTER';$6D='CLEAR'
'SET LANGUAGE (ADD TAF USER'
'VMFCLEAR'
/*ERROR_MESSAGE=COPIES(' ',50)*/
MESSAGE.1=COPIES(' ',50)
RESET_ALL:
DROP_USER=-1;DROP_MENU=-1
UID='        '
MENU='         '
AUTH.='';MEN_LINE.=''
UPDYN='N'
ATTR.='C031410042F4'X

'PIPE < MENU XAUTH A | DROP 4 | SPECS 1.8 1 | SORT UNIQUE | STEM USERS.'
'PIPE < MENU XLINES A | DROP 4 | SPECS 1.8 1 | NLOCATE /XLINEHLP/ | SORT
UNIQUE | STEM MENUS.',

LOAD_ARRAYS:
IF UID='' THEN UID='        '
IF MENU='' THEN MENU='         '
'PIPE (ENDCHAR ?) < MENU XAUTH A',
'| DROP 4',
'| LOCATE (1.8) /'UID'/',
'| LOCATE (10.8) /'MENU'/',
'| A: FANOUT ',
'| SPECS 19-* 1',
'| SPLIT AT /!/',
'| STEM AUTH.',
'?A: | VAR AUTHLINE'

'PIPE (ENDCHAR ?) < MENU XLINES A',
'| LOCATE (1.8) /'MENU'/',
'| A: FANOUT',
'| DROP FIRST',
'| DROP LAST',
'| SPECS 10.28 1 ',
'| STEM MEN_LINE.',
'?A: | TAKE FIRST ',
'| SPECS 20.17 1 ',
'| VAR MENDESCR'
IF MENDESCR='MENDESCR' THEN MENDESCR=''
IF MEN_LINE.0=0 & STRIP(MENU)¬='' THEN DO
'XMITMSG 56 MENU (APPLID TAF CALLER XAU NOCOMP VAR'
END
DO I=1 TO MEN_LINE.0
```

```
  IF AUTH.I='X' THEN ATTR.I='C00141F242F4'X
                ELSE ATTR.I='C00141F442F4'X
END
 XAUTH :
 SCREEN='8003'X||,
'1100002903C020410042F3'X||COPIES('=',77)||'11004E1DF0'X||,
'11004F2903C020410042F3'X||'|'||'1100511DF0'X||,
'1100632903C02041F242F5'X||' DYNAMIC MENUS ADMINISTRATION UTILITIES
'||'11008C1DF0'X||,
'11009D2903C020410042F3'X||'|'||'11009F1DF0'X||,
'11009F2903C020410042F3'X||'|'||COPIES('=',77)||'|'||'1100EF1DF0'X||,
'1100EF2903C020410042F3'X||'|'||'1100F11DF0'X||,
'1101062903C02041F242F5'X||' AUTHORIZATIONS MANAGEMENT PANEL
'||'1101281DF0'X||,
'11013D2903C020410042F3'X||'|'||'11013F1DF0'X||,
'11013F2903C020410042F3'X||'|'||COPIES('=',77)||'|'||'11018F1DF0'X||,
'11018F2903C020410042F3'X||'|'||'1101911DF0'X||,
'1101942903C02041F242F7'X||' USER NAME : '||'1101A21DF0'X||,
'1101A52903C00141F442F7'X||UID||'1101AE1DF0'X||,
'1101B22903C02041F242F7'X||' MENU NAME : '||'1101C01DF0'X||,
'1101C32903C00141F442F7'X||MENU||'1101CC1DF0'X||,
'1101DD2903C020410042F3'X||'|'||'1101DF1DF0'X||,
'1101DF2903C020410042F3'X||'|'||,
'1101F52903C02041F242F6'X||'PF02/03'||'1101FD1DF0'X||,
'1102132903C02041F242F6'X||'PF10/11'||'11021B1DF0'X||,
'11022D2903C020410042F3'X||'|'||'11022F1DF0'X||,
'11022F2903C020410042F3'X||'|'||COPIES('_',77)||'|'||'11027F1DF0'X||,
'11027F2903C020410042F3'X||'|'||'1102811DF0'X||,
'1102842903C02041F242F7'X||' MENU DESCRIPTION : '||'1102991DF0'X||,
'11029C2903C02041F442F7'X||MENDESCR||'1102AF1DF0'X||,
'1102CD2903C020410042F3'X||'|'||'1102CF1DF0'X||,
'1102CF2903C020410042F3'X||'|'||'1102D11DF0'X||,
'11031D2903C020410042F3'X||'|'||'11031F1DF0'X||,
'11031F2903C020410042F3'X||'|'||'1103211DF0'X||,
'1103242903'X||ATTR.1 ||AUTH.1 ||'1103261DF0'X||,
'1103292903C020410042F4'X||MEN_LINE.1 ||'1103451DF0'X||,
'1103462903'X||ATTR.2 ||AUTH.2 ||'1103481DF0'X||,
'11034B2903C020410042F4'X||MEN_LINE.2 ||'11036C1DF0'X||,
'11036D2903C020410042F3'X||'|'||'11036F1DF0'X||,
'11036F2903C020410042F3'X||'|'||'1103711DF0'X||,
'1103742903'X||ATTR.3 ||AUTH.3 ||'1103761DF0'X||,
'1103792903C020410042F4'X||MEN_LINE.3 ||'1103951DF0'X||,
'1103962903'X||ATTR.4 ||AUTH.4 ||'1103981DF0'X||,
'11039B2903C020410042F4'X||MEN_LINE.4 ||'1103BC1DF0'X||,
'1103BD2903C020410042F3'X||'|'||'1103BF1DF0'X||,
'1103BF2903C020410042F3'X||'|'||'1103C11DF0'X||,
'1103C42903'X||ATTR.5 ||AUTH.5 ||'1103C61DF0'X||,
'1103C92903C020410042F4'X||MEN_LINE.5 ||'1103E51DF0'X||,
'1103E62903'X||ATTR.6 ||AUTH.6 ||'1103E81DF0'X||,
'1103EB2903C020410042F4'X||MEN_LINE.6 ||'11040C1DF0'X||,
```

