145 VM

September 1998

In this issue

update

VM Update

Published by
Xephon
27-35 London Road
Newbury
Berkshire RG14 1JL
England
Telephone: 01635 38030
From USA: 01144 1635 38030
E-mail: xephon@compuserve.com

North American office
Xephon/QNA
1301 West Highway 407, Suite 201-405
Lewisville, TX 75077-2150
USA
Telephone: 940 455 7050

Editorial panel
Articles published in *VM Update* are reviewed by our panel of experts. Members of the panel include John Illingworth (UK), Reinhard Meyer (Germany), Philippe Taymans (Belgium), Romney White (USA), Martin Wicks (UK), and Jim Vincent (USA).

Subscriptions and back-issues
A year's subscription to *VM Update*, comprising twelve monthly issues, costs £175.00 in the UK; $265.00 in the USA and Canada; £181.00 in Europe; £187.00 in Australasia and Japan; and £185.50 elsewhere. In all cases the price includes postage. Individual issues, starting with the January 1990 issue, are available separately to subscribers for £14.50 ($22.50) each including postage.

Editor
Robert Burgess

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.

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.

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*.

Printed in England.

Using ADDPIPEs from REXX procedures

INTRODUCTION

This article shows how we were able to overcome the limitations of an old VM/SP HPO procedure that submitted a job to MVS by using a CMS Pipeline, specifically ADDPIPEs.

The VM/SP HPO procedure took a data file, wrapped some JCL around it, checked that the data did not contain the data delimiter characters, and then sent it to MVS.

The main problem with the old HPO procedure was the length of time it took to generate the data file for transmission to MVS. This limited the size of the file that could be generated without having to wait a substantial time. The generation process also consumed a substantial amount of CPU.

These two limitations led to the larger files being written to tape instead of being sent over the network. The operators would then pick up the tape, walk across the machine room, and load the data into MVS. This operator-intensive procedure was just acceptable provided that the VM and MVS systems stayed together in the same machine room.

The objective of the exercise was to reduce the CPU consumption and the length of time it took to generate the data file ready for transmission to MVS, and stop writing the larger files to tape and send them via the network.

A CMS Pipeline solution was chosen because it could easily be integrated into the current procedures and could be used to convert the data to NETDATA format and check for the data delimiter character in one pass of the data – ie remove the need for the READCARD stage of the old procedure. This would reduce the CPU consumption and length of time it took to generate the data file.

LIMITATIONS OF OLD PROCEDURE

The VM/SP HPO procedure had numerous limitations:

- The length of time for the file to be sent limited the size of files transmitted to MVS.
- A large amount of CPU was consumed.
- Larger files had to be written to tape and then loaded into MVS.

OLD METHODOLOGY

The old methodology required a user to:

- Send data to himself, using SENDFILE to convert the data to NETDATA format.
- Locate the sent file in the reader:
  - QUERY READER * ALL.
  - Loop through the result until the file was found.
- READCARD it to disk to preserve the NETDATA format.
- PUNCH the initial JCL.
- Loop through the file, on disk, checking for the occurrence of the data delimiter characters and then punching the record – this stage consumed a lot of CPU and also took a long time.
- PUNCH the post JCL.

SOLUTION

The solution was to rewrite the application, using CMS Pipelines, to reduce the CPU usage and the elapsed time.

PIPELINE DEFINITION

The basic procedure used to transfer data to MVS is outlined below:

- SPOOL PUN TO RSCS CONT.
- TAG DEV PUN *systemid* JOB.
- PUNCH JCL HEADER.

  - JOB information (account, DSN information, etc).

- PUNCH DATA if the data is fixed 80-byte records, or else use NETDATA for other data file formats:

```
NETDATA SEND fn ft fm TO uid AT nodeid ( NOSPOOL
```

- PUNCH JCL TRAILER.
  - Data delimiter checking and additional post-processing JCL.
- SPOOL PUNCH CLOSE NOCONT.

The three PUNCH steps can be replaced by a CMS Pipeline. The following code shows the basic pipeline to send a job to MVS, but it does not check for the occurrence of the data delimiter characters in the data:

```
'PIPE (end ?) ',
   '  < IPACA JOB1 A' ,
   '| punchit: fanin Ø data jcl2',
   '| punch ØØØD' ,
   '? ',
   '  NETDATA SEND fn ft fm TO uid AT nodeid ( NOSPOOL ,
   '| punchit.data: ',
   '? ',
   '  < IPACA JOB2 A' ,
   '| punchit.jcl2: '
```

HOW CAN THIS BE CHECKED?

One solution is to replace the PUNCH DATA or NETDATA stage with a user stage, which will create the NETDATA file and check for the data delimiter characters in the data.

The following code shows how the CMS Pipeline would look. Extra stages have been added to customize the JCL:

```
'PIPE (end ?) ',
'  < IPACA JOB1 '$fm ,
'| change /%%%%%%%%/'systemid'/' ,
'| change /$$$$$$$$/'delim.1'/' ,
'| PAD 8Ø 4Ø',
'| punchit: fanin Ø data jcl2',
'| punch ØØØD' ,
'? ',
'  HASNET 'touser tonode fn ft fm 'DATADELIMIT' delim.1 delim.2 userp ,
```

```
'| punchit.data: ',
'? ',
'  < IPACA JOB2 '$fm ,
'| change /%%%%%%%%/'systemid'/' ,
'| PAD 80 40',
'| punchit.jcl2: '
```

This is a three pipe pipeline, which combines all the pipes into one stream at a multi-stream pipeline stage:

- Pipe 1 processes the initial JCL.
- Pipe 2 (data) processes the data file.
- Pipe 3 (jcl2) the post JCL requirements.
- 'PAD 80 40' – the JCL records are padded with blanks to ensure that they are 80 bytes in size.
- 'Punchit: fanin 0 data jcl2' – the three pipes of this pipeline are combined together by fanin with the label punchit. The order is expressly specified in the fanin stage command using *streamnum* or *streamid*. The position of the label (punchit:) determines that the '*.OUTPUT.0:' stream from each pipe connects to the multi-stream stage punchit: as '*.INPUT.*num*:' (where *num* is 0, 1, or 2).

Figure 1 illustrates the pipeline above, and shows the flow of data through it. The input and output streams are also shown ('*.INPUT.*num*:' and '*.OUTPUT.*num*:'). Stage g is the user stage HASNET.

USER STAGE DEFINITION (HASNET – STAGE G)

The user stage is used to generate the NETDATA file and check for the occurrence of the data delimiters. This is shown in the following code:

```
STREAMSTATE INPUT 0
STREAMSTATE OUTPUT 0
ADDSTREAM OUTPUT Ctl
'ADDPIPE (end ? name NETBUF)' ,
   '  *.Output.Ctl:',
   '| Buffer ',
   '| Blockit: fanin 0 data tail ',
   '| BLOCK 80 NETDATA ',
```

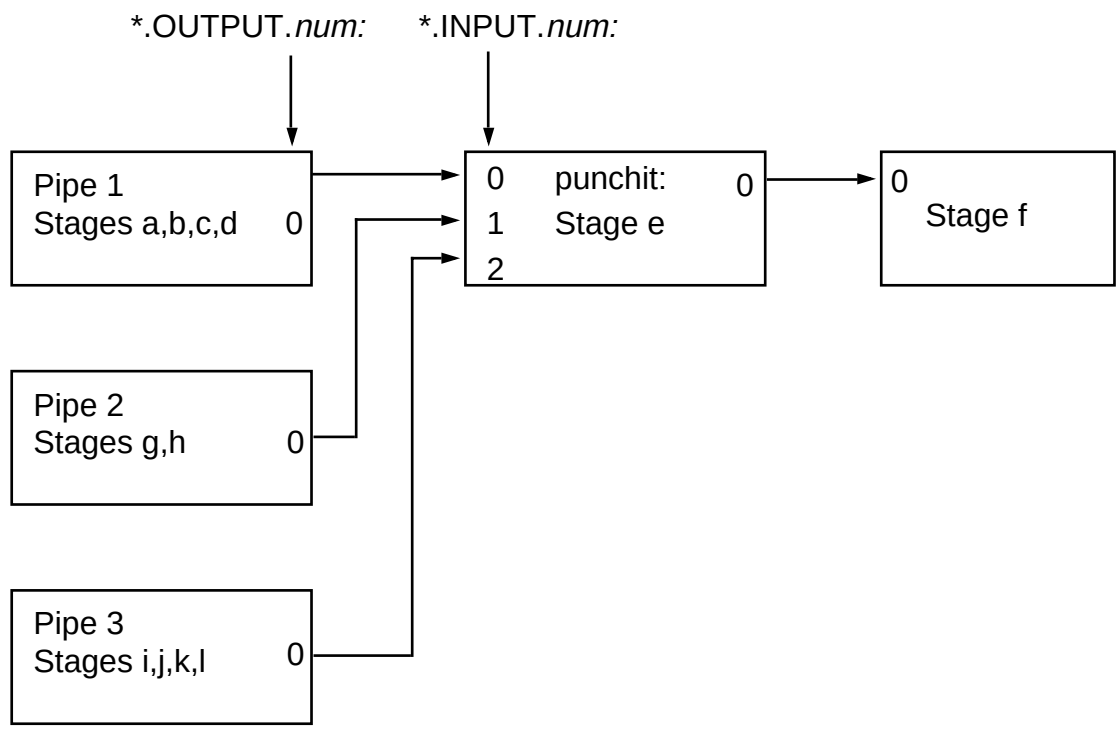

Figure 1: Generic map for the basic pipeline

```
   '| PAD 8Ø ØØ ',
   '| *.Input.Ø: ',
   '? ',
   ' < 'fn ft fm  ',
   '| change // xØØ  ',
   '| Blockit.data: ',
   '? ',
   ' literal \INMRØ6'  ',
   '| Blockit.tail:'
'ADDPIPE (end ? name NETPAD)' ,
   '  *.Output.Ø:',
   '| PAD 8Ø 4Ø  ',
   '| *.Output.Ø:'
STREAMSTATE OUTPUT Ctl
SELECT OUTPUT Ctl
   OUTPUT (INMRØ1, INMRØ2, INMRØ3 records)
SEVER OUTPUT Ctl
SELECT OUTPUT Ø
curr_delim = delim1
curr_delim_count = 1
```

```
Do Forever
   'CALLPIPE (name CHECKDATA) ',
      '  *.Input.Ø: ',
      '| tolabel 'curr_delim ||,
      '| *.output.Ø: '
   'Output 'curr_delim
   'PEEKTO'
   Select
      When rc=Ø  Then Do
         If curr_delim = delim1 Then curr_delim = delim2
                                Else curr_delim = delim1
         "Output //      DD     DATA,DLM='"curr_delim"'"
         curr_delim_count = curr_delim_count +1
         If curr_delim_count > 255 Then Exit(1233)
      End
      When rc=12 Then Leave
      Otherwise Exit(1232)
   End
End.
```

PIPE SUBCOMMANDS

The following section describes the purpose of each of the pipe subcommands.

STREAMSTATE

This user stage is a device driver, ie it interacts with devices or other system resources. STREAMSTATE is used to check that the '*.INPUT.0:' stream is not connected and that the '*.OUTPUT.0:' stream is defined and connected. STREAMSTATE sets the return code as:

0 The stream is defined and connected (waiting).

4 The stream is defined and connected (waiting) (different commit level).

8 The stream is defined and connected (not waiting).

12 The stream is defined but not connected.

-4 The stream is not defined.

ADDSTREAM

Before ADDPIPE can connect to a new stream, it must be defined. ADDSTREAM does this by:

```
ADDSTREAM OUTPUT Ctl
```

ADDPIPE (NETBUF)

Define a new pipeline using the defined pipe stream '*.OUTPUT.Ctl:' The ADDPIPE is added into the existing pipeline and control is returned to the procedure. The added pipeline runs in parallel with the stage that created it. The return code indicates only whether the operands are syntactically correct. (Note: CALLPIPE does not return control until the pipeline has run to completion. Its return code is the return code resulting from the added pipeline.) This ADDPIPE (name NETBUF) generates the NETDATA file which is the same as NETDATA SEND in the original pipe.

