December 2001

In this issue

3  Understanding the performance basics of Unix System Services
15  The effects of implementing snapshot copy
19  Sending e-mails from a mainframe
28  z/OS migration considerations
29  A C preprocessor
33  A REXX program to initialize DASD
48  Comprehensive dynamic allocation facilitation
68  Dynamic Channel-path Management
71  The z900 and z/OS
72  MVS news

© Xephon plc 2001
MVS Update

Published by
Xephon
27-35 London Road
Newbury
Berkshire RG14 1JL
England
Telephone: 01635 33598
From USA: 01144 1635 33598
E-mail: Jaimek@xephon.com

North American office
Xephon/QNA
PO Box 350100,
Westminster, CO 80035-0100
USA
Telephone: (303) 410 9344
Fax: (303) 438 0290

Editor
Jaime Kaminski

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 MVS Update, comprising twelve monthly issues, costs £340.00 in the UK; $505.00 in the USA and Canada; £346.00 in Europe; £352.00 in Australasia and Japan; and £350.00 elsewhere. In all cases the price includes postage. Individual issues, starting with the January 1992 issue, are available separately to subscribers for £29.00 ($43.50) each including postage.

Contributions
Articles published in MVS Update are paid for at the rate of £170 ($260) per 1000 words and £100 ($160) per 100 lines of code for the first 200 lines of original material. The remaining code is paid for at the rate of £50 ($80) per 100 lines. In addition, there is a flat fee of £30 ($50) per article. To find out more about contributing an article, you can download a copy of our Notes for Contributors from www.xephon.com/nfc.

MVS Update on-line
Code from MVS Update, and complete issues in Acrobat PDF format, can be downloaded from our Web site at http://www.xephon.com/mvs; you will need to supply a word from the printed issue.

© Xephon plc 2001. All rights reserved. None of the text in this publication may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, without the prior permission of the copyright owner. Subscribers are free to copy any code reproduced in this publication for use in their own installations, but may not sell such code or incorporate it in any commercial product. No part of this publication may be used for any form of advertising, sales promotion, or publicity without the written permission of the publisher. Copying permits are available from Xephon in the form of pressure-sensitive labels, for application to individual copies. A pack of 240 labels costs £36 (£24), giving a cost per copy of 15 cents (10 pence). To order, contact Xephon at any of the addresses above.

Printed in England.
Understanding the performance basics of Unix System Services

INTRODUCTION
This article considers the resource requirements of the USS environment. The areas which are considered include identifying resource consumption, OS/390 applications, client/server processes, physical resources, and controlling processes. The pertinent parameters are identified in each area, and the monitoring necessary to ensure that USS remains active and available to support your enterprise workload is described.

When you get right down to it, there is nothing new in this world. IBM’s Unix System Services (USS) for OS/390 may seem new and unique; however, one of the benefits of having lived through years of MVS and its related subsystems is that the general principles of performance management can be applied to the USS environment.

What about USS itself? Unix system administrators do not like it because it is not Unix. OS/390 systems programmers have enough trouble keeping up with new features and functions in their environment without having to learn USS. Because it is not MVS, it can appear cryptic and confusing due to a lack of systems tools. OS/390 systems are affected by it beginning at IPL time. If you are using IBM’s TCP/IP stack, it resides in the USS environment. Problems in USS, which affect the availability of resources, can lead to a high visibility event. If USS slows down, or falls over, the entire OS/390 dependent TCP/IP world will know about it. New application environments such as Web servers and three-tier client/server paradigms depend on TCP/IP access to legacy systems and databases as the source of their enterprise information.

For the benefit of the intrepid adventurers who have been tasked with responsibility for the USS environment, I have compiled a series of steps which highlight the options you can set so that USS gives you reasonable performance for a reasonable chunk of your MVS processing resources.
The recommendations contained herein are correct to the best of my experience. These recommendations have been tested at numerous field sites, but as always, before you change anything in a production environment, you should test the settings for their suitability within your own enterprise systems.