```
'11040D2903C020410042F3'X||'|'||'11040F1DF0'X||,
'11040F2903C020410042F3'X||'|'||'1104111DF0'X||,
'1104142903'X||ATTR.7 ||AUTH.7 ||'1104161DF0'X||,
'1104192903C020410042F4'X||MEN_LINE.7 ||'1104351DF0'X||,
'1104362903'X||ATTR.8 ||AUTH.8 ||'1104381DF0'X||,
'11043B2903C020410042F4'X||MEN_LINE.8 ||'11045C1DF0'X||,
'11045D2903C020410042F3'X||'|'||'11045F1DF0'X||,
'11045F2903C020410042F3'X||'|'||'1104611DF0'X||,
'1104642903'X||ATTR.9 ||AUTH.9 ||'1104661DF0'X||,
'1104692903C020410042F4'X||MEN_LINE.9 ||'1104851DF0'X||,
'1104862903'X||ATTR.10||AUTH.10||'1104881DF0'X||,
'11048B2903C020410042F4'X||MEN_LINE.10||'1104AC1DF0'X||,
'1104AD2903C020410042F3'X||'|'||'1104AF1DF0'X||,
'1104AF2903C020410042F3'X||'|'||'1104B11DF0'X||,
'1104B42903'X||ATTR.11||AUTH.11||'1104B61DF0'X||,
'1104B92903C020410042F4'X||MEN_LINE.11||'1104D51DF0'X||,
'1104D62903'X||ATTR.12||AUTH.12||'1104D81DF0'X||,
'1104DB2903C020410042F4'X||MEN_LINE.12||'1104FC1DF0'X||,
'1104FD2903C020410042F3'X||'|'||'1104FF1DF0'X||,
'1104FF2903C020410042F3'X||'|'||'1105011DF0'X||,
'1105042903'X||ATTR.13||AUTH.13||'1105061DF0'X||,
'1105092903C020410042F4'X||MEN_LINE.13||'1105251DF0'X||,
'1105262903'X||ATTR.14||AUTH.14||'1105281DF0'X||,
'11052B2903C020410042F4'X||MEN_LINE.14||'11054C1DF0'X||,
'11054D2903C020410042F3'X||'|'||'11054F1DF0'X||,
'11054F2903C020410042F3'X||'|'||'1105511DF0'X||,
'1105542903'X||ATTR.15||AUTH.15||'1105561DF0'X||,
'1105592903C020410042F4'X||MEN_LINE.15||'1105751DF0'X||,
'1105762903'X||ATTR.16||AUTH.16||'1105781DF0'X||,
'11057B2903C020410042F4'X||MEN_LINE.16||'11059C1DF0'X||,
'11059D2903C020410042F3'X||'|'||'11059F1DF0'X||,
'11059F2903C020410042F3'X||'|'||'1105A11DF0'X||,
'1105A42903'X||ATTR.17||AUTH.17||'1105A61DF0'X||,
'1105A92903C020410042F4'X||MEN_LINE.17||'1105C51DF0'X||,
'1105C62903'X||ATTR.18||AUTH.18||'1105C81DF0'X||,
'1105CB2903C020410042F4'X||MEN_LINE.18||'1105EC1DF0'X||,
'1105ED2903C020410042F3'X||'|'||'1105EF1DF0'X||,
'1105EF2903C020410042F3'X||'|'||'1105F11DF0'X||,
'11063D2903C020410042F3'X||'|'||'11063F1DF0'X||,
'11063F2903C020410042F3'X||'|'||'1106411DF0'X||,
'11068D2903C020410042F3'X||'|'||'11068F1DF0'X||,
'11068F2903C020410042F3'X||'|'||COPIES('_',77)||'|'||'1106DF1DF0'X||,
'1106DF2903C020410042F3'X||'|'||'1106E11DF0'X||,
'1106E42903C02041F242F6'X||'UPDATE ? (Y/N)'||'1106F31DF0'X||,
'1106F62903C00141F442F6'X||UPDYN||'1106F81DF0'X||,
'1106F92903C03041F142F2'X||MESSAGE.1||'11072C1DF0'X||,
'11072D2903C020410042F3'X||'|'||'11072F1DF0'X||,
'1107302903C020410042F3'X||COPIES('=',77)||'11077E1DF0'X||,
'1101A613'X
'PIPE (ENDCHAR ?) VAR SCREEN | FULLSCREEN ',
```