The function of the other commands in the pipe are shown below:

- '*.OUTPUT.Ctl:' – takes all the NETDATA header records from this stream. The records are generated later.
- 'BUFFER' – prevents a pipeline stall by buffering all the header records. These header records must be processed first. The BUFFER stage command accumulates all the records from its primary input stream. When BUFFER reaches the end of the file on its input stream, it writes all the records to its primary output stream, if it is connected.
- 'BLOCKIT: fanin 0 data tail' – the three pipes of this pipeline are combined together by fanin with the label BLOCKIT:
  - Pipe 1 processes the NETDATA header records.
  - Pipe 2 (data) processes the data file.
  - Pipe 3 (tail) processes the NETDATA trailer record.

  The order is expressly specified in the fanin stage command using the *streamnum* or *streamid.*
- 'BLOCK 80 NETDATA | PAD 80 00' – blocks the records into

80-byte records in NETDATA format and pads the last record with nulls.

- '*.INPUT.0:' – passes all the records to the stream.
- '< fn ft fm | change // x00' – reads in the data from the file fn ft fm and prefixes each record with X'00' indicating the record is a data record.
- 'BLOCKIT.data:' – passes the records to the label BLOCKIT with a *streamid* of data.
- 'Literal X'E0'INMR06' – the last record in a NETDATA file is the INMR06 record, prefixed with X'E0', where:
  - X'80' indicates the record is the first segment of the original record.
  - X'40' indicates the record is the last segment of the original record.
  - X'20' indicates the record is (part of) a control record.
- 'Blockit.tail:' – passes the records to the label blockit with a *streamid* of tail.

STREAMSTATE OUTPUT Ctl

This shows that the ADDSTREAM and ADDPIPE have defined and connected a new '*.OUTPUT.Ctl:' stream. If the return code is 8, the stream is defined and connected – not waiting.

ADDPIPE (NETPAD)

This defines a new pipeline using the defined pipe stream '*.OUTPUT.0:'. The ADDPIPE is added into the existing pipeline and control is returned to the procedure. This stage ensures that data is padded to 80 bytes with blanks. The only records that should be padded are the data delimiter JCL records.

SELECT and OUTPUT

SELECT an OUTPUT stream '*.OUTPUT.CTL:' and generate the

NETDATA header records. All subsequent OUTPUT pipe subcommands place data into the '*.OUTPUT.0:' stream. These records are consumed by the buffer stage on the NETBUF ADDPIPE. Control is returned to the REXX procedure only when OUTPUT relinquishes control (ie when the record is consumed by the buffer). The REXX procedure is able to generate all the required NETDATA header records. For further guidance, refer to Appendix E, *CMS Application Development Reference for Assembler* – there is a sample procedure of John Hartmann's called INMR123 REXX on MAINT 193, showing how to generate the NETDATA control records; however, this is not Year 2000 compliant! *[Editor's note: a Year-2000 compliant version of this procedure is now available on the pipelines homepage and can be downloaded at http:pucc.princeton.edu/~pipeline/inmr123.rexx.]*

```
SELECT OUTPUT CTL
OUTPUT record
```

NETDATA header control records are INMR01, INMR02, and INMR03.

SEVER OUTPUT Ctl

After an ADDPIPE subcommand runs, the restoration of the stage's original connection is not automatic. A SEVER subcommand must be issued to restore the connection. 'SEVER *.OUTPUT.Ctl:' stream after NETDATA header records have been generated. The buffered records are only available to other stages when the buffer '*.INPUT.0:' connection is severed by the REXX procedure on the SEVER command.

SELECT OUTPUT 0

Select '*.OUTPUT.0:' as the primary output stream. All subsequent OUTPUT pipe subcommands place data into the '*.OUTPUT.0:' stream.

CALLPIPE (CHECKDATA)

Finally, run the CMS Pipeline and place all the records into the stream

‘*.OUTPUT.0:’ for the next stage following the user stage in the original pipeline.

The CALLPIPE (CHECKDATA) processes records until the TOLABEL filter is honoured or there are no more records to process.

TOLABEL copies all the records that do not begin with the specified target to its primary output stream. Because the secondary stream is not connected, the TOLABEL stage stops before reading the record beginning with the specified target string.

Control is then returned to the REXX procedure. PEEKTO is used to inspect the next record in the input stream, without consuming the record.

PEEKTO subcommand returns a return code value of 12 when there are no more records to process, or 0 when there are more records to process.

If PEEKTO returns with a return code value of 0, the data delimiter value has been found in the first two columns of a record. The data delimiter is changed to the other data delimiter value and two new records are added to the ‘*.OUTPUT.0:’ stream by the OUTPUT command. The procedure loops to call CALLPIPE (name CHECKDATA) again (this is regarded as a new CALLPIPE pipe) and continues from the previous PEEKTO record in the ‘*.INPUT.0:’ stream. MVS JCL will allow the data delimiter characters to change a maximum of 255 times and this is checked for in the code.

Figure 2 is a generic representation of the pipeline and shows how the data flows through the pipeline. The input and output streams are for each stage. (‘*.INPUT.*num*:’ and ‘*.OUTPUT.*num*:’).

Note that the output stream from stage g is not connected throughout the user stage.

DATA FLOW THROUGH THE PIPELINE

Various trace points were inserted into the pipeline at the points shown in Figure 3. In the NETBUF ADDPIPE, there are three trace points:

- Before the buffer of the NETDATA header records (NETBUF0).

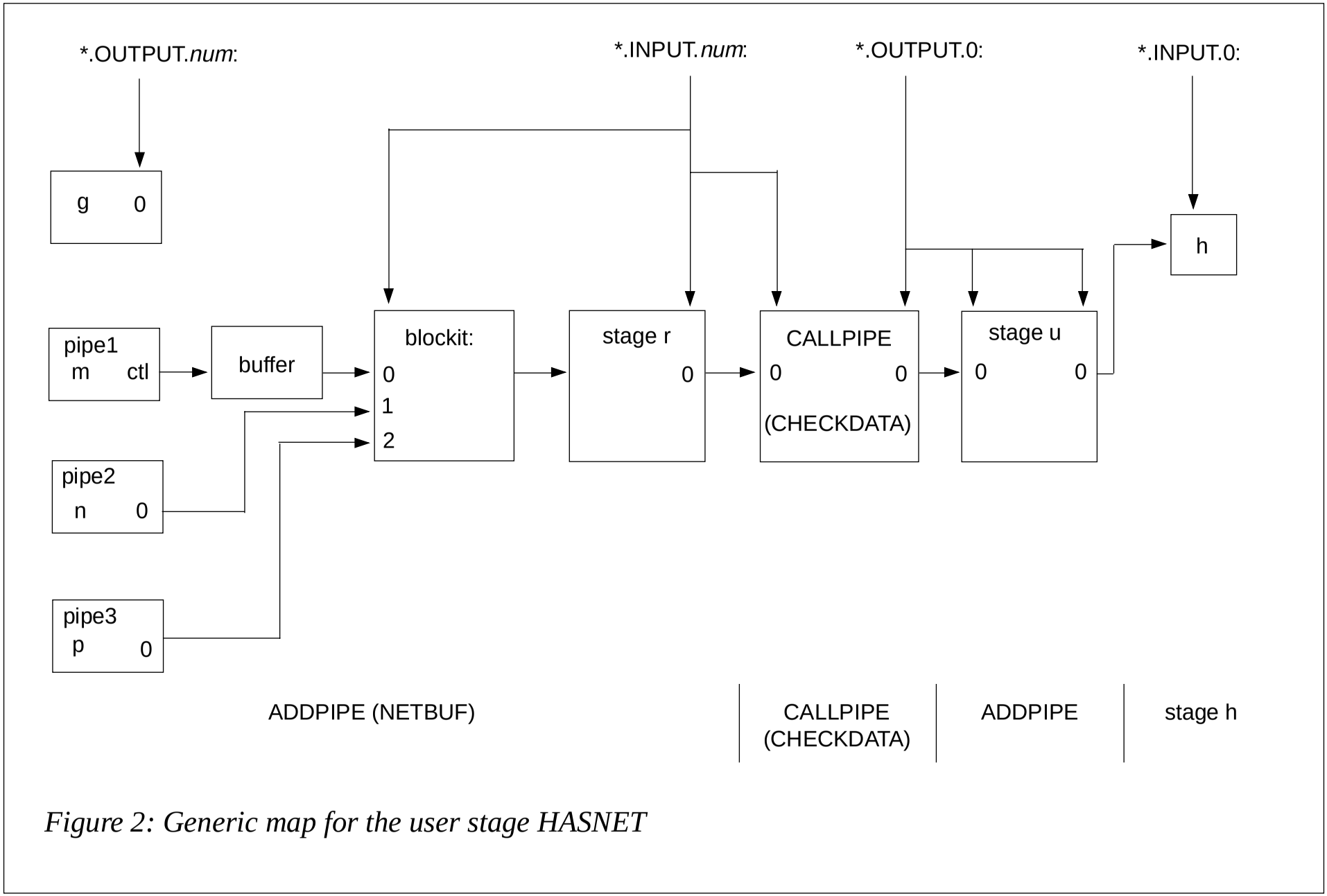

Figure 2: Generic map for the user stage HASNET

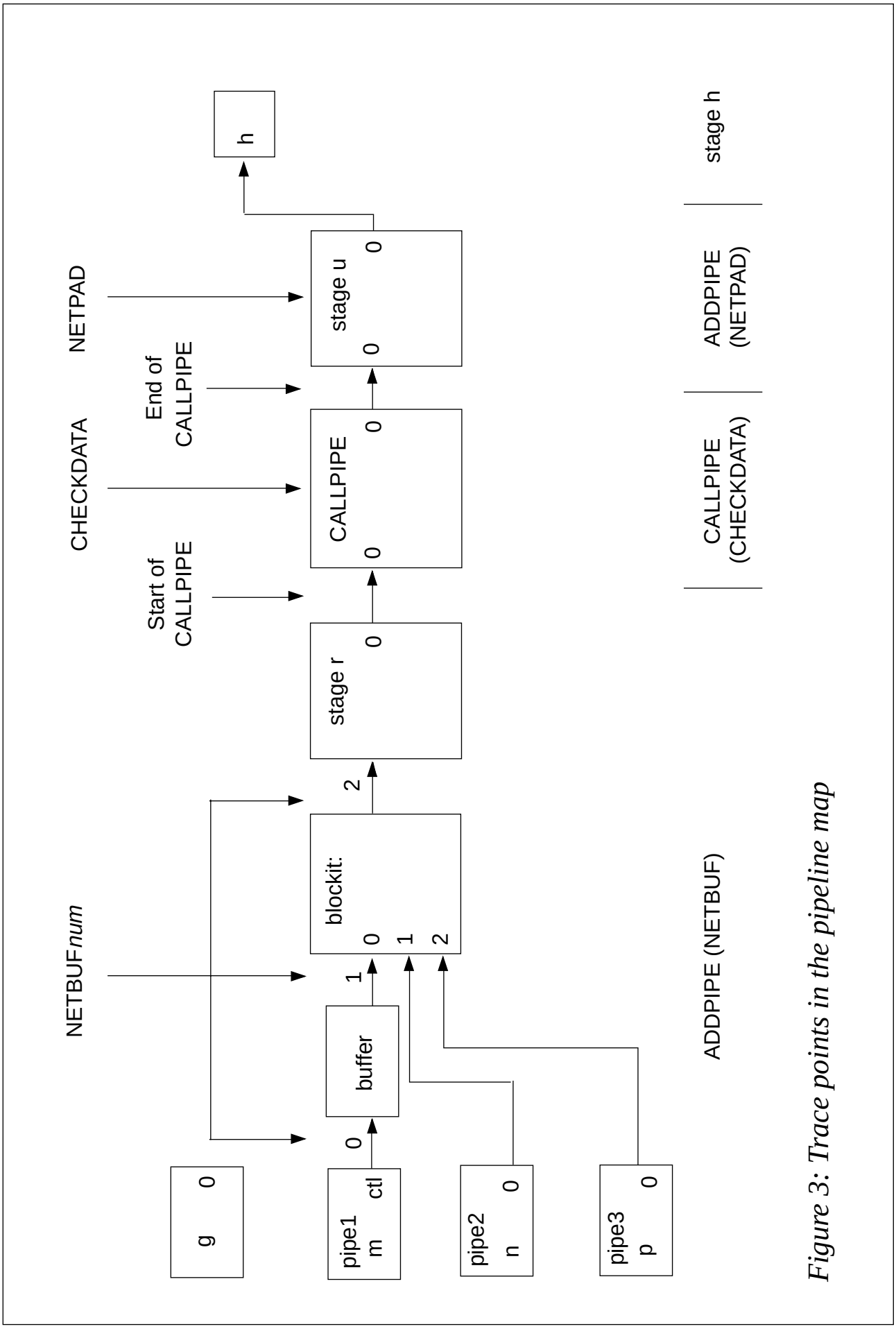

Figure 3: Trace points in the pipeline map

- Before the three pipes are combined into one stream (NETBUF1).
- After the data is combined into one stream and the data has been blocked in NETDATA format (NETBUF2).