THE PERSONALITY OF USS

Let’s start with the processing parameters that USS uses when it initializes during the IPL of an OS/390 system. These parameters can be found in the dataset SYS1.PARMLIB (or your site’s choice of concatenated PARMLIB file) and the member name will be BPXPRM.xx (where the xx is specified on the OMVS parameter statement in SYS1.PARMLIB(IEASYS00). BPXPRM00 is the default used, if no other suffix is specified. The BPXPRM00 member may look something like this:

```plaintext
IPCSEMNSEMS(50)  
IPCSHMMPAGES(2048)  
MAXFILEPROC(256)  
MAXPROCUSER(25)  
MAXASIZE(200000000)  
MAXPID(500)  
MAXTHREATAKS(500)  
MAXUIDS(200)  
CTRACE(CTIBPX00)  
STEPLIBLIST('/system/steplib')  
USERIDALIATABLE('/etc/aliastable')  
FILESYSTYPE TYPE(HFS)  
ENTRYPOINT(GFUAINIT)  
ROOT FILESYSTEM('OMVS.S390R9.&SYSNAME..ROOT')  
TYPE(HFS)  
MODE(RDWR)  
MOUNT FILESYSTEM('OMVS.DEV.UCD-SNMP.NFS')  
MOUNTPOINT('/ucd-snmp')  
TYPE(HFS)  
MODE(READ)  
FILESYSTYPE TYPE(UDS)  
ENTRYPOINT(BPXUINTEGER)  
NETWORK DOMAINNAME(AF_UNIX)  
DOMAINNUMBER(1)  
MAXSOCKETS(2000)  
TYPE(UDS)
```
There are approximately 55 parameters that can be assigned in the BPXPRM00 file. Not all of them are appropriate to this discussion. For basic performance purposes, we will concentrate on some of the following 15 entries:

* BPXPRMxx Parameter Description
* Parameter
* MAXUIDS Maximum concurrent users in system
* MAXSHAREPAGES Maximum concurrent shared-storage pages in use
* MAXPROCUSER Maximum concurrent processes for one user
* MAXPROCSYS Maximum concurrent processes in system
* MAXMMAPAREA Maximum data space pages for memory-mapped files
* IPCSHMPAGES Maximum number of pages for shared-memory segments
* IPCSHMSEG Maximum shared-memory segments per address space
* IPCSHMNIDS Maximum number of unique shared-memory segments
* IPCSHMMPAGES Maximum pages in a shared-memory segment
* IPCSEMNSEMS Maximum semaphores in any one semaphore set
* IPCSEMNOPS Maximum operations in any one semaphore call
* IPCSEMNIDS Maximum number of unique semaphore sets
* IPCMSGQMNUM Maximum messages in any one message queue
* IPCMSGQBYTES Maximum bytes in any one message queue
* IPCMSGNIDS Maximum number of unique message queues

Each parameter applicable to a specific topic will be considered at the point at which that topic is introduced.

**Identifying resource consumption (users)**

First, in the OS/390 region, you need to identify what address spaces are using USS facilities. Any MVS performance-monitoring package should be able to provide you with a list, which will be similar to the one below:

* SRM STATS: CPU= 10.93 UIC=254 TPR= 0
* DISPLAY: U (*.B.T,S,C,D,I,U,E,A,Y,M) SCREEN: 4
* SORT : CC (JN,CC) _ ASCENDING X DESCENDING
<table>
<thead>
<tr>
<th>Jobname</th>
<th>CICS</th>
<th>DB2</th>
<th>IMS</th>
<th>USS</th>
<th>APPC</th>
<th>TSO</th>
<th>ENCL</th>
<th>CPU%</th>
</tr>
</thead>
<tbody>
<tr>
<td>NASIØTCP</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td>1.37</td>
</tr>
<tr>
<td>TCPIP</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>.63</td>
</tr>
<tr>
<td>IMWEBsrv</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>.17</td>
</tr>
<tr>
<td>NASIØTUS</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td>.06</td>
</tr>
<tr>
<td>OSNMPD</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>.04</td>
</tr>
<tr>
<td>QEO1CHIN</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>.02</td>
</tr>
<tr>
<td>SYSLOGD4</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>.01</td>
</tr>
<tr>
<td>EDUCHUB</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td>.00</td>
</tr>
<tr>
<td>FTPD1</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td>.00</td>
</tr>
<tr>
<td>NASUHUB</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td>.00</td>
</tr>
<tr>
<td>NASTHUBR</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td>.00</td>
</tr>
<tr>
<td>NASTHUB</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td>.00</td>
</tr>
<tr>
<td>BPXOINIT</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td>.00</td>
</tr>
<tr>
<td>PORTMAP</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td>.00</td>
</tr>
<tr>
<td>CSN1F</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td>.00</td>
</tr>
<tr>
<td>SNMPQOE</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>.00</td>
</tr>
<tr>
<td>TUS1CLKR</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>.00</td>
</tr>
<tr>
<td>CSNJH</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>.00</td>
</tr>
<tr>
<td>NASUHUBP</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td>.00</td>
</tr>
<tr>
<td>FTPD5</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td>.00</td>
</tr>
<tr>
<td>INETD6</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td>.00</td>
</tr>
<tr>
<td>CSNJH1</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td>.00</td>
</tr>
<tr>
<td>NASUHUBR</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td>.00</td>
</tr>
<tr>
<td>NASTHUBR</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td>.00</td>
</tr>
<tr>
<td>TUS1CLKH</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td>.00</td>
</tr>
</tbody>
</table>