```
'| SPLIT AT ANYOF /'"11"X'/',
'| A: FANOUT',
'| TAKE FIRST',
'| SPECS /$/ 1 1-1 C2X NEXT',
'| VAR AIDKEY',
'? A:',
'| DROP FIRST',
'| SPECS 3-* 1',
'| STEM SCR_OUT.'
MESSAGE.1=COPIES(' ',5Ø)
SELECT
WHEN VALUE(AIDKEY)='ENTER' THEN DO              /* PROCESS ENTER KEY */
     'PIPE (ENDCHAR ?) STEM SCR_OUT.',
     '| A: FANOUT ',
     '| TAKE FIRST | VAR N_UID',
     '?A: | DROP FIRST | TAKE 1 | VAR N_MENU',
     '?A: | TAKE LAST| VAR UPDYN',
     '?A: | DROP 2 | DROP LAST | STEM N_AUTH.'
       IF N_UID¬=UID | N_MENU¬=MENU THEN DO
        AUTH.='';MEN_LINE.='';ATTR.='CØ314ØØ42F4'X
        MENU=N_MENU;UID=N_UID
        SIGNAL LOAD_ARRAYS
        END /* END NEW UID OR MENU */
       IF UPDYN='Y' | UPDYN='y' THEN DO
        IF STRIP(N_UID)='' | STRIP(N_MENU)='' THEN DO
        'XMITMSG 5Ø (APPLID TAF CALLER XAU NOCOMP VAR'
         UPDYN='N';SIGNAL XAUTH
         END /* END UID/MENU NULL */
        IF N_AUTH.Ø=Ø THEN DO
        'XMITMSG 51 (APPLID TAF CALLER XAU NOCOMP VAR'
         UPDYN='N';SIGNAL XAUTH
         END /* END NO AUTHORIZATION */
        N_AUTHLINE=LEFT(UID,8)||'!'||LEFT(MENU,8)||'!'
             X_COUNT=Ø
             DO J=1 TO 18
             IF STRIP(N_AUTH.J)='X' THEN X_COUNT=X_COUNT+1
                IF X_COUNT>11 THEN DO
               'XMITMSG 55 (APPLID TAF CALLER XAU NOCOMP VAR'
                UPDYN='N';SIGNAL XAUTH
                END
             IF STRIP(N_AUTH.J)='' THEN N_AUTH.J=' '
             N_AUTHLINE=N_AUTHLINE||N_AUTH.J||'!'
             END
             IF X_COUNT=Ø THEN DO
           'PIPE < MENU XAUTH A | NLOCATE /'AUTHLINE'/ | > MENU XAUTH A'
            'XMITMSG 52 UID MENU (APPLID TAF CALLER XAU NOCOMP VAR'
         ERROR_MESSAGE='AUTHORIZATION FOR 'UID'/'MENU' HAS BEEN REMOVED'
             UPDYN='N';SIGNAL LOAD_ARRAYS
             END
             IF SYMBOL('AUTHLINE')='VAR' THEN DO
```