There are trace points before, during, and after the CHECKDATA CALLPIPE:

- Starting CALLPIPE *num*.
- CHECKDATA.
- Ending CALLPIPE *num*.

There is a trace point during the NETPAD ADDPIPE (NETPAD).

```
Selecting *.OUTPUT.Ctl
NETBUFØ stage 2:\INMRØ1….AEBOBVM..XZR74D….AEBOBVM…XZR74D…..(158)
NETBUFØ stage 2:\INMRØ2….INMCOPY…......ACCOUNT…..(126)
NETBUFØ stage 2:\INMRØ3…..............(44)
Severing  *.OUTPUT.Ctl
NETBUF1 stage 4:\INMRØ1….AEBOBVM..XZR74D…AEBOBVM…XZR74D…
NETBUF2 stage 8:¤\INMRØ1…AEBOBVM..XZR74D...AEBOBVM…XZR74D….
Selecting *.OUTPUT.Ø
 Starting CALLPIPE call 1
NETPAD stage 2:¤\INMRØ1….AEBOBVM...XZR74D…AEBOBVM…XZR74D…..
checkdata stage 2:¤\INMRØ1….AEBOBVM..XZR74D.…AEBOBVM…XZR74D…..
NETBUF1 stage 4:\INMRØ2…..IMCOPY……ACCOUNT….
NETBUF2 stage 8:155225000000….Contact TS EMEA UKOSG if you can not read this file"
checkdata stage 2:155225000000….Contact TS EMEA UKOSG if you can not read this file"
NETPAD stage 2:155225000000….Contact TS EMEA UKOSG if you can not read this file"
NETBUF2 stage 8:\INMRØ2….INMCOPY…….ACCOUNT….
checkdata stage 2:\INMRØ2….INMCOPY…….ACCOUNT….
NETPAD stage 2:\INMRØ2….INMCOPY……ACCOUNT……
NETBUF1 stage 4:\INMRØ3…….
NETBUF2 stage 8:EBOBVM…..199801201345220000000…\INMRØ3……
checkdata stage 2:EBOBVM…199801201345220000000…..\INMRØ3……
NETPAD stage 2:EBOBVM….199801201345220000000…..\INMRØ3……
NETBUF2 stage 8:….ACCOUNT 99999999¢{ HAS      FICHELOG
checkdata stage 2:…..ACCOUNT 99999999¢{ HAS      FICHELOG
NETPAD stage 2:…..ACCOUNT 99999999¢{ HAS      FICHELOG
NETBUF2 stage 8:                        {ACCOUNT DISASTER¢{ DRA
checkdata stage 2:                        {ACCOUNT DISASTER¢{ DRA
NETPAD stage 2:                        {ACCOUNT DISASTER¢{ DRA
..
```

Figure 4: Initial record flow through the pipeline

Figure 4 shows the initial record flow through the pipeline. This shows the NETDATA header records and the first few data records.

The NETDATA header records are buffered by the buffer until the '*.OUTPUT.Ctl:' is severed by the 'SEVER OUTPUT Ctl' pipe subcommand. The header records now become available to the next stage and are formatted into NETDATA records.

BLOCK NETDATA 80 will generate 80 byte NETDATA format records. For example:

- INMR01 is 158 bytes in length.
- INMR02 is 126 bytes in length.
- INMR03 is 44 bytes in length.

The length byte contains the length of control or data record, plus one for the length byte itself.

The NETDATA record numbers and their contents are as follows:

- 1 is the length byte plus the first 79 bytes on the INMR01 record.
- 2 is the last 79 bytes of the INMR01 record plus the length byte of the INMR02 record (plus one).
- 3 is the first 80 bytes of the INMR02 record.
- 4 is the last 46 bytes of the INMR02 record plus the length byte of the INMR03 record (plus one) plus the first 33 bytes of the INMR03 record.
- 5 is the last 11 bytes of the INMR03 record plus the length byte of the first data record plus the first 68 bytes of the first data record.

Figure 5 shows the flow of records when the data delimiter characters appear in the data.

Records flow through the pipeline:

NETBUF2 -> CHECKDATA -> NETPAD -> stage h ...

until the TOLABEL stage command detects the data delimiter


```
NETBUF2 stage 8: BX028   ¢{ BX029    BX030    BX031    BX032    BX033    BX034    BX035    BX036
checkdata stage 2: BX028   ¢{ BX029    BX030    BX031    BX032    BX033    BX034    BX035    BX036
NETPAD stage 2: BX028   ¢{ BX029    BX030    BX031    BX032    BX033    BX034    BX035    BX036
NETBUF2 stage 8:   ¢{ BX037    BX038    BX039    BX040    BX041    BX042    BX043    BX044   ¢{
checkdata stage 2:   ¢{ BX037    BX038    BX039    BX040    BX041    BX042    BX043    BX044   ¢{
NETPAD stage 2:   ¢{ BX037    BX038    BX039    BX040    BX041    BX042    BX043    BX044   ¢{
NETBUF2 stage 8:BX045    BX046    BX047    BX048    BX049    BX050    BX051    BX052   ¢{ BX053
checkdata stage 2:BX045    BX046    BX047    BX048    BX049    BX050    BX051    BX052   ¢{ BX053
Ending CALLPIPE call 1
NETPAD stage 2:BX
NETPAD stage 2://      DD     DATA,DLM='DE'
 Starting CALLPIPE call 2
NETPAD stage 2:BX045    BX046    BX047    BX048    BX049    BX050    BX051    BX052   ¢{ BX053
NETBUF2 stage 8:   BX054    BX055    BX056    BX057    BX058    BX059    BX060   ¢{ BX061    BX0
checkdata stage 2:   BX054    BX055    BX056    BX057    BX058    BX059    BX060   ¢{ BX061    BX0
NETPAD stage 2:   BX054    BX055    BX056    BX057    BX058    BX059    BX060   ¢{ BX061    BX0
NETBUF2 stage 8:62    BX063    BX064    BX065    BX066    BX067    BX068   ¢{ BX069    BX070
checkdata stage 2:62    BX063    BX064    BX065    BX066    BX067    BX068   ¢{ BX069    BX070
NETPAD stage 2:62    BX063    BX064    BX065    BX066    BX067    BX068   ¢{ BX069    BX070
..
```

Figure 5: Record flow with data delimiter characters in the data

characters in the data. The CALLPIPE (CHECKDATA) terminates and the current data delimiter is OUTPUT to '*.OUTPUT.0:' and the new record flows through the pipeline.

PEEKTO tests the status of '*.INPUT.0:'. The return code should be 0 (record(s) to process). A new DD card is OUTPUT to '*.OUTPUT.0:' and it flows through the pipeline, being padded to 80 bytes with blanks at the NETPAD stage.

CALLPIPE (CHECKDATA) is called again and the data records flow through the pipeline, starting with the record that terminated the CALLPIPE (CHECKDATA).

Data flows through the pipeline until there are no more records to be processed by the TOLABEL stage, as follows:

```
NETBUF2 stage 8:                          {ACCOUNT CSWB¢{ WINOUT
checkdata stage 2:                        {ACCOUNT CSWB¢{ WINOUT
NETPAD stage 2:                           {ACCOUNT CSWB¢{ WINOUT
NETBUF2 stage 8:                         —\INMRØ6
```

```
checkdata stage 2:                    —\INMRØ6
NETPAD stage 2:                       —\INMRØ6
Ending CALLPIPE call 3
NETPAD stage 2:BX
```

CALLPIPE (CHECKDATA) terminates and the current data delimiter is OUTPUT to '*.OUTPUT.0:' and the new record flows through the pipeline.

PEEKTO tests the status of '*.INPUT.0:'. The return code is 12 and so the user stage terminates.

	Old method	*New method*
Total CPU used (secs)	1.19	0.22
Elapsed time (secs)	3	<1
RDR I/O	1291	0
PRT I/O	17	22
PUN I/O	2623	1332
DISK I/O	81	43

Figure 6: Effects of the CMS Pipeline procedure

CONCLUSION

Figure 6 shows how the new CMS Pipelines procedure reduced the CPU consumption by 81% and reduced the number of I/O operations for a 767 (fixed length 133 byte) record data file.

Hugh Suter
Software Engineer
EDS (UK)

Administering multiple machines

Event Services is a collection of REXX procedures written to help administer and control multiple machines in our installation.

Several agent procedures send events (alert messages) to a central server. This server logs all events in a logfile and performs actions, which are defined for each type of event.

Our agent procedures run in CMS under VM and on AIX – but it would be very easy to write agents for VSE and NT. They are started at regular intervals (by VMUTIL or CRONTAB) and perform checks such as the usage of VM paging space or an AIX filesystem. Whenever they find that a defined threshold is reached, they send an event containing information about the problem to the server.

The server runs in a disconnected CMS machine on our VM host. Its purpose is to log the events and to notify predefined CMS users about them so that they can take appropriate action.

There is also a procedure allowing the user to browse the event log and to see detailed information about an event, including a help text.

EVENTS

Events are sent from agent to server in the form of an event string which has the following structure:

- On position 1 is the delimiter character, which can be any character, but must be used in this event string only as a delimiter.
- The remainder of the event string consists of a maximum of 20 elements, each of which begins with an element keyword (or tag), followed by at least one blank and the element value, followed by a delimiter character.

Commas are not allowed anywhere in the event string.

Each event string contains at least these mandatory elements:

- ID – the predefined event-id or event type.

- ORIGIN – the name of the machine originating the event.
- TIMESTAMP – the creation date and time of the event in the format YYYY-MMDD/HH.MM.SS.

The event server adds an element NUMBER which contains a unique sequence number.

Each event type (identified by the ID element) must be defined in ES17 EXEC with the following parameters:

- title() – a short description of this event type.
- notify() – the names of CMS users who are to be notified on receipt of this event.
- action() – the names of REXX procedures to be executed on receipt of this event.
- helptext() – the name of a CMS file containing a help text for this event.

INSTALLATION PREREQUISITES

You will need:

- REX Sockets and REXXWAIT, two packages you can download from IBM's VM site.
- PERL on your AIX machines (for ES07).
- REXX/6000 on your AIX machines.
- IBM's Display I/O Facility for the panels in ES13.

INSTALLATION INSTRUCTIONS

- Transfer ES06, ES07, ES12, and ES19 onto your AIX machine(s).
- Put all the other procedures/panels on a generally available mini-disk.
- Make appropriate changes in ES17 EXEC.
- Prepare help text files.

- Prepare a virtual machine for the event server (with a 191-disk for the logfile) in this machine; run ES03 with the INIT parameter to create the initial logfile.
- Create a profile EXEC that calls ES03.
- Autolog the server machine.
- Prepare VMUTIL to call ES18 periodically.
- Prepare the AIX CRONTAB(s) to call ES19 periodically.

USAGE

As soon as the first events come in, execute ES13 to browse the event log.

To get rid of old events, press PF6 in the event browser.

To create your own events and agents, see ES17 (for event definitions) and ES18/ES19 (as examples for agent procedures in CMS and AIX).