Notice that the 25 address spaces shown as using USS resources equate to a minimum of 25 users within USS. The number of processes from the OS/390 environment could actually be higher if any given user spawns multiple processes. This would be in addition to any other applications running within USS without a comparable OS/390 component. From the performance parameter list, the key parameter with which we are concerned would be MAXUIDS. Make sure that you have enough ‘users’ specified to include any workload that will come from OS/390, as well as that which comes from client/server requestors of USS resources.

Users, however, are not necessarily the most effective measure of work within USS. Once we know who is running in USS, we then need to know how those users, and their applications, run within the constraints of the USS subsystem.
Identifying resource consumption (physical resources)

Identifying USS processes, and understanding their impact on your organization will go a long way to making sure that your tenure, as USS systems programmer, will be long and trouble free.

**CPU**

When considering the physical resources that will be consumed within USS, CPU is one resource with no solid limit. As CPU requirements increase, OS/390 Workload Manager will start to manage all the processes and their CPU requirements. This can lead to a situation where, due to over commitment of resource, performance of all applications will degrade.

If TCP/IP is implemented using IBM’s stack, as stated above, it relies on USS to service certain types of request. When performance degradation occurs, TCP/IP will be affected along with the rest of the workload. This problem will become apparent to everyone with a TCP/IP connection to the mainframe. At that point, it is necessary to determine which processes are the heaviest consumers of CPU:

<table>
<thead>
<tr>
<th>Image</th>
<th>Process ID</th>
<th>Jobname</th>
<th>ASID</th>
<th>SYSCELL</th>
<th>System CPU</th>
<th>User CPU</th>
<th>CPU</th>
</tr>
</thead>
<tbody>
<tr>
<td>EDUC</td>
<td>10 TCPIP</td>
<td>0040</td>
<td>26,524</td>
<td>1:47.5900</td>
<td>5:22.7800</td>
<td></td>
<td></td>
</tr>
<tr>
<td>EDUC</td>
<td>8 TCPIP</td>
<td>0040</td>
<td>58</td>
<td>1:47.5900</td>
<td>5:22.7800</td>
<td></td>
<td></td>
</tr>
<tr>
<td>EDUC</td>
<td>9 TCPIP</td>
<td>0040</td>
<td>1,503</td>
<td>1:47.5900</td>
<td>5:22.7800</td>
<td></td>
<td></td>
</tr>
<tr>
<td>EDUC</td>
<td>5 TCPIP</td>
<td>0040</td>
<td>55</td>
<td>1:47.5900</td>
<td>5:22.7800</td>
<td></td>
<td></td>
</tr>
<tr>
<td>EDUC</td>
<td>6 TCPIP</td>
<td>0040</td>
<td>4</td>
<td>1:47.5900</td>
<td>5:22.7800</td>
<td></td>
<td></td>
</tr>
<tr>
<td>EDUC</td>
<td>12 TCPIP</td>
<td>0040</td>
<td>6,915</td>
<td>1:47.5900</td>
<td>5:22.7800</td>
<td></td>
<td></td>
</tr>
<tr>
<td>EDUC</td>
<td>17 OSNMPD</td>
<td>0068</td>
<td>27</td>
<td>1:11.2100</td>
<td>3:33.6500</td>
<td></td>
<td></td>
</tr>
<tr>
<td>EDUC</td>
<td>4 IMWEBSRV</td>
<td>001E</td>
<td>805,894</td>
<td>28.5000</td>
<td>1:25.5300</td>
<td></td>
<td></td>
</tr>
<tr>
<td>EDUC</td>
<td>16777237</td>
<td>NAS10TCP</td>
<td>0075</td>
<td>110,866</td>
<td>11.4000</td>
<td>34.2000</td>
<td></td>
</tr>
<tr>
<td>EDUC</td>
<td>13 SNOPOE</td>
<td>0069</td>
<td>77,420</td>
<td>6.5900</td>
<td>19.8000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>EDUC</td>
<td>83886154</td>
<td>NASTCP</td>
<td>02F5</td>
<td>39,837</td>
<td>5.2200</td>
<td>15.6600</td>
<td></td>
</tr>
<tr>
<td>EDUC</td>
<td>27 QEOICHIN</td>
<td>0070</td>
<td>5</td>
<td>4.4300</td>
<td>13.3000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>EDUC</td>
<td>32 QEOICHIN</td>
<td>0070</td>
<td>5</td>
<td>4.4300</td>
<td>13.3000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>EDUC</td>
<td>16777242</td>
<td>QEOICHIN</td>
<td>0070</td>
<td>3</td>
<td>4.4300</td>
<td>13.3000</td>
<td></td>
</tr>
<tr>
<td>EDUC</td>
<td>33 QEOICHIN</td>
<td>0070</td>
<td>5</td>
<td>4.4300</td>
<td>13.3000</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
As you can see from the list above, individual users of USS may have multiple instances of active processes that allow them to utilize system resources more effectively. There are three methods of controlling CPU usage within an instance of USS:

- Limit the total number of processes within USS.
- Limit the number of concurrent processes that any given user can spawn.
- Control the amount of CPU that any given application requires to complete its processing.

The third option is obviously something that everyone would like to have and is actually available through the MAXCPUTIME parameter. However, when used as a resource limit, this parameter will affect all processes, not just discretionary workloads. There will be high usage applications in any environment, with a host of users who absolutely must have that function available for their continued survival. So, we move on to the two things we can more easily control.

The parameter that controls how many processes a system can support is MAXPROCSYS. Once you know what applications are requesting resources from the OS/390 side, and you have determined what processes are required for any three-tier applications, you have a base number to use in calculating the value of MAXPROCSYS. Remember, however, the maximum number of processes in any USS instance will also depend on how many sub-processes can be spawned by existing users.

The parameter that controls how many processes each user can spawn is MAXPROCUSER. Multiplying the number of multi-threaded applications by the MAXPROCUSER value, and adding that to the base value of MAXPROCSYS, as calculated above, will give you a fair estimation of the final value that MAXPROCSYS should have.

**Memory**

The next resource that needs to be considered is the amount of MEMORY used within USS. Memory will be dynamically allocated, to meet the requirements of application processing. These memory
requests are serviced by the OS/390 environment, and when excessive memory requests are made the impact will be visible across all applications within the subsystem. In order to determine who the memory hogs on the system are, your USS monitor should be able to provide a list of processes, sorted by MEMORY usage in descending order:

<table>
<thead>
<tr>
<th>Image</th>
<th>Process ID</th>
<th>Jobname</th>
<th>Memory Used (K)</th>
</tr>
</thead>
<tbody>
<tr>
<td>EDUC</td>
<td>10</td>
<td>TCPIP</td>
<td>12492</td>
</tr>
<tr>
<td>EDUC</td>
<td>27</td>
<td>QEQ1CHIN</td>
<td>8524</td>
</tr>
<tr>
<td>EDUC</td>
<td>16777242</td>
<td>QEQ1CHIN</td>
<td>8524</td>
</tr>
<tr>
<td>EDUC</td>
<td>29</td>
<td>QEQ1CHIN</td>
<td>8524</td>
</tr>
<tr>
<td>EDUC</td>
<td>30</td>
<td>QEQ1CHIN</td>
<td>8524</td>
</tr>
<tr>
<td>EDUC</td>
<td>32</td>
<td>QEQ1CHIN</td>
<td>8524</td>
</tr>
<tr>
<td>EDUC</td>
<td>33</td>
<td>QEQ1CHIN</td>
<td>8524</td>
</tr>
<tr>
<td>EDUC</td>
<td>35</td>
<td>QEQ1CHIN</td>
<td>8524</td>
</tr>
<tr>
<td>EDUC</td>
<td>36</td>
<td>QEQ1CHIN</td>
<td>8524</td>
</tr>
<tr>
<td>EDUC</td>
<td>34</td>
<td>QEQ1CHIN</td>
<td>8524</td>
</tr>
<tr>
<td>EDUC</td>
<td>31</td>
<td>QEQ1CHIN</td>
<td>8524</td>
</tr>
<tr>
<td>EDUC</td>
<td>37</td>
<td>QEQ1CHIN</td>
<td>8524</td>
</tr>
<tr>
<td>EDUC</td>
<td>33554505</td>
<td>TUS1CLKH</td>
<td>8308</td>
</tr>
<tr>
<td>EDUC</td>
<td>33554504</td>
<td>TUS1CLKH</td>
<td>8308</td>
</tr>
<tr>
<td>EDUC</td>
<td>33554503</td>
<td>TUS1CLKH</td>
<td>8308</td>
</tr>
</tbody>
</table>