```
            'PIPE < MENU XAUTH A | CHANGE /'AUTHLINE'/'N_AUTHLINE'/',
            '| > MENU XAUTH A'
            'XMITMSG 53 UID MENU (APPLID TAF CALLER XAU NOCOMP VAR'
             UPDYN='N';SIGNAL LOAD_ARRAYS
             END
                                       ELSE DO
    'PIPE < MENU XAUTH A | APPEND LITERAL 'N_AUTHLINE' | > MENU XAUTH A'
            'XMITMSG 54 UID MENU (APPLID TAF CALLER XAU NOCOMP VAR'
             UPDYN='N';SIGNAL LOAD_ARRAYS
             END
       END /* END UPDATE=YES */
       ELSE SIGNAL XAUTH
     END /* END ENTER */
WHEN VALUE(AIDKEY)='PF11' THEN DO   /*SCROLL FORWARD THROUGH MENU LIST*/
     AUTH.='';MEN_LINE.='';ATTR.='C031410042F4'X
     DROP_MENU=DROP_MENU+1
     IF DROP_MENU >= MENUS.0 THEN DROP_MENU=DROP_MENU-MENUS.0
     'PIPE STEM MENUS.',
     '| DROP 'DROP_MENU,
     '| TAKE 1 ',
     '| VAR MENU'
     SIGNAL LOAD_ARRAYS
     END /* END PF11 */
WHEN VALUE(AIDKEY)='PF10' THEN DO  /*SCROLL BACKWARD THROUGH MENU LIST*/
     AUTH.='';MEN_LINE.='';ATTR.='C031410042F4'X
     DROP_MENU=DROP_MENU-1
     IF DROP_MENU < 0 THEN DROP_MENU=MENUS.0-1
     'PIPE STEM MENUS.',
     '| DROP 'DROP_MENU,
     '| TAKE 1 ',
     '| VAR MENU'
     SIGNAL LOAD_ARRAYS
     END /* END PF10 */
WHEN VALUE(AIDKEY)='PF02' THEN DO  /*SCROLL BACKWARD THROUGH USER LIST*/
     AUTH.='';MEN_LINE.='';ATTR.='C031410042F4'X
     DROP_USER=DROP_USER-1
     IF DROP_USER < 0 THEN DROP_USER=USERS.0-1
     'PIPE STEM USERS.',
     '| DROP 'DROP_USER,
     '| TAKE 1 ',
     '| VAR UID'
     SIGNAL LOAD_ARRAYS
     END /* END PF02 */
WHEN VALUE(AIDKEY)='PF03' THEN DO   /*SCROLL FORWARD THROUGH USER LIST*/
     AUTH.='';MEN_LINE.='';ATTR.='C031410042F4'X
     DROP_USER=DROP_USER+1
     IF DROP_USER >= USERS.0 THEN DROP_USER=DROP_USER-USERS.0
     'PIPE STEM USERS.',
     '| DROP 'DROP_USER,
     '| TAKE 1 ',
```

```
     '| VAR UID'
     SIGNAL LOAD_ARRAYS
     END /* END PFØ3 */
WHEN VALUE(AIDKEY)='PF13' THEN DO                                  /* TEST */
     SAY AUTHLINE
     SAY UID N_UID
     SAY MENU N_MENU
     RETURN
     END /* END PF13 TEST */
WHEN VALUE(AIDKEY)='PF24' | VALUE(AIDKEY)='PF12' THEN EXIT
WHEN VALUE(AIDKEY)='CLEAR' | VALUE(AIDKEY)='PA2' THEN SIGNAL RESET_ALL
OTHERWISE DO
     'XMITMSG 1 'VALUE(AIDKEY)' (APPLID TAF CALLER XAU NOCOMP VAR'
     SIGNAL XAUTH
     END /* END OTHERWISE */
END /* END SELECT */
RETURN
```

XAUTH.MEN