EXPLANATION OF FIGURES

The Figures show Log Viewer example panels. Figure 1 shows an

```
EventServices     Log Viewer

      ID

      ORIGIN

      All events are shown when no selection criteria are entered

      ID and ORIGIN can be entered partially


     ENTER:Display Log          PF3:Exit     PF6:Reset Log
```

Figure 1: Example of selection screen

```
EventServices   LogViewer

 Number          ID                  Origin              Timestamp
 00000000000000 DUMMYEVENT          SERVER              1998-06-13/14:18:01
 00000000000001 VMPAGINGFULL        MAINT               1998-06-13/14:20:38
 00000000000003 AIXFSFULL           chefvc01            1998-06-13/14:44:34
 00000000000004 AIXFSFULL           chefvc01            1998-06-13/14:44:36

      Number

    PF2:Show Details  PF3:Exit  PF4:Bot  PF5:Top  PF7:Up  PF8:Down
```

Figure 2: Example of event list screen

example of the selection screen and Figure 2 shows an example of the event list screen. Figures 3 and 4 show examples of help display screens.

```
EVENT    HELPTEXT A1  V 80            1 BLKS 98/06/13 LINE       1 OF      16
 ===>                                                                 BROWSE
── Event Data ──────────────────

ID                  :   VMPAGINGFULL
ORIGIN              :   MAINT
TIMESTAMP           :   1998-06-13/14:20:38
USED                :   18
ALLOWED             :   003

── Description ─────────────────

VM paging space nearly full

The paging space in VM is 18 % used.

Please notify your VM systems programmer immediately]

* * * End of File * * * *
```

Figure 3: Example of a help display screen

```
EVENT    HELPTEXT A1  V 83            1 BLKS 98/Ø6/13 LINE       1 OF      19
 ===>                                                                 BROWSE
── Event Data ──────────────

ID                  :   AIXFSFULL
ORIGIN              :   chefvcØ1
TIMESTAMP           :   1998-Ø6-13/14:44:36
FSNAME              :   /dev/lvØØ
USED                :   93%
LIMIT               :   8Ø%
MNTPNT              :   /tftpboot

── Description ─────────────

AIX file system nearly full

The file system /dev/lvØØ on machine  chefvcØ1
is 93% used.
The defined upper limit is  8Ø%.

The mountpoint is /tftpboot
 * * * End of File * * * *
```

Figure 4: Example of another help display screen

HELP FILE ES002 TEXT

The text for Help File ES002 reads ‘The paging space in VM is &USED % used. Please notify your VM systems programmer immediately!’.

HELP FILE ES003 TEXT

The text for Help File ES003 reads ‘The file system &FSNAME on machine &ORIGIN is &USED used. The defined upper limit is &LIMIT. The mountpoint is &MNTPNT’.

ES01 EXEC

```
/* ================================================================== */
/* Name        :  ESØ1  EXEC                                          */
```

```
/* ================================================================== */
/* Application :  Event Services                                      */
/*                                                                    */
/* Usage       :  Procedure                                           */
/*                                                                    */
/* Arguments   :  id, tag contents, tag contents, ...                 */
/*                                                                    */
/* Result      :  -                                                   */
/*                                                                    */
/* Function    :  Send an event  (CMS Version)                        */
/*                                                                    */
/* sends an event to the server via VM spooling                       */
/* ================================================================== */
server=eS17('event.server')
origin='ORIGIN' userid()

parse value date(sorted) with yyyy 5 mm 7 dd
isodate= yyyy'-'mm'-'dd

timestamp='TIMESTAMP' isodate'/'time()
dl='#'
id='ID' translate(arg(1))

s.1=dl || id || dl || origin || dl || timestamp || dl
do i=2 to arg()
   s.i=arg(i) || dl
end

s.0=arg()

'CP SPOOL 000d TO' server
'pipe stem s. | punch 000D'
'pipe cp close 000D'
'CP SPOOL 000D OFF'
```

ES02 EXEC

```
/* ================================================================== */
/* Name        :  ES02  EXEC                                          */
/* ================================================================== */
/* Application :  Event Services                                      */
/*                                                                    */
/* Usage       :  Function                                            */
/*                                                                    */
/* Arguments   :  EventString                                         */
/*                                                                    */
/* Result      :  InterpretString                                     */
/*                                                                    */
/* Function    :  Interpret an EventString                            */
```

```
/*                                                                    */
/* INTERPRET InterpretString sets the following variables:            */
/*    es.tags    the names of all tags                                */
/*    es.<tag>   the value of a tag                                   */
/* ================================================================== */
parse arg eventstring
is=''
es.=''
es.tags=''

dl=left(eventstring,1)
eventstring=substr(eventstring,2)
eventstring_o=eventstring
do forever
   parse value eventstring with element (dl) eventstring
   if element = '' then leave
   if words(element) < 2
   then
      do
        say 'error in esØ2, element='element
        say 'error in esØ2, eventstring='eventstring_o
        leave
      end
   tag=word(element,1)
   tagvalue=substr(element,wordindex(element,2))
   es.tags=es.tags tag
   is=is „es."tag"='"tagvalue"';"
end
is=is „es.tags='"es.tags"';"
return is
```

ES03 EXEC

```
/* ================================================================== */
/* Name        :  ESØ3  EXEC                                          */
/* ================================================================== */
/* Application :  Event Services                                      */
/*                                                                    */
/* Usage       :  Procedure                                           */
/*                                                                    */
/* Arguments   :  §INIT·                                              */
/*                                                                    */
/* Result      :  -                                                   */
/*                                                                    */
/* Function    :  Event Services Server Driver                        */
/*                                                                    */
/* When called with INIT, this procedure creates a new logfile        */
/* with the required dummy record                                     */
/* ================================================================== */
```

```
vmhost=es17('event.serveripaddr')
logfile=eS17('event.logfile')
logdef=es08()  /* recdef of logfile */
port=eS17('event.serverport')
if arg(1) = 'INIT'
then
   do
     parse value date(sorted) with yyyy 5 mm 7 dd
     isodate= yyyy'-'mm'-'dd
     timestamp=isodate'/'time()
     number=copies('0',14)
     logrec=left(number,15) || ,
       left('DUMMYEVENT',20) || ,
       left('SERVER',20) || ,
       left(timestamp,20) || ,
       left('dummy eventstring',400)
     'pipe var logrec | >' logfile 'a'
     return
   end

/* Set error code values                                              */
ecpref = 'RXS'
ecname = 'SER'
initialized = 0

/* Initialize                                                         */
say 'EventServer: Initializing'
call Socket 'Initialize', 'ES'
if src=0 then initialized = 1
else call error 'E', 200, 'Unable to initialize RXSOCKET MODULE'
ipaddress = Socket('GetHostId')
if src¬=0 then call error 'E', 200, 'Unable to get the local ipaddress'
address command 'REXXWAIT TEST'
if rc¬=0 then call error 'E', 200, 'Unable to load REXXWAIT MODULE'
call ResetValue 'All'
say 'EventServer: Initialized: ipaddress='ipaddress 'port='port

/* Initialize for accepting connection requests                       */
s = Socket('Socket')
if src¬=0 then call error 'E', 32, 'SOCKET(SOCKET) rc='src
call Socket 'Bind', s, 'AF_INET' port ipaddress
if src¬=0 then call error 'E', 32, 'SOCKET(BIND) rc='src
call Socket 'Listen', s, 10
if src¬=0 then call error 'E', 32, 'SOCKET(LISTEN) rc='src
call SetValue 'Socket' s 'Non-Blocking'
if wrc¬=0 then call error 'E', 36, 'Cannot set mode of socket' s
```