As in the previous CPU discussion, you need to be aware of how many processes are running in USS, and you need to be able to determine the overall impact of any given application, by knowing the number of processes that can be spawned from a parent process.

**I/O**

In most applications, I/O activity inversely correlates to performance. If performance of the USS workload requires favoured status, you can fix memory within USS to facilitate HFS I/O processing. It is important to know how much memory is being used overall, and how much fixed memory has been requested:

<table>
<thead>
<tr>
<th>Image</th>
<th>Virtual</th>
<th>Virtual</th>
<th>Virtual</th>
<th>Fixed</th>
<th>Fixed</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Max (MB)</td>
<td>Used (MB)</td>
<td>Used %</td>
<td>Max (MB)</td>
<td>Used (MB)</td>
</tr>
<tr>
<td>EDUC</td>
<td>75.000</td>
<td>13.992</td>
<td>18.66</td>
<td>0.000</td>
<td>0.000</td>
</tr>
</tbody>
</table>

Without fixed memory, HFS I/O path length may increase based on the other memory requirements along with that required by the USS I/O function.

The final piece in the physical resource usage puzzle is the amount of I/O that will occur within a given process. When working with files, USS stores its HFS files in the OS/390 side. The file type of HFS can identify the files:

```
<table>
<thead>
<tr>
<th>Command - Enter &quot;/&quot; to select action</th>
<th>Dsorg</th>
<th>Recfm</th>
<th>Lrecl</th>
<th>Blksz</th>
</tr>
</thead>
<tbody>
<tr>
<td>OMVS.S390R9.EDUC.ROOT</td>
<td>HFS</td>
<td>U</td>
<td>Ø</td>
<td>Ø</td>
</tr>
<tr>
<td>OMVS.S390R9.EDUC.UKMRW1.HFS</td>
<td>HFS</td>
<td>U</td>
<td>Ø</td>
<td>Ø</td>
</tr>
<tr>
<td>OMVS.S390R9.PROD.ROOT</td>
<td>HFS</td>
<td>U</td>
<td>Ø</td>
<td>Ø</td>
</tr>
<tr>
<td>OMVS.S390R9.ROOT</td>
<td>PO</td>
<td>?</td>
<td>Ø</td>
<td>Ø</td>
</tr>
<tr>
<td>OMVS.S390R9.SYSA.CSABC1.HFS</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>OMVS.S390R9.SYSA.CSBXS1.HFS</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>OMVS.S390R9.SYSA.CSDLM1.HFS</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

I/O for USS may begin in the Kernel address space; however, systems services will include processing satisfied by OS/390:

```
JOBNAME: IMWEBSRV STENAME: IMWEBSRV PROCSTEP: WEBSRV

<table>
<thead>
<tr>
<th>DDNAME</th>
<th>DATA SET NAME</th>
<th>VOLUME</th>
<th>IO</th>
<th>CNT</th>
<th>RFM</th>
<th>LRECL</th>
</tr>
</thead>
<tbody>
<tr>
<td>FIRST LOCATED UNDER TCB: IMWHTTPD</td>
<td>S3909R</td>
<td>308</td>
<td>U</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>STEPLIB</td>
<td>CEE.SCEERUN</td>
<td>S3909R</td>
<td>Ø</td>
<td></td>
<td>Ø</td>
<td></td>
</tr>
<tr>
<td>32760</td>
<td>CBC.SCLBDLL</td>
<td>S3909R</td>
<td>Ø</td>
<td></td>
<td>Ø</td>
<td></td>
</tr>
<tr>
<td>SYSIN</td>
<td>NULLFILE</td>
<td>DUMMY</td>
<td>N/A</td>
<td>U</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6144</td>
<td>SYSPRINT</td>
<td></td>
<td></td>
<td>N/A</td>
<td>V</td>
<td>137</td>
</tr>
<tr>
<td>882</td>
<td>SYSERR</td>
<td></td>
<td></td>
<td>N/A</td>
<td></td>
<td></td>
</tr>
<tr>
<td>STDOUT</td>
<td>WEBsrv.IMWEBSRV.STC01975.D0000101.?</td>
<td></td>
<td></td>
<td>N/A</td>
<td>F</td>
<td>121</td>
</tr>
<tr>
<td>STDERR</td>
<td>WEBsrv.IMWEBSRV.STC01975.D0000103.?</td>
<td></td>
<td></td>
<td>N/A</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SYSOUT</td>
<td>WEBsrv.IMWEBSRV.STC01975.D0000105.?</td>
<td></td>
<td></td>
<td>N/A</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12100</td>
<td>CEEDUMP</td>
<td></td>
<td></td>
<td>N/A</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