```
================================AUTHORIZATIONS=========
|USERID | MENU   |1|2|3|4|5|6|7|8|9|1|1|1|1|1|1|1|1|
|------ | ----   | | | | | | | | |Ø|1|2|3|4|5|6|7|8|
=======================================================
MENUADMN|CMSUTIL |X|X|X|X|X| | | | | | | | | | | | |
MENUADMN|SYSCENT |X|X|X|X|X|X|X|X| | | | | | | | | |
MENUADMN|TLMENU  |X|X|X|X|X|X| | | | | | | | | | | |
MENUADMN|MLTPRT  |X|X|X|X|X|X| | | | | | | | | | | |
MENUADMN|FOCUS-1 |X|X|X|X|X| | | | | | | | | | | | |
MENUADMN|FOCUS-2 |X| |X|X|X|X|X| | | | | | | | | | |
MAINT   |CMSUTIL | | | | | | | | | | | | | | | | |
MAINT   |SYSCENT | |X| |X|X| |X| | | | | | | | | | |
MAINT   |MLTPRT  | | | | | | | | | | | | | | | | |
FOCUSER |FOCUS-1 |X| |X| |X| | | | | | | | | | | | |
FOCUSER |FOCUS-2 | | |X| | |X| | | | | | | | | | | |
INFOCENT|FOCUS-1 |X|X| | |X| | | | | | | | | | | | |
INFOCENT|FOCUS-2 |X| | | | | | | | | | | | | | | | |
INFOCENT|CMSUTIL | | | | | | | | | | | | | | | | |
TESTER  |FOCUS-1 | | |X|X|X| | | | | | | | | | | | |
TEMPUS  |TLMENU  |X|X| |X| | | | | | | | | | | | | |
TESTER  |CMSUTIL |X| |X|X|X| | | | | | | | | | | | |
MAINT   |FOCUS-1 |X| | | | | | | | | | | | | | | | |
MENUADMN|OAMENU  |X|X|X|X|X| | | | | | | | | | | | |
MENUADMN|FOCUS-P |X|X|X|X| | | | | | | | | | | | | |
```

XLINES EXEC

```
 /* XLINES EXEC */
/* AIDKEY EQUATES */
$F1='PFØ1';$F2='PFØ2';$F3='PFØ3';$F4='PFØ4';$F5='PFØ5';$F6='PFØ6'
$F7='PFØ7';$F8='PFØ8';$F9='PFØ9';$7A='PF1Ø';$7B='PF11';$7C='PF12'
$C1='PF13';$C2='PF14';$C3='PF15';$C4='PF16';$C5='PF17';$C6='PF18'
$C7='PF19';$C8='PF2Ø';$C9='PF21';$4A='PF22';$4B='PF23';$4C='PF24'
$Ø1='PA1';$6E='PA2';$7D='ENTER';$6D='CLEAR'
'SET LANGUAGE (ADD TAF USER'
'VMFCLEAR'
MESSAGE.1=COPIES(' ',56)

RESET_ALL:
DROP_MENU=-1
MENU='       '
MENDESCR='                '
MENLEXEC.=''
MENLDESC.=''
MENLTYPE.=''
TOPBOT=1
UPDYN='N'
LATTR.='CØ31410042F4'X
ATTR.='CØ31410042F4'X
'PIPE < MENU XLINES ',
'| DROP 4',
'| SPECS 1.8 1',
'| NLOCATE /XLINEHLP/ ',
'| SORT UNIQUE ',
'| STEM MENUS.'