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

A full screen console interface – part 2

Editor's note: this month we continue the article on the full screen console interface for Disconnected Service Machines (DSM). This article is an extensive piece of work which will be published over several issues of VM Update. *Any comments or recommendations would be welcomed and should be addressed either to Xephon or directly to the author at fernando_duarte@vnet.ibm.com.*

```
WRITE100 L     R0,0(,R8)              Load length prefix
         LA    R1,4                   Length of length prefix!!!
         SR    R0,R1                  Length of record (data)
         LA    R2,4(,R8)              Address of record
         FSWRITE FSCB=PRINTLOG,BSIZE=(R0),BUFFER=(R2) Write record
         LTR   R15,R15                Any problems?
         BNZ   WRITE800               Yes, check error
         A     R8,0(,R8)              No, address next record
         CR    R8,R3                  Do we have one
         BL    WRITE100               Yes, process new record
         L     R1,PWBEG               No, address command again
         LA    R0,PWLAST              Is it the last block?
         N     R0,L'COMMWRT(,R1)
         BZ    WRITE300               No, request another block
         FSCLOSE FSCB=PRINTLOG        Yes, close PrintLog file
         B     INPR900                Go back
         SPACE
WRITE300 LA    R6,6+L'WRITECMD+4      Length of request
         MVC   READDATA(L'WRITECMD),WRITECMD Move command to buffer
         ST    R15,READDATA+L'WRITECMD Zero Control word
         BAS   R14,SEND               Send request to CSC
         B     WRITE900
         SPACE
WRITE800 LA    R6,6+L'WRITECMD+4      Length of request
         MVC   READDATA(L'WRITECMD),WRITECMD
         ST    R15,READDATA+L'WRITECMD Store return code
         FSCLOSE FSCB=PRINTLOG
         L     R1,PWBEG               Address command
         LA    R0,PWLAST              Is it the last block?
         N     R0,L'COMMWRT(,R1)
         BZ    WRITE820               No, cancel command
         LA    R1,MSGE0212
         BAS   R14,MSGDISP            Display error message
         B     WRITE900
         SPACE
WRITE820 BAS   R14,SEND               Send request to CSC
         B     WRITE900
```

```
         SPACE
WRITE900 L     R14,PRWRSV14
         BR    R14
         SPACE
         DROP  R11
         SPACE 3
*
* Setup and Configuration
*
*
CONFIG   EQU   *                       Setup and Configuration
         USING CONFIG,R11
         MVC   CSCTRGID,0(R1)          IUCV Target user-id
         MVC   PROFFILE+FSCBFN-FSCBD(L'FSCBFN),0(R1) Copy to Profile-id
         CLI   8(R1),X'FF'             Any override
         BE    CONF200
         LA    R1,8(,R1)               Yes, address first argument
         CLI   0(R1),C'('              Check for options
         BE    CONF100
         MVC   CSCTRGID,0(R1)          Move target-id
         CLI   8(R1),X'FF'             Is that all?
         BE    CONF200                 Yes, done...
         LA    R1,8(,R1)
         CLI   0(R1),C'('              Check for options again
         BE    CONF100                 Found it, process options
         MVC   CSCTRGID,0(R1)          Copy invalid operand
         LA    R15,8                   Set return code
         ERROR ERRIOPER                Display error message
         B     RETURN                  Close the shop
         SPACE
CONF100  LA    R1,8(,R1)
         CLI   0(R1),X'FF'             End of options
         BE    CONF200                 Yes, continue initialization
         CLC   CSCNOALT,0(R1)          Check for NOALT option
         BNE   CONF110
         OI    CSCFLG01,CNSLNALT       Do not use Alternate size
         B     CONF100
         SPACE
CONF110  CLC   CSCNOEDS,0(R1)          Check for NOEDS option
         BNE   CONF120
         OI    CSCFLG01,CNSLNEDS       Do not use Extended attributes
         B     CONF100
         SPACE
CONF120  MVC   CSCTRGID,0(R1)          Copy invalid option
         LA    R15,8                   Set return code
         ERROR ERRIOPTN                Display error message
         B     RETURN                  Close the shop
         SPACE
CONF200  L     R0,TRACESZ          *T* Trace Table size (double words)
         CMSSTOR OBTAIN,DWORDS=(0),MSG=YES,BNDRY=PAGE *T* Allocate stg
```

```
ST    R1,TRACEPTR         *T*
ST    R1,TRACEBEG         *T*
SLL   RØ,3                *T*
AR    R1,RØ               *T*
ST    R1,TRACEEND         *T*
L     RØ,CVBUFFSZ             IUCV Buffer size (double words)
CMSSTOR OBTAIN,DWORDS=(Ø),MSG=YES,BNDRY=PAGE
ST    R1,CVBUFBEG             Buffer address
SLL   RØ,3                    Convert double words to bytes
ST    RØ,CVBUFLEN             Buffer size in bytes
AR    R1,RØ
ST    R1,CVBUFEND             Buffer end address
L     RØ,SCBUFFSZ             Screen Buffer size
CMSSTOR OBTAIN,DWORDS=(Ø),MSG=YES,BNDRY=PAGE
ST    R1,SCBUFBEG
SLL   RØ,3
ST    RØ,SCBUFLEN
AR    R1,RØ
ST    R1,SCBUFEND
L     RØ,DSBUFFSZ             Screen Data Stream size
CMSSTOR OBTAIN,DWORDS=(Ø),MSG=YES,BNDRY=PAGE
ST    R1,DSBUFBEG
SLL   RØ,3
ST    RØ,DSBUFLEN
AR    R1,RØ
ST    R1,DSBUFEND
L     RØ,RTRVESZ              Retrieve buffer size
CMSSTOR OBTAIN,DWORDS=(Ø),MSG=YES,BNDRY=PAGE
ST    R1,RTRVEBEG
ST    R1,RTRVECUR             Initialize current entry and
XC    RTRVELST,RTRVELST       Reset last referenced entry
SLL   RØ,3
AR    R1,RØ
ST    R1,RTRVEEND
LA    RØ,CPQUSER              Address CP command
LA    R1,USERID               Area address for CP response
LA    R2,L'CPQUSER            Length of CP command
O     R2,RESPBUFF             Request response in buffer
LA    R3,USERIDL              Length of buffer
DIAG  RØ,R2,X'Ø8'
BCTR  R3,Ø                    Offset of newline byte X'15'
LA    R1,USERID(R3)           Address newline byte
MVI   Ø(R1),C' '              Replace with a blank
MVC   DSPUSER,USERID          Move userid to screen header
MVC   DSPNODE,NODEID          Move nodeid to screen header
MVC   DSPMSG,DSPMSGD          Move default message
XC    DSPCMD,DSPCMD           Clear command line
MVI   DSPCMD,IC               Position cursor
MVC   EDSUSER,USERID          Do the same for EDS screen
MVC   EDSNODE,NODEID
```

```
         MVC   EDSMSG,DSPMSGD
         MVC   EDSCMD,DSPCMD
         BR    R14
         SPACE
         DROP  R11
         SPACE 3
*
* Check and process Profile
*
*
PROFILE  EQU   *                      Check and process Profile
         USING PROFILE,R11
         ST    R14,PROFSV14
         TM    CSCFLG02,PROFRUN        Processing in progress?
         BO    PROF100                 Yes, read next record
         OI    CSCFLG02,PROFRUN        Remember we are doing it
         L     R0,CVBUFBEG             It is OK to use IUCV buffer
         FSOPEN FSCB=PROFFILE,FORM=E,BUFFER=(R0) Does Profile exist?
         LTR   R15,R15
         BNZ   PROF800                 No, forget everything
PROF100  FSREAD FSCB=PROFFILE,FORM=E,BSIZE=256
         LTR   R15,R15                 Any problems?
         BNZ   PROF700                 Yes, check for eon-of-file
         L     R1,CVBUFBEG             Address input buffer
         LR    R2,R1                   Add record length
         AR    R2,R0                   That's the end address of data
PROF200  CLI   0(R1),C' '              Skip leading blanks
         BNE   PROF300
         LA    R1,1(,R1)               Increment pointer
         CR    R1,R2                   Is it all done? (blank line)
         BL    PROF200                 No, keep going
         B     PROF100                 Yes, a blank line, ignore it
         SPACE
PROF300  CLI   0(R1),C'*'              Is it a comment?
         BE    PROF100                 Yes, ignore it
PROF400  BCTR  R2,0                    Remove trailing blanks
         CLI   0(R2),C' '
         BE    PROF400
PROF500  SR    R2,R1                   Length of data minus 1
         LA    R0,L'READDATA           Compare with field length
         CR    R0,R2
         BNH   PROF600                 Too long, display error message
         MVI   READATTN,ENTER          AID key is always ENTER
         EX    R2,PROFMVC              Move text to console buffer
         LA    R6,7(,R2)               Total length includes prfix (6)
         BAS   R14,SEND                Send it to the CSC Service-id
         B     PROF900                 Wait for a reply
         SPACE
PROF600  LR    R15,R0                  Maximum statement length
         ERROR ERRPROFL                Display error message
```

```
         B     PROF1ØØ
         SPACE
PROF7ØØ  C     R15,EOFRC               Is it just a normal EOF?
         BE    PROF8ØØ                 Yes, done with the profile
         ERROR ERRPROF                 No, display error message
PROF8ØØ  FSCLOSE FSCB=PROFFILE         Close file
         NI    CSCFLGØ2,X'FF'-PROFRUN  Remember we are finished with it
PROF9ØØ  L     R14,PROFSV14
         BR    R14
         SPACE
PROFMVC  MVC   READDATA(*-*),Ø(R1)     Move data to console I/O buffer
         DROP  R11
         SPACE 3
*
* Release allocated storage
*
*
RELEASE  EQU   *                       Release allocated storage
         USING RELEASE,R11
         L     RØ,TRACESZ          *T* Start with trace table (testing)
         L     R1,TRACEBEG         *T*
         CMSSTOR RELEASE,ADDR=(1),DWORDS=(Ø),MSG=YES *T*
         L     RØ,CVBUFFSZ             IUCV Buffer
         L     R1,CVBUFBEG
         CMSSTOR RELEASE,ADDR=(1),DWORDS=(Ø),MSG=YES
         L     RØ,SCBUFFSZ             Screen Buffer
         L     R1,SCBUFBEG
         CMSSTOR RELEASE,ADDR=(1),DWORDS=(Ø),MSG=YES
         L     RØ,DSBUFFSZ             Screen Data Stream
         L     R1,DSBUFBEG
         CMSSTOR RELEASE,ADDR=(1),DWORDS=(Ø),MSG=YES
         L     RØ,RTRVESZ              Retrieve buffer
         L     R1,RTRVEBEG
         CMSSTOR RELEASE,ADDR=(1),DWORDS=(Ø),MSG=YES
         BR    R14
         SPACE
         DROP  R11
         SPACE 3
         DS    ØD
*
* CSC Data area
*
*
CSCDATA  EQU   *                       CSC Data area
CSCNAME  DC    C'CSC     '             CMS/IUCV name
CSCPATH  DC    C'CSC     '             CONSOLE I/O path
CSCNOALT DC    C'NOALT   '             Option not to use Alternate size
CSCNOEDS DC    C'NOEDS   '             Option not to use EDS
CSCTRGID DC    C'        '             IUCV Target user-id
         SPACE
```

```
COMMRSK  DC    C'<CSC>RSK'             Reset keyboard
COMMACL  DC    C'<CSC>ACL'             Add command line to screen
COMMALM  DC    C'<CSC>ALM'             Sound the alarm
COMMCNN  DC    C'<CSC>CNN'             Display Connect Node
COMMDCL  DC    C'<CSC>DCL'             Delete command line
COMMHDR  DC    C'<CSC>HDR'             Replace screen header
COMMMCL  DC    C'<CSC>MCL'             Move data to command line
COMMMSG  DC    C'<CSC>MSG'             Display message
COMMPRT  DC    C'<CSC>PRT'             Data from PRINT command
COMMSCR  DC    C'<CSC>SCR'             Screen Data Stream
COMMTTL  DC    C'<CSC>TTL'             Replace screen title
COMMWRT  DC    C'<CSC>WRT'             Data from WRITE command
         SPACE
CHCKSEND DC    C'<CSC>INI'             Command to initiate the session
CONSBUFF EQU   *                       Buffer for CONSOLE exit
         ORG   *+CQYSIZE
CHCKLEN  EQU   *-CHCKSEND
         SPACE
         DS    ØD
READSEND DC    C'<CSC>CMD'             Buffer to send console input
READBUFF DS    ØCL8Ø                   Buffer to read console input
READATTN DS    X                       Attention key
READCRSR DS    XL2                     Cursor address
READSBA  DS    X                       SBA order
READADDR DS    XL2                     Field address
READDATA DS    CL74                    Data entered
         SPACE
CSCPARMC DS    ØD                      IUCV Parmlist for CP
         ORG   *+IPSIZE*8
         SPACE
PRINTLOG FSCB  'CSC      PrintLog A1',FORM=E,RECFM=V
PROFFILE FSCB  'CSCUSR   $PROFILE  *',FORM=E
         SPACE
PWBEG    DS    F
PWFIRST  EQU   X'Ø1'
PWLAST   EQU   X'Ø2'
PRINTCMD DC    C'PRINT '
WRITECMD DC    C'WRITE '
PRINTMAX DC    F'121'                  Maximum print line
CPCLOSEB DC    C'CLOSE PRINTER'
CPCLOSEA DC    C'CLOSE PRINTER NAME CSC PrintLog'
         SPACE
HELPSAVE DS    D                       Area to save Help command
HELPPL   DC    C'HELP    '             Help Parameter List
HELPTYPE DC    C'CSC     '
         DC    C'MENU    '
         DC    X'FFFFFFFFFFFFFFFF'
HELPEPL  DC    A(HELPPL)               Help Extended Parameter List
HELPEPLA DS    F
HELPEPLE DS    F
```

```
         DC    F'Ø'               *4* Extended Parameter List word 4
         SPACE
CMSSS    DC    C'SUBSET  '            Command to enter CMS Subset
CMSPL    DC    C'CMS     '            Command to execute CMS command
CMSEPL   DC    A(CMSPL)               CMS Extended Parameter List
CMSEPLA  DS    F
CMSEPLE  DS    F
         DC    F'Ø'               *4* Extended Parameter List word 4
CSCWORK  DS    F
CSCBLANK DC    C'    '
CSCCMSP  DC    C'CMS '
         SPACE
FFFFFFFF DC    X'FFFFFFFF'
RESPBUFF DC    X'40000000'            Option for DIAG ØØØ8
EOFRC    DC    F'12'                  End of file RC from FSREAD
STOLEN   DC    F'32'                  Console stolen by another appl
         SPACE
CSCFLGØ1 DC    X'ØØ'                  CSC Flag byte Ø1
HNDIOS   EQU   X'8Ø'                  HNDIO SET executed
HNDIUCVS EQU   X'4Ø'                  HNDIUCV SET executed
CMSIUCVC EQU   X'2Ø'                  CMSIUCV CONNECT executed