Any OS/390 I/O mitigation technique you wish to apply to the HFS datasets will be helpful. Additionally, it is a good idea to minimize the amount of data sharing that occurs between USS and OS/390-based
applications. Unix I/O is stream I/O. This means that every record being transmitted to, or received from, an HFS dataset is sent or received byte by byte. OS/390, on the other hand, is block I/O oriented. If you try to mix the two I/O types, it is quite possible to slow down OS/390 I/O to the rate of the USS system. My personal recommendation is that you isolate the HFS datasets as much as possible, so that they contend only with each other. If that is not an option, then consider placing HFS datasets on packs that contain infrequently referenced OS/390 datasets.

For further ideas in tuning HFS I/O, please see the article Monitoring HFS Performance with DFSMS 1.5 by Clark Kidd. You can receive a copy of this paper by registering with Landmark Systems Corporation at the following URL: http://www.landmark.com/offers/homepage/tuss/register.cfm.

**Controlling processes**

As shown above, the performance of the USS subsystem will depend on what is running, and what resources are available to the existing workload. Over-committing resources will result in an impact that will be felt throughout the entire workload.

If it becomes necessary to manage the processes within the USS workload, you will need the ability to determine what processes are active, and which step in their execution they are currently processing. By sorting the list your monitor provides to you by process ‘STATE’, you can tell when interrupting a process will cause the least amount of damage to the application that spawned it:

<table>
<thead>
<tr>
<th>Image</th>
<th>Process ID</th>
<th>Jobname</th>
<th>State</th>
<th>Proc Actv</th>
<th>Stat</th>
<th>Flag</th>
<th>Flag</th>
<th>Mult</th>
<th>PTrc</th>
<th>Thrd</th>
<th>Swap</th>
<th>Stop</th>
<th>Idle Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>EDUC</td>
<td>1</td>
<td>BPXOINIT</td>
<td>FILE</td>
<td>N</td>
<td>N</td>
<td>M</td>
<td>Y</td>
<td>N</td>
<td>19:28:46.23</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EDUC</td>
<td>16777218</td>
<td>FTPD1</td>
<td>FILE</td>
<td>N</td>
<td>N</td>
<td>1</td>
<td>Y</td>
<td>N</td>
<td>3:18:49.610</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EDUC</td>
<td>50331651</td>
<td>FTPD5</td>
<td>FILE</td>
<td>N</td>
<td>N</td>
<td>1</td>
<td>Y</td>
<td>N</td>
<td>19:29:22.21</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EDUC</td>
<td>4</td>
<td>IMWEBSRV</td>
<td>OTH</td>
<td>N</td>
<td>N</td>
<td>P</td>
<td>N</td>
<td>19:33:36.20</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EDUC</td>
<td>5</td>
<td>IMWEBSRV</td>
<td>OTH</td>
<td>N</td>
<td>N</td>
<td>P</td>
<td>N</td>
<td>19:33:36.20</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EDUC</td>
<td>6</td>
<td>TCPIP</td>
<td>RUNNING</td>
<td>Y</td>
<td>N</td>
<td>M</td>
<td>N</td>
<td>N</td>
<td>0.0000</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EDUC</td>
<td>8</td>
<td>TCPIP</td>
<td>RUNNING</td>
<td>Y</td>
<td>N</td>
<td>1</td>
<td>N</td>
<td>N</td>
<td>0.0000</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EDUC</td>
<td>10</td>
<td>TCPIP</td>
<td>FILE</td>
<td>Y</td>
<td>N</td>
<td>1</td>
<td>N</td>
<td>N</td>
<td>2:40:74.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EDUC</td>
<td>33554443</td>
<td>NASUHBP</td>
<td>RUNNING</td>
<td>Y</td>
<td>N</td>
<td>1</td>
<td>N</td>
<td>N</td>
<td>14:6800</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EDUC</td>
<td>12</td>
<td>TCPIP</td>
<td>RUNNING</td>
<td>Y</td>
<td>N</td>
<td>M</td>
<td>N</td>
<td>N</td>
<td>0.0000</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