LOAD_ARRAYS:
IF STRIP(MENU)='' THEN SIGNAL XLINES
'PIPE (ENDCHAR ?) < MENU XLINES ',
'| DROP 4',
'| LOCATE (1.8) /'MENU'/',
'| DROP LAST',
'| A: FANOUT ',
'| TAKE 1 ',
'| SPECS 2Ø.17 1 ',
'| VAR MENDESCR',
'?A:| DROP 'TOPBOT' | SPECS 1Ø.8  1 | STEM MENLEXEC.',
'?A:| DROP 'TOPBOT' | SPECS 2Ø.17 1 | STEM MENLDESC.',
'?A:| DROP 'TOPBOT' | SPECS 39.8  1 | STEM MENLTYPE.',
'?A:| DROP 'TOPBOT' | SPECS 1-*   1 | STEM MENULINE.'
IF MENU='' THEN MENU='        '
IF MENULINE.Ø=Ø & STRIP(MENU)¬='' THEN DO
'XMITMSG 56 MENU (APPLID TAF CALLER XLI NOCOMP VAR'
END
IF MENDESCR='MENDESCR' THEN DO
```

```
 'XMITMSG 67 MENU (APPLID TAF CALLER XLI NOCOMP VAR'
  MENDESCR='        '
END
DO I=1 TO MENLEXEC.Ø
  IF STRIP(MENLEXEC.I)¬='' THEN DO
                           ATTR.I='C00141F442F4'X
                           LATTR.I='C001410042F4'X
                           END
END
 XLINES :
 SCREEN='8003'X||,
'1100002903C020410042F3'X||COPIES('=',77)||'11004E1DF0'X||,
'11004F2903C020410042F3'X||'|'||'1100511DF0'X||,
'1100632903C02041F242F5'X||' DYNAMIC MENUS ADMINISTRATION UTILITIES
'||'11008C1DF0'X||,
'11009D2903C020410042F3'X||'|'||'11009F1DF0'X||,
'11009F2903C020410042F3'X||'|'||COPIES('=',77)||'|'||'1100EF1DF0'X||,
'1100EF2903C020410042F3'X||'|'||'1100F11DF0'X||,
'11010C2903C02041F242F5'X||' MENUS UPDATE UTILITY '||'1101231DF0'X||,
'11013D2903C020410042F3'X||'|'||'11013F1DF0'X||,
'11013F2903C020410042F3'X||'|'||COPIES('=',77)||'|'||'11018F1DF0'X||,
'11018F2903C020410042F3'X||'|'||'1101911DF0'X||,
'1101942903C02041F242F7'X||' MENU NAME : '||'1101A21DF0'X||,
'1101A52903C00141F442F7'X||MENU||'1101AE1DF0'X||,
'1101B02903C02041F242F7'X||' MENU DESCRIPTION : '||'1101C51DF0'X||,
'1101C82903C02041F442F7'X||MENDESCR||'1101DC1DF0'X||,
'1101DD2903C020410042F3'X||'|'||'1101DF1DF0'X||,
'1101DF2903C020410042F3'X||'|'||,
'1101F52903C02041F242F6'X||'PF10/11'||'1101FD1DF0'X||,
'11022D2903C020410042F3'X||'|'||'11022F1DF0'X||,
'11022F2903C020410042F3'X||'|'||COPIES('_',77)||'|'||'11027F1DF0'X||,
'11027F2903C020410042F3'X||'|'||'1102811DF0'X||,
'1102CD2903C020410042F3'X||'|'||'1102CF1DF0'X||,
'1102CF2903C020410042F3'X||'|'||'1102D11DF0'X||,
'1102D42903C02041F242F7'X||'COMMAND'||'1102DC1DF0'X||,
'1102DE2903C02041F242F7'X||'   DESCRIPTION'||'1102F01DF0'X||,
'1102F42903C02041F242F7'X||'  EXEC'||'1102FD1DF0'X||,
'1103012903C02041F242F7'X||'  TYPE'||'11030A1DF0'X||,
'11031D2903C020410042F3'X||'|'||'11031F1DF0'X||,
'11031F2903C020410042F3'X||'|'||'1103211DF0'X||,
'1103262903'X||ATTR.1 ||' '||'1103281DF0'X||,
'11032E2903'X||LATTR.1 ||MENLDESC.1 ||'1103401DF0'X||,
'1103442903'X||LATTR.1 ||MENLEXEC.1 ||'11034D1DF0'X||,
'1103512903'X||LATTR.1 ||MENLTYPE.1 ||'11035A1DF0'X||,
'11036D2903C020410042F3'X||'|'||'11036F1DF0'X||,
'11036F2903C020410042F3'X||'|'||'1103711DF0'X||,
'1103762903'X||ATTR.2 ||' '||'1103781DF0'X||,
'11037E2903'X||LATTR.2 ||MENLDESC.2 ||'1103901DF0'X||,
'1103942903'X||LATTR.2 ||MENLEXEC.2 ||'11039D1DF0'X||,
'1103A12903'X||LATTR.2 ||MENLTYPE.2 ||'1103AA1DF0'X||,
```