CNSLOP   EQU   X'1Ø'                  CONSOLE OPEN executed
CNSLALT  EQU   X'Ø8'                  CONSOLE Alternate size supported
CNSLNALT EQU   X'Ø4'                  Do not use Alternate size
CNSLEDS  EQU   X'Ø2'                  CONSOLE EDS supported
CNSLNEDS EQU   X'Ø1'                  Do not use Extended attributes
         SPACE
CSCFLGØ2 DC    X'ØØ'                  CSC Flag byte Ø2
CSCWAIT  EQU   X'8Ø'                  Waiting for work
WORKIO   EQU   X'4Ø'                  Work to do (Console IO)
WORKID   EQU   X'2Ø'                  Work to do (IUCV connection)
WORKCV   EQU   X'1Ø'                  Work to do (IUCV interrupt)
WORKEND  EQU   X'Ø8'                  END command entered
CMDREDSP EQU   X'Ø4'                  Redisplay (&) command entered
PROFRUN  EQU   X'Ø1'                  Profile being processed
         SPACE
CSCFLGØ3 DC    X'ØØ'                  CSC Flag byte Ø3 (IUCV messages)
CVFIRST  EQU   X'8Ø'                  First time scheduled
CVCC     EQU   X'4Ø'                  Connection completed
CVCCERR  EQU   X'2Ø'                  Connection error
         SPACE
CSCFLGØ4 DC    X'ØØ'                  CSC Flag byte Ø4
CVSEVER  EQU   X'8Ø'                  IUCV connection severed
CVERROR  EQU   X'4Ø'              ??? IUCV error (maybe not)
CVMSGIN  EQU   X'2Ø'                  IUCV incoming message
CVSEND   EQU   X'1Ø'                  IUCV SEND in progress
CVRCVE   EQU   X'Ø8'                  IUCV RECEIVE completed
SCRDISP  EQU   X'Ø4'                  Screen refresh required
SCRTRL   EQU   X'Ø2'                  Refresh screen trailer (bottom)
SCRALM   EQU   X'Ø1'                  Sound the alarm
```

```
         SPACE
@DISPLAY DC    A(DISPLAY)              Routine addresses
@PRINT   DC    A(PRINT)
@WRITE   DC    A(WRITE)
@CONFIG  DC    A(CONFIG)
@PROFILE DC    A(PROFILE)
@RELEASE DC    A(RELEASE)
         SPACE
ERROSV11 DS    F                       Save R11   ERROR
IOPRSV14 DS    F                       Save R14   IOPROC
IDPRSV14 DS    F                                  IDPROC
CVPRSV14 DS    F                                  CVPROC
COMMSV14 DS    F                                  COMMAND
INPRSV14 DS    F                                  INPROC
CHCKSV14 DS    F                                  CHECK
SENDSV14 DS    F                                  SEND
DISPSV14 DS    F                                  DISPLAY
PRWRSV14 DS    F                                  PRINT / WRITE
PROFSV14 DS    F                                  PROFILE
         SPACE
TRACESZ  DC    F'512'              *T* Trace Table size (double words)
TRACEPTR DS    F                   *T* Pointer to current entry
TRACEBEG DS    F                   *T* Begin of Trace Table
TRACEEND DS    F                   *T* End of Trace Table
         SPACE
CVBUFFSZ DC    F'512'                  IUCV Buffer size (double words)
CVBUFBEG DS    F                          Address
CVBUFEND DS    F                          End address
CVBUFLEN DS    F                          Length
         SPACE
SCBUFFSZ DC    F'512'                  Screen Buffer size
SCBUFBEG DS    F                          Address
SCBUFEND DS    F                          End address
SCBUFLEN DS    F                          Length
         SPACE
DSBUFFSZ DC    F'512'                  Screen Data Stream size
DSBUFBEG DS    F                          Address
DSBUFEND DS    F                          End address
DSBUFLEN DS    F                          Length
         SPACE
RTRVESZ  DC    F'512'                  User Retrieve buffer size
RTRVEBEG DS    F                          Address
RTRVEEND DS    F                          End address
RTRVECUR DS    F                          Current entry
RTRVELST DS    F                          Last referenced entry
         SPACE
CSCPOST  EQU   X'40'                   ECB Event completed bit
CSCECB   DS    F                       ECB
CSCRC    DS    F                       CSC Return Code
CVMSGQ   DS    F                       CP messages queued
```

```
THREE    DC    F'3'
         SPACE
DIAG000C DS    4D                      Work are for DIAG 000C
USERID   DS    CL8                     Response from CP Query USERID
         DS    CL4                     ' AT '
NODEID   DS    CL12
USERIDL  EQU   *-USERID
CPQUSER  DC    C'QUERY USERID'         CP command QUERY USERID
         SPACE
ALARM    EQU   X'04'                   Alarm bit in WCC
EHIRESET DC    X'28400'                Reset extended highlight
SCRLEN1  DS    F                       Length for top and middle screen
SCRLEN2  DS    F                       Length for complete screen
         SPACE
         DS    0D
DSPBUFF  DC    X'C3114040'             Data stream (no EDS)
         DC    X'1D60'
DSPNODE  DC    C'________'              Node-id
         DC    X'1140D8',X'1DF8'
DSPTITLE DC    C'<CSC>    <CSC>  <CSC>    <CSC>' Default title
         DC    X'11C1C6',X'1D60'
DSPUSER  DC    C'________'              User-id
         DC    X'11C14F',X'1D60'
DSPHDR   DC    60C'_'                  Header line
DSPHDRL  EQU   *-DSPHDR
         SPACE
DSPDATE  DC    C'__/__/_____'          Date
DSPTIME  DC    C'__:__:__'             Time
         DC    X'1D60'
DSPLENT  EQU   *-DSPBUFF               Length of top screen
         SPACE
DSPTRL   DC    C'______'
DSPCNN   DC    C'________'
         DC    C'___'
         DC    C'________'
         DC    54C'_'
         DC    X'1D60',C'====>',X'1DC1'
DSPLENC  EQU   *-DSPTRL                Length of middle screen
         SPACE
DSPCMD   DS    CL73                    Command line
         DC    X'1D'
DSPMSGA  DC    X'60'                   Message attributes
         DC    C'       '
DSPMSG   DS    CL74                    Message line
DSPLENB  EQU *-DSPCMD                  Length of bottom screen
         SPACE
EDSBUFF  DC    X'C3114040'             Data stream (EDS)
         DC    X'290242F1C060'
EDSNODE  DC    C'________'              Node-id
         DC    X'1140D9',X'2841F4'
```

```
EDSTITLE DC    C'<CSC>    <CSC>  <CSC>    <CSC>' Default title
         DC    X'11C1C7',X'284100'
EDSUSER  DC    C'________'             User-id
         DC    X'11C150',X'2842F5'
EDSHDR   DC    60C'_'                 Header line
EDSHDRL  EQU   *-EDSHDR
         SPACE
         DC    X'2842F3'
EDSDATE  DC    C'__/__/__'            Date
         DC    X'2842F5',C'___',X'2842F3'
EDSTIME  DC    C'__:__:__'            Time
         DC    X'1D60'
EDSLENT  EQU   *-EDSBUFF              Length of top screen
         SPACE
EDSTRL   DC    X'2842F5',C'______'
         DC    X'2841F2'
EDSCNNA  DC    X'284100'
EDSCNN   DC    C'________'
         DC    C'___'
         DC    C'________'
         DC    54C'_'
         DC    C' ====>',X'2842F61DC1'
EDSLENC  EQU *-EDSTRL                Length of middle screen
         SPACE
EDSCMD   DS    CL73                   Command line
         DC    X'1D60',C'       ',X'2842'
EDSMSGC  DC    X'F1'                  Default colour (blue)
EDSMSG   DS    CL74                   Message line
EDSLENB  EQU *-EDSCMD                Length of bottom screen
         SPACE
DSPMSGD  DC    C'PF  1=Hlp 3=End 4=Top 5=Bot 6=Rep 7=Bwd 8=Fwd 9=Cur 10*
               =Shf 11=Rtf 12=Rtb  '
         SPACE
         DS    0D
USRCOMM  DS    CL16
BLANKS   DC    CL16' '
         SPACE
LCLTABLE DC    X'00',C'?          ',A(RTVCMD)    One byte commands
         DC    X'00',C'=          ',A(REPCMD)
         DC    X'FFFFFFFFFFFFFFFF',X'0000000000000000'
         SPACE
         DC    X'02',C'CMS        ',A(CMSCMD)    Local commands
         DC    X'00',C'HELP       ',A(HELPCMD)
         DC    X'FFFFFFFFFFFFFFFF'
         SPACE
PFKTABLE DC    A(PF1,HELPPFK)
         DC    A(PF3,ENDCMD)
         DC    A(PF6,REPCMD)
         DC    A(PF11,RTVFPFK)
         DC    A(PF12,RTVBPFK)
```

```
         DC    A(PF13,HELPPFK)
         DC    A(PF15,ENDCMD)
         DC    A(PF18,REPCMD)
         DC    A(PF23,RTVFPFK)
         DC    A(PF24,RTVBPFK)
         DC    A(CLEAR,CLEARCMD)
         DC    X'FFFFFFFFFFFFFFFF'
         SPACE 3
         LTORG
         SPACE 3
         DS    0F
ERRIOPER DC    X'FFFFFFFF',A(ERROR800,MSGE0220) Error messages control
ERRIOPTN DC    X'FFFFFFFF',A(ERROR800,MSGE0222)
ERRIUCVS DC    X'00000004',A(ERROR600,MSGE0230)
         DC    X'FFFFFFFF',A(ERROR200,MSGE0231)
ERRIUCVC DC    X'000003F3',A(ERROR800,MSGE0240)
         DC    X'000003F4',A(ERROR800,MSGE0241)
         DC    X'000003F6',A(ERROR800,MSGE0242)
         DC    X'000003F7',A(ERROR800,MSGE0243)
         DC    X'000003F8',A(ERROR800,MSGE0244)
         DC    X'FFFFFFFF',A(ERROR200,MSGE0249)
ERRCNSLO DC    X'FFFFFFFF',A(ERROR200,MSGE0250)
ERRCNSLC DC    X'FFFFFFFF',A(ERROR200,MSGE0252)
ERRIUCVE DC    X'FFFFFFFF',A(ERROR200,MSGE0254)
ERRIUCVT DC    X'FFFFFFFF',A(ERROR100,MSGE0256)
ERRIOPR  DC    X'FFFFFFFF',A(ERROR200,MSGE0260)
ERRIDPR  DC    X'FFFFFFFF',A(ERROR100,MSGE0262)
ERRPROF  DC    X'FFFFFFFF',A(ERROR200,MSGE0264)
ERRPROFL DC    X'FFFFFFFF',A(ERROR200,MSGE0265)
ERRCVPR  DC    X'00000064',A(ERROR100,MSGE0270)
         DC    X'00000065',A(ERROR100,MSGE0271)
         DC    X'FFFFFFFF',A(ERROR100,MSGE0279)
ERRCPRV  DC    X'00000005',A(ERROR100,MSGE0280)
         DC    X'FFFFFFFF',A(ERROR400,MSGE0289)
ERRCMM   DC    X'FFFFFFFF',A(ERROR400,MSGE0290)
ERRDSP   DC    X'FFFFFFFF',A(ERROR200,MSGE0292)
         SPACE
         DC    AL1(L'MSGE0200)
MSGE0200 DC    C'CSCUSR0200W Only valid after first command is entered.*
               '
         DC    AL1(L'MSGE0202)
MSGE0202 DC    C'CSCUSR0202W No more data found in Retrieve buffer.'
         DC    AL1(L'MSGE0204)
MSGE0204 DC    C'CSCUSR0204W CMS command ended with non zero return cod*
               e.'
         DC    AL1(L'MSGE0206)
MSGE0206 DC    C'CSCUSR0206E Command ACL not yet implemented.'
         DC    AL1(L'MSGE0208)
MSGE0208 DC    C'CSCUSR0208E Command DCL not yet implemented.'
         DC    AL1(L'MSGE0210)
```

```
MSGE021Ø DC    C'CSCUSRØ21ØE Invalid data received from the CSC Service*
                machine.'
         DC    AL1(L'MSGEØ212)
MSGEØ212 DC    C'CSCUSRØ212E Error creating PrintLog file.'
         SPACE
         DC    AL1(L'MSGEØ22Ø)
MSGEØ22Ø DC    C'CSCUSRØ22ØE Invalid operand: &&1.'
         DC    AL1(L'MSGEØ222)
MSGEØ222 DC    C'CSCUSRØ222E Invalid option: &&1.'
         DC    AL1(L'MSGEØ23Ø)
MSGEØ23Ø DC    C'CSCUSRØ23ØE Program &&1 already active.'
         DC    AL1(L'MSGEØ231)
MSGEØ231 DC    C'CSCUSRØ231E Error executing HNDIUCV SET. Return code i*
               s &&1.'
         DC    AL1(L'MSGEØ24Ø)
MSGEØ24Ø DC    C'CSCUSRØ24ØE CSC Service machine ''&&1'' is not logged *
               on.'
         DC    AL1(L'MSGEØ241)
MSGEØ241 DC    C'CSCUSRØ241E CSC Servive machine ''&&1'' is not initial*
               ized.'
         DC    AL1(L'MSGEØ242)
MSGEØ242 DC    C'CSCUSRØ242E Maximum connections for ''&&1'' CSC Servic*
               e machine exceeded.'
         DC    AL1(L'MSGEØ243)
MSGEØ243 DC    C'CSCUSRØ243E You are not authorized to connect to CSC S*
               ervice machine ''&&1''.'
         DC    AL1(L'MSGEØ244)
MSGEØ244 DC    C'CSCUSRØ244E Invalid IUCV System Service name ''&&1''.'
         DC    AL1(L'MSGEØ249)
MSGEØ249 DC    C'CSCUSRØ249E Unexpected error from CMSIUCV CONNECT. Ret*
               urn code is &&1.'
         DC    AL1(L'MSGEØ25Ø)
MSGEØ25Ø DC    C'CSCUSRØ25ØE Unable to initialize console. Return code *
               from CONSOLE OPEN is &&1.'
         DC    AL1(L'MSGEØ252)
MSGEØ252 DC    C'CSCUSRØ252E Unable to restore console. Return code fro*
               m CONSOLE CLOSE is &&1.'
         DC    AL1(L'MSGEØ254)
MSGEØ254 DC    C'CSCUSRØ254E Unable to clear IUCV handler. Return code *
               from CMSIUCV SEVER is &&1.'
         DC    AL1(L'MSGEØ256)
MSGEØ256 DC    C'CSCUSRØ256E Unable to terminate IUCV session.'
         DC    AL1(L'MSGEØ26Ø)
MSGEØ26Ø DC    C'CSCUSRØ26ØE Console read error. Return code from CONSO*
               LE READ is &&1.'
         DC    AL1(L'MSGEØ262)
MSGEØ262 DC    C'CSCUSRØ262E Invalid destination.'
         DC    AL1(L'MSGEØ264)
MSGEØ264 DC    C'CSCUSRØ264E Error reading Profile. Return code from FS*
               READ is &&1.'
```

```
         DC    AL1(L'MSGE0265)
MSGE0265 DC    C'CSCUSR0265E Profile statement too long. Maximum is &&1*
                characters.'
         DC    AL1(L'MSGE0270)
MSGE0270 DC    C'CSCUSR0270E Unable to connect to the CSC Service Machi*
               ne.'
         DC    AL1(L'MSGE0271)
MSGE0271 DC    C'CSCUSR0271E IUCV connection severed by the CSC Service*
                machine.'
         DC    AL1(L'MSGE0279)
MSGE0279 DC    C'CSCUSR0279E Unexpected interrupt from IUCV.'
         DC    AL1(L'MSGE0280)
MSGE0280 DC    C'CSCUSR0280E IUCV Receive buffer too small.'
         DC    AL1(L'MSGE0289)
MSGE0289 DC    C'CSCUSR0289E IUCV Receive error. IPRCODE is &&1.'
         DC    AL1(L'MSGE0290)
MSGE0290 DC    C'CSCUSR0290E IUCV Send error. IPRCODE is &&1.'
         DC    AL1(L'MSGE0292)
MSGE0292 DC    C'CSCUSR0292E Console I/O error. Return code from CONSOL*
               E WRITE is &&1.'
         SPACE
RTVSECT  DSECT
RTVPLUS  DS    H                       Offset to next entry
RTVLESS  DS    H                       Offset to previous entry
RTVLEN   DS    H                       Length of current entry
RTVDATA  DS    C                       Data
         SPACE 3
         PUSH  PRINT
         PRINT OFF
         COPY  IPARML
         POP   PRINT
         CQYSECT
         DMSDSBLK
         FSCBD
         NUCON
         REGEQU
         END
```

THE CSC MACRO LIBRARY

Before you can assemble any of the components of the service program CSCSVP, you must create the CSC macro library. It uses the files CSCHDR, CSCLINK, CSCDATA, CSCDS, and CSCCMMD. Create the macro library with the command:

```
MACLIB GEN CSC CSCHDR CSCLINK CSCDATA CSCDS CSCCMMD
```

Then you can assemble the subprograms with the commands:

```
GLOBAL MACLIB CSC DMSGPI DMSOM HCPGPI
HLASM CSCxxx
```

To create the executable module enter:

```
LOAD CSCSVP (CLEAR RLDSAVE
GENMOD CSCSVP
```

CSCHDR MACRO

```
         MACRO
&LABEL   CSCHDR
         LCLC  &NAME,&DATE,&TIME
&NAME    SETC  '&SYSECT        '(1,8)
&DATE    SETC  '&SYSDATC'(1,4)'/'.'&SYSDATC'(5,2)'/'.'&SYSDATC'(7,2)
&TIME    SETC  ' &SYSTIME'
*-------------------------------------*
*                                                                     *
*  CSCSVP Register usage                                              *
*                                                                     *
*              RØ-R3 Work registers                                   *
*              R4-R5 Work registers (carefully)                       *
*                                                                     *
*              R5  MSGSECT   MSG table                                *
*              R6  CCSBUFF   Scanning IUCV message                    *
*              R7  CCHSECT   Cache record                             *
*              R8  UIDSECT   User block                               *
*              R9  IUCV Parameter List                                *
*                                                                     *
*              R1Ø Base - Data area                                   *
*              R11 Base - Independent routines                        *
*              R12 Base - Common code                                 *
*-------------------------------------*
&SYSECT  RMODE ANY
         DC    A(HDRLEN-*)             Length of timestamp + prefix
         DC    C'&NAME&DATE&TIME'
         DC    C' Copyright CSC Inc, 1997'
HDRLEN   EQU   *
         USING *,R11                   Base for code
         USING CSCDATA,R1Ø             Base for common Data area
         DS    5F                      Area for Offset and R11-R14
         STM   R11,R14,4(R15)          Save R11-R14
         LR    R13,R15                 Address new save area
         LR    R11,R15                 Load base address
         MEND
```

CSCLINK MACRO

```
        MACRO
&LABEL   LINK  &WHERE
```

```
         LCLC  &ADC
&ADC     SETC  '@&WHERE'(1,8)
&LABEL   L     R15,&ADC                 Load routine address
         BASR  R14,R15                  Execute
         MEND
         MACRO
&LABEL   GO    &WHERE
         LCLC  &ADC
         AIF   (T'&WHERE NE 'O').GEN
&LABEL   BAS   R14,2Ø(,R15)             Execute
         MEXIT
.GEN     ANOP
&ADC     SETC  '@'.'&WHERE'(2,7)
&LABEL   L     R15,&ADC                 Load routine address
         A     R15,Ø(,R15)              Skip timestamp
         BAS   R14,2Ø(,R15)             Execute
         MEND
         MACRO
&LABEL   RELOC
         ENTRY &LABEL
&LABEL   DC    A(4)                     Length of timestamp + prefix
         DC    A(*-&SYSECT)             Entry offset to CSECT
         DS    4F                       Save area for R11-R14
         STM   R11,R14,4(R15)           Save R11-R14
         LR    R13,R15                  Address new save area
         S     R15,Ø(,R15)              Address CSECT
         A     R15,Ø(,R15)              Skip CSECT timestamp
         LR    R11,R15                  Restore base register
         MEND
         MACRO
&LABEL   BACK
&LABEL   LM    R11,R14,4(R13)           Restore R11-R14
         BR    R14                      Return
         MEND
         MACRO
&LABEL   MSG   &MSGNUM,&OUT,&GARB
         LCLA  &COUNT
         LCLB  &RC,&USER,&NOCMD,&NOALARM,&SPACE,&CC
         LCLC  &OPTS,&MODULE
&RC      SETB  Ø
&USER    SETB  Ø
&NOCMD   SETB  Ø
&NOALARM SETB  Ø
&SPACE   SETB  Ø
&CC      SETB  Ø
.*
```

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

Fernando Duarte
Analyst (Canada)

The L-Soft International Web site

Continuing the series of VM Web site reviews, we visit the L-Soft International Web site, which can be accessed at http://www.lsoft.com.

It isn't surprising that a company offering a product that predated and helped shape the modern Internet uses the Net for marketing, commerce, visibility, information dissemination, and public service. E-mail is often called the 'killer application', and L-Soft International's LISTSERV, the premier mailing list management software product, greatly enhances the benefits of using e-mail. In fact, subscription-based mailing lists were an early form of information 'push' distribution, long before that buzzword was coined.

It's a constant irritant to people with perspectives longer than the last few Microsoft Windows releases, that the computing industry has very little collective memory, persists in repeatedly solving the same problems, and painfully gathers the same insights for each new technology generation. Specifically, mainframe technology is often scorned and discounted by devotees of more fashionable technologies. It would no doubt surprise them that the structure and implementation of e-mail mailing lists emerged, developed, and often remains on System/390 platforms. The briefest definition of LISTSERV comes from L-Soft's opening page on the product:

"LISTSERV(r) is a system that allows you to create, manage, and control electronic mailing lists on your corporate network or on the Internet. Since its inception in 1986, LISTSERV has been continually improved and remains the predominant system in use today."

Apart from the functional definition given, there's an important but subtle fact here, and another fact omitted. The (r) designation means that the product name is a registered trademark and that LISTSERV is distinctly not a generic term for 'mailing list' or 'mailing list management software', no matter how many people misuse it in those fashions.

The omitted fact is that LISTSERV was born on academic VM systems, which were at that time connected via a network called

BITNET. While still used and popular on large and small VM systems, the Web site notes that it is currently also available for VMS, 13 Unix brands, Windows NT, and Windows 95, and is also being ported to the Macintosh and to MPE (HP3000). Many aspects of LISTSERV's VM heritage remain visible, and many features are still unique to the VM version, so a tour of L-Soft's Web site offers both historical insight into an important product's twelve-year (so far!) evolution, and resources to use mailing lists more effectively, whether as a subscriber or a list owner.

Snippets from LISTSERV's history on the Web site show the birth of a technical idea, as has been common in the VM community, followed by its adoption, popularization, and ultimate commercialization. The story begins with Eric Thomas, LISTSERV's developer and still driving force, in Paris.

Together with the other computer enthusiasts in his class, Eric Thomas visited various universities in the Paris area to find out what kind of computing equipment they had, and under what conditions students could get access to it. At the time, most university students did not have access to any computer equipment at all – at least not unless they were studying computing science.

Deciding that boldness was required, Eric headed for the system administrator's office and offered to help with system management and to develop software in exchange for an account on the machine. The deal was sealed when he mentioned that he was an expert on VM security. A number of students had recently broken into the system, attracting the attention of the higher-ups. Eric showed the system administrator how to seal the system tightly and identify the perpetrators, and in exchange was allowed to use the system.

The machine was connected to BITNET, and this is where the history of LISTSERV begins.

LISTSERV improved over the years, with an average of two new versions a year (and Eric moved to Sweden). Until 1991, there was no similar software for Unix. This was not a problem because LISTSERV supported remote list owners. Thus, you did not need to learn anything about mainframes or even get a mainframe account to use LISTSERV,

you just had your computing centre set up the list for you on the local mainframe. Mainframes were routinely upgraded as they ran out of processing power. LISTSERV was usually a drop in the ocean, because networks were not fast enough to let LISTSERV use much CPU time. Besides, most Internet people felt that there was no need for LISTSERV at all and that Usenet should be used instead, and no one was really interested in developing a Unix list manager.

This situation, however, changed gradually over the years as IBM went through the crisis that we all know about. People began to get rid of their mainframes – and since LISTSERV ran only under VM, it would have to go with the IBM iron. Eric did not want his software to disappear with mainframes, so in 1991 he started looking for ways to port LISTSERV to other environments. Initially, he changed existing code to make it easier to port should the need arise. In 1992, he determined that the time had come to take a more active role, and started writing proposals. Eric's goal was to 'make it happen', and to remain involved in the design and coordination of this development so that the 'spirit' of LISTSERV would be preserved in the new package.

In 1993, Eric decided that this grant search was going nowhere, and that it was time to stop talking and start getting things done. But where would the money come from? Eric could not borrow that kind of money in Sweden, because he had only lived there for three years, and the banking world is just not designed for people who emigrate every few years.

Several people had contacted Eric over the years, suggesting that he start a business and offering assistance. Initially, this had not sounded very interesting; however, grant money had failed to materialize, and going commercial appeared to be the only way to save LISTSERV from certain extinction in the mainframe's death grip. With a fresh determination, Eric reached business people who had contacted him earlier, and this is how L-Soft came to be.

Illustrating growth, we see that on 6 May 1988, the 1,000th public LISTSERV list was created; worldwide statistics for Sunday 5 July 1998 were:

- Number of public lists – 18,772
- Number of local lists – 66,676
- Total number of lists – 85,448
- Total membership (public+local) – 33,290,995
- Total messages delivered today – 18,488,910.

Many links are provided to one of the most useful sections of the Web site – on-line LISTSERV documentation. It's so easy to establish LISTSERV lists, or subscribe to them, that some list owners and subscribers never consult any publications. This is sometimes convenient for quick-starting a list-related effort, but is not always the best long-term strategy. Documentation divides into three categories:

- For subscribers (people reading and posting to mailing lists).
- For list owners (people establishing and maintaining mailing lists).
- For LISTSERV software maintainers (people formerly – and sometimes still – proudly called system programmers).