```
'1103BD2903C020410042F3'X||'|'||'1103BF1DF0'X||,
'1103BF2903C020410042F3'X||'|'||'1103C11DF0'X||,
'1103C62903'X||ATTR.3 ||' '||'1103C81DF0'X||,
'1103CE2903'X||LATTR.3 ||MENLDESC.3 ||'1103E01DF0'X||,
'1103E42903'X||LATTR.3 ||MENLEXEC.3 ||'1103ED1DF0'X||,
'1103F12903'X||LATTR.3 ||MENLTYPE.3 ||'1103FA1DF0'X||,
'11040D2903C020410042F3'X||'|'||'11040F1DF0'X||,
'11040F2903C020410042F3'X||'|'||'1104111DF0'X||,
'1104162903'X||ATTR.4 ||' '||'1104181DF0'X||,
'11041E2903'X||LATTR.4 ||MENLDESC.4 ||'1104301DF0'X||,
'1104342903'X||LATTR.4 ||MENLEXEC.4 ||'11043D1DF0'X||,
'1104412903'X||LATTR.4 ||MENLTYPE.4 ||'11044A1DF0'X||,
'11045D2903C020410042F3'X||'|'||'11045F1DF0'X||,
'11045F2903C020410042F3'X||'|'||'1104611DF0'X||,
'1104662903'X||ATTR.5 ||' '||'1104681DF0'X||,
'11046E2903'X||LATTR.5 ||MENLDESC.5 ||'1104801DF0'X||,
'1104842903'X||LATTR.5 ||MENLEXEC.5 ||'11048D1DF0'X||,
'1104912903'X||LATTR.5 ||MENLTYPE.5 ||'11049A1DF0'X||,
'1104AD2903C020410042F3'X||'|'||'1104AF1DF0'X||,
'1104AF2903C020410042F3'X||'|'||'1104B11DF0'X||,
'1104B62903'X||ATTR.6 ||' '||'1104B81DF0'X||,
'1104BE2903'X||LATTR.6 ||MENLDESC.6 ||'1104D01DF0'X||,
'1104D42903'X||LATTR.6 ||MENLEXEC.6 ||'1104DD1DF0'X||,
'1104E12903'X||LATTR.6 ||MENLTYPE.6 ||'1104EA1DF0'X||,
'1104FD2903C020410042F3'X||'|'||'1104FF1DF0'X||,
'1104FF2903C020410042F3'X||'|'||'1105011DF0'X||,
'1105062903'X||ATTR.7 ||' '||'1105081DF0'X||,
'11050E2903'X||LATTR.7 ||MENLDESC.7 ||'1105201DF0'X||,
'1105242903'X||LATTR.7 ||MENLEXEC.7 ||'11052D1DF0'X||,
'1105312903'X||LATTR.7 ||MENLTYPE.7 ||'11053A1DF0'X||,
'11054D2903C020410042F3'X||'|'||'11054F1DF0'X||,
'11054F2903C020410042F3'X||'|'||'1105511DF0'X||,
'1105562903'X||ATTR.8 ||' '||'1105581DF0'X||,
'11055E2903'X||LATTR.8 ||MENLDESC.8 ||'1105701DF0'X||,
'1105742903'X||LATTR.8 ||MENLEXEC.8 ||'11057D1DF0'X||,
'1105812903'X||LATTR.8 ||MENLTYPE.8 ||'11058A1DF0'X||,
'11059D2903C020410042F3'X||'|'||'11059F1DF0'X||,
'11059F2903C020410042F3'X||'|'||'1105A11DF0'X||,
'1105A62903'X||ATTR.9 ||' '||'1105A81DF0'X||,
'1105AE2903'X||LATTR.9 ||MENLDESC.9 ||'1105C01DF0'X||,
'1105C42903'X||LATTR.9 ||MENLEXEC.9 ||'1105CD1DF0'X||,
'1105D12903'X||LATTR.9 ||MENLTYPE.9 ||'1105DA1DF0'X||,
'1105ED2903C020410042F3'X||'|'||'1105EF1DF0'X||,
'1105EF2903C020410042F3'X||'|'||'1105F11DF0'X||,
'1105F62903'X||ATTR.10||' '||'1105F81DF0'X||,
'1105FE2903'X||LATTR.10||MENLDESC.10||'1106101DF0'X||,
'1106142903'X||LATTR.10||MENLEXEC.10||'11061D1DF0'X||,
'1106212903'X||LATTR.10||MENLTYPE.10||'11062A1DF0'X||,
'11063D2903C020410042F3'X||'|'||'11063F1DF0'X||,
'11063F2903C020410042F3'X||'|'||'1106411DF0'X||,
```