Most people won't read or need the last category, but power subscribers can benefit from reading about what's involved in defining and administering lists. For all interested parties, LISTSERV offers extensive customization options, so that different subscribers and mailing lists are likely to interact differently with LISTSERV. In addition to manuals available from the Web site, each LISTSERV server will distribute three reference cards which answer the most common usage questions. To request these, send e-mail to your favourite – or any – LISTSERV system containing in the body (LISTSERV always ignores e-mail subjects) 'get listserv refcard'.

Remember to delete extraneous text such as your signature file. You'll receive two e-mails in return. The first will show execution of your job (since that's how the e-mailed request is handled, just like a batch job submitted to a service virtual machine). An example is shown in Figure 1.

```
Date: Mon, 6 Jul 1998 23:36:45 -0400
  From: "L-Soft list server at Penn State (1.8c)" <LISTSERV@LISTS.PSU.EDU>
  To: Gabe Goldberg <gabe@CPCUG.ORG>
  Subject: Output of your job "gabe"

  > info refcard

  Summary of resource utilization
  -------------------

   CPU time:        0.284 sec               Device I/O:       58
   Overhead CPU:    0.018 sec               Paging I/O:       90
   CPU model:        9021                   DASD model:     3390
```

Figure 1: Example of first e-mail reply

The second e-mail contains the promised reference card. An example of the beginning of this is shown in Figure 2.

```
LISTSERV(R) System Reference Library, release 1.8c
                          Last update: 13 Aug 1996

        ************************************************************
        *                                                          *
        * LISTSERV command reference cards:                        *
        *                                                          *
        *-> LISTSERV REFCARD: General user commands                *
        *   LISTOWNR REFCARD: List and file management commands    *
        *   LISTMAST REFCARD: Commands for the LISTSERV administrat*
        *                                                          *
        ************************************************************

  List subscription commands (from most to least important)

  SUBscribe    listname <full_name>   Subscribe to a list, or change
                                      your name if already subscribed
                          ANONYMOUS   -> Subscribe anonymously

  SIGNOFF                            Remove yourself:
               listname              - From the specified list
               *                     - From all lists on that server
               * (NETWIDE            - From all lists in the network
```

Figure 2: Example of reference card

Clearly, the most important general-user commands deal with joining and leaving mailing lists. Other command and information categories are 'Other list-related commands', 'Informational commands', 'Commands related to file server functions', 'Other (advanced) commands', and 'Syntax of parameters'. Occasionally perusing reference cards can trigger ideas for optimizing one's use of mailing lists. For example, when a list's traffic grows to the point where it's a burden, switching to receiving list postings in digest form can be pleasant. Setting the 'digest' option causes LISTSERV to send postings in groups, each as a single e-mail, often just once per day. So (for example) sending the command 'set vmesa-l digest' to LISTSERV@UAFSYSB.UARK.EDU (after subscribing to the list, of course) will get you a daily compilation of messages (ranging from a few to a few dozen) about today's VM system and its use/operation. This is quite wide ranging and dynamic, with strong IBM participation.

You can request the other two reference cards listed – LISTOWNR REFCARD and LISTMAST REFCARD – in similar fashion. For more detailed and truly comprehensive information, visit the on-line documentation page, with links for several publications, many of which offer online viewing or downloading in numerous formats.

- On-line LISTSERV documentation:
  - *LISTSERV General User's Guide*
  - *LISTSERV List Owner's Quick Start*
  - *LISTSERV List Owner's Manual*
  - *LISTSERV Site Manager's Manual*
  - *The LISTSERV Support FAQ.*
- Installation guides for LISTSERV Version 1.8c:
  - *LISTSERV for VM Installation Guide*
  - *LISTSERV for VMS Installation Guide*
  - *LISTSERV for Unix Installation Guide*
  - *LISTSERV for Windows NT Installation Guide.*

- Release notes for LISTSERV Version 1.8c (via FTP):
  - *List Owner's Release Notes for LISTSERV 1.8c*
  - *Maintainer's Release Notes for LISTSERV 1.8c.*

The final LISTSERV topic is something of an anti-e-mail page – their Spam-o-rama information, explaining 'Spamming' as "*an Internet term invented to describe the act of cross-posting the same message to as many newsgroups and/or mailing lists as possible, whether or not the message is germane to the targeted newsgroups or mailing lists. It also refers to unwanted e-mail solicitations sent to an individual whose e-mail address has fallen into the wrong hands.*"

Although in the business of facilitating e-mail communication, L-Soft – like all responsible vendors and ISPs – actively discourages and inhibits spamming. LISTSERV has a combination of design features and configuration options which can greatly reduce the ability of spammers to use mailing lists as a spam launch vehicle.

Since L-Soft is, after all, interested in marketing LISTSERV and its other products, a few pages are devoted to the advantages of LISTSERV compared with other list managers, performance of LISTSERV under various loads, user testimonials, and uses of LISTSERV by business and individuals. In fact, several L-Soft offerings appeal to small and medium sites. Firstly, LISTSERV Lite offers most features of its big brother, but does not include advanced functions critical for larger sites. Sadly, this version doesn't support VM. There's even a Free Edition, described as "*a freeware version of LISTSERV Lite, limited to a maximum of 10 mailing lists with up to 500 subscribers each. This version costs absolutely nothing as long as the licensee does not derive a profit, directly or indirectly, from using the software. You can download the Free Edition from our WWW or FTP servers.*"

A key valuable support resource is various e-mail discussion lists covering different LISTSERV versions, in which L-Soft staffers participate vigorously, which often provides tips on list usage and maintenance, perspective on how things came to be the way they are, and previews of things to come.

For those wanting to run mailing lists without the mixed joys and

sorrows of arranging for host sites and performing technical administration, L-Soft offers:

"EASE(sm) services, standing for Expert Administration and Supervision of E-mail lists. EASE services' primary goal is to allow non-technical users to set up mailing lists to meet business or academic needs: electronic newsletters, announcements, conference planning, academic discussion groups, working groups, etc. An L-Soft expert list administrator describes various options available and implements necessary changes. Customers need not learn how LISTSERV works.

EASE Business services offer five levels of technical assistance, ranging from simple list hosting to "electronic secretary" services. EASE Bulk services can handle hundreds of thousands of deliveries in a very cost-effective manner. And home users, clubs and associations, can now economically create (non-commercial) mailing lists EASE Home services."

As statistics cited earlier show, with nearly 20,000 public LISTSERV-managed lists available, there are lists to meet nearly any taste or requirement. But slogging through the myriad host sites and directories would be a daunting task, since lists can emerge and remain anywhere in the world where an interested and energetic person chooses to start one. L-Soft maintains a database and several search Web pages to bring order out of the chaos. Clicking the CataList logo brings up a page beginning:

"Welcome to CataList, the catalog of LISTSERV lists! From this page, you can browse any of the 17,505 public LISTSERV lists on the Internet, search for mailing lists of interest, and get information about LISTSERV host sites. This information is generated automatically from LISTSERV's LISTS database and is always up-to-date."

and offering several ways to search for information:

- List information:
  - search for a mailing list of interest.
  - view lists by host site.

  - view lists by host country.
  - view lists with 10,000 subscribers or more.
  - view lists with 1,000 subscribers or more.
- Site information:
  - search for a LISTSERV site of interest.
  - view sites in alphabetical order.
  - view sites by country.
- Information for list owners:
  - provide HTML descriptions for your lists to enhance their appearance in the database.
  - view or download the LISTSERV list owner's guide.
  - check mailing lists of interest to list owners.

Note the last item listed, which offers ten mailing lists on various topics related to operating mailing lists – surely an appropriate use of the technology. Searching for lists with 'mainframe', 'System/390', or 'S/390' in the name or title yielded the following:

- ASSEMBLER-LIST@UGA.CC.UGA.EDU – IBM mainframe Assembler list (458 subscribers).
- CANDLE-L@UA1VM.UA.EDU – Candle Corporation mainframe monitoring products discussion list (161 subscribers).
- DOM390-L@WVNVM.WVNET.EDU – OS/390 Domino Server list (18 subscribers).
- IBM-KERM@CUVMB.BITNET — IBM mainframe KERMIT developers (89 subscribers).
- IBM-KERM@CUVMB.COLUMBIA.EDU – IBM mainframe KERMIT developers.
- IBM-MAIN@UA1VM.UA.EDU – IBM mainframe discussion list (1,761 subscribers).

- OPERS-L@VM1.CC.UAKRON.EDU – mainframe operations discussion list (330 subscribers).
- OS390-INSTALLATION@NOLA4.MCDERMOTT.COM – OS/390 installation (26 subscribers).
- UNT-MAINFRAME@UNT.EDU – academic mainframe research group (204 subscribers).
- VMY2K-L@MITVMA.MIT.EDU – IBM/VM Mainframe Y2K Q & A for MIT community (1 subscriber).

There's enough information here to decide which lists might be of interest, and each results page item links to additional information and interactive subscription interfaces.

The final Web site topic describes a rather technical product, LSMTP(tm), Internet e-mail delivery software for Windows NT and OpenVMS.

E-mail delivery (as opposed to e-mail reading/composition, performed by tools called mail user agents such as Eudora, Pegasus, and Pine) is a rather invisible function. As a host's e-mail volume increases, the need for a robust and efficient delivery vehicle – called mail transport agent – increases. This motivates replacing common MTAs such as sendmail with higher-capacity alternatives. In fact, L-Soft uses LSMTP to operate the largest LISTSERV site in the world, delivering over 3,000,000 e-mails daily on one host.

This Web site can be explored on many levels, with different goals. It offers insights into some very focused e-mail-related products and services, so it can enhance understanding and using them. It also provides tools for establishing and operating a key Internet resource, e-mail, in ways to further business or personal goals. Finally, it illustrates how an idea combined with energy and perseverance – not to mention applying my favourite fortune cookie saying that 'Nothing is impossible to a man who will not listen to reason' – can result in technical and industrial success.

Gabe Goldberg
Computers and Publishing (USA)

VM users can now benefit from Version 1.3 of Maintenance 2000, IBM's mainframe-based source code and JCL cross-reference analysis tool. Version 1.3 includes support for VisualAge COBOL MLE by generating data identification files for CCCA, improvements to its impact analysis and search function, and continuing support for PL/I, CA-Easytrieve Plus, and COBOL. The software integrates with CCCA for OS/390, MVS, and VM Version 2, which uses the data identification files generated by Version 1.3. It also integrates with either VisualAge COBOL MLEs for OS/390 and VM, or for MVS and VM.

Also new are program correlation chart enhancements, showing the programs having the specified system-ids and subsystem-ids, along with a cross-reference list enhancement, providing the resource information from a job point of view, and new search function enhancements including a dataset accepted as the input of retrieval conditions, and similar data item retrieval limited to data item declarations.

For further information contact your local IBM representative.

* * *

Software Diversified Services has announced the availability of its MultiTerm session manager for VM. MultiTerm provides multiple-session management, end-user support facilities, automatic keystroke recording and replay, and VM/CMS access for VTAM terminals.

SDS has also announced the availability of Multiprint, which provides print routing for VM and VM/VSE environments, allowing printer sharing and extended operator control. Multiprint/VM includes laser printer support and allows use of all laser printer capabilities without altering the programs that create printed output.

For further information contact:
Software Diversified Services, 5155 East River Road, Minneapolis, MN 55421-1025, USA.
Tel:(612) 571 9000.
URL: http://www.sdsusa.com.

* * *

VM users can now benefit from NCR's Teradata Parallel Data Pump (TPump) software, designed to allow data warehouse administrators to update information selectively in Teradata databases in real-time while operational queries are being run.

DBAs can set update rates for TPump manually or create a script that will vary the update load by time of day to accommodate database user activity. The software uses row hash locks, which allow multiple changes to one row to occur in a single job and supports concurrent updates on a single table. It's also got an automatic re-start feature, enabling an update to be stopped at any time and later resumed at the point of interruption.

Information can be maintained for real-time utilization in a Teradata database using TPump from any IBM-compatible mainframe running VM or MVS.

For further information contact:
NCR, 1700 South Patterson Boulevard, Dayton, OH 45479, USA.
Tel: (937) 445 5000.
URL: http://www.ncr.com.