```
'11068D2903C020410042F3'X||'|'||'11068F1DF0'X||,
'11068F2903C020410042F3'X||'|'||COPIES('_',77)||'|'||'1106DF1DF0'X||,
'1106DF2903C020410042F3'X||'|'||'1106E11DF0'X||,
'1106E42903C02041F242F6'X||'UPDATE?(Y/N)'||'1106F11DF0'X||,
'1106F12903C00141F442F6'X||UPDYN||'1105F31DF0'X||,
'1106F32903C03041F142F2'X||MESSAGE.1||'11072C1DF0'X||,
'11072D2903C020410042F3'X||'|'||'11072F1DF0'X||,
'1107302903C020410042F3'X||COPIES('=',77)||'11077E1DF0'X||,
'1101A613'X
'PIPE (ENDCHAR ?) VAR SCREEN | FULLSCREEN ',
'| SPLIT AT ANYOF /'"11"X'/',
'| A: FANOUT',
'| TAKE FIRST',
'| SPECS /$/ 1 1-1 C2X NEXT',
'| VAR AIDKEY',
'? A:',
'| DROP FIRST',
'| SPECS 3-* 1',
'| STEM SCR_OUT.'
MESSAGE.1=COPIES(' ',50)
SELECT
```

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

Yaakov J Hazan
Technical Support Manager
Ynon Technologies & Computers Ltd (Israel)

Why not share your expertise and earn money at the same time? *VM Update* is looking for REXX EXECs, macros, program code, etc, that experienced VMers have written to make their life, or the lives of their users, easier. We will publish it (after vetting by our expert panel) and send you a cheque when the article is published. Articles can be of any length and can be sent or e-mailed to Trevor Eddolls at any of the addresses shown on page 2. Why not call now for a free copy of our *Notes for contributors*?

Beyond Software is to provide its entire family of solutions for Web enabling legacy applications to Cap Gemini to market throughout Europe. The deal includes Beyond's native Web servers for VM/ESA, the OS/390 and MVS/ESA platforms, as well as its integrated e-mail and calendaring applications for the Office Vision environment and browser technology for 3270 users.

For further information contact:
Beyond Software, 5201 Great American Parkway, Suite 351, Santa Clara, CA 95054, USA.
Tel: (408) 496 1920.
Cap Gemini, Cap Gemini House, 130 Shaftesbury Avenue London, W1V 8HH, UK.
Tel: (01483) 786321.

* * *

IBM has announced Millennium Language Extensions (MLE) for changing COBOL and PL/I applications to handle year 2000 dates. The technology, to be integrated into forthcoming releases of its COBOL and PL/I compilers, will automate date century windowing.

With date century windowing, you can specify a new 100 year span so that the two-digit year value, say 05, is interpreted correctly as 2005. MLE is said to automate much of the code-intensive process for applications that are appropriate for windowing.

The compilers to incorporate the MLE are part of VisualAge 2000. There are MLE for workstation versions of PL/I on OS/2 and Windows NT. There is also VisualAge for COBOL on OS/2 and NT, and PL/I and COBOL for VM, OS/390, MVS, and VSE compilers.

For further information contact your local IBM representative.

* * *

IBM has announced Version 7.6 of its Advanced Communications Function/ Network Control Program for OS/390, VM and VSE. New features include IP Internal Coupling, RIP Version 2 support, additional Frame Relay traffic management enhancements, call connection balancing in duplicate Token-Ring Coupler (TIC) environments, APPN refinements, and more network management mechanisms.

For further information contact your local IBM representative.