When considering the ‘STATE’ of a process, be aware of any processes in the STATE of ‘ZOMBIE’. A ZOMBIE process is one that has stopped responding to the Unix environment, but which is still consuming all of the various resources it acquired during its execution. Just like the creatures in horror movies, ZOMBIE processes can drain the life out of a USS system. If you find any processes in a ‘ZOMBIE’ state, cancelling these will allow you to release their resources while maintaining the integrity of the USS system.

Before you terminate a process, you need to know how to restart that process. This requires the knowledge of what command started the task, including the flags that have been set upon command execution:

<table>
<thead>
<tr>
<th>Image</th>
<th>Process ID</th>
<th>Jobname</th>
<th>Command That Started Process</th>
</tr>
</thead>
<tbody>
<tr>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>EDUC</td>
<td>33</td>
<td>QEQ1CHIN</td>
<td>CSQXDISP</td>
</tr>
<tr>
<td>EDUC</td>
<td>34</td>
<td>QEQ1CHIN</td>
<td>CSQXDISP</td>
</tr>
<tr>
<td>EDUC</td>
<td>35</td>
<td>QEQ1CHIN</td>
<td>CSQXDISP</td>
</tr>
<tr>
<td>EDUC</td>
<td>36</td>
<td>QEQ1CHIN</td>
<td>CSQXRCTL</td>
</tr>
<tr>
<td>EDUC</td>
<td>37</td>
<td>QEQ1CHIN</td>
<td>CSQXTNSV</td>
</tr>
<tr>
<td>EDUC</td>
<td>16777254</td>
<td>NASUHUBP</td>
<td>TUSSKERN</td>
</tr>
<tr>
<td>EDUC</td>
<td>39</td>
<td>INETD6</td>
<td>/usr/sbin/inetd -f -a</td>
</tr>
<tr>
<td>EDUC</td>
<td>50331688</td>
<td>NASUHUBP</td>
<td>TUSSREQP</td>
</tr>
<tr>
<td>EDUC</td>
<td>67108905</td>
<td>CSNJH1</td>
<td>/bin/sh</td>
</tr>
<tr>
<td>EDUC</td>
<td>43</td>
<td>CSNJH1</td>
<td>OMVS</td>
</tr>
<tr>
<td>EDUC</td>
<td>44</td>
<td>CSNJH1</td>
<td>sh</td>
</tr>
<tr>
<td>EDUC</td>
<td>49</td>
<td>CSNJH</td>
<td>/usr/sbin/ftpdns</td>
</tr>
<tr>
<td>EDUC</td>
<td>16777284</td>
<td>TUS1CLKR</td>
<td>THPHUBM</td>
</tr>
<tr>
<td>EDUC</td>
<td>16777285</td>
<td>TUS1CLKR</td>
<td>THPTCPL</td>
</tr>
<tr>
<td>EDUC</td>
<td>16777286</td>
<td>TUS1CLKH</td>
<td>THPHUBM</td>
</tr>
<tr>
<td>EDUC</td>
<td>33554503</td>
<td>TUS1CLKH</td>
<td>TUSSCTRL</td>
</tr>
</tbody>
</table>

Once you have the command syntax, you may also need to know the path that a program executes from:

<table>
<thead>
<tr>
<th>Image</th>
<th>Process ID</th>
<th>Jobname</th>
<th>Path Name Used by Process</th>
</tr>
</thead>
<tbody>
<tr>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>EDUC</td>
<td>33</td>
<td>QEQ1CHIN</td>
<td>CSQXDISP</td>
</tr>
<tr>
<td>EDUC</td>
<td>34</td>
<td>QEQ1CHIN</td>
<td>CSQXDISP</td>
</tr>
<tr>
<td>EDUC</td>
<td>35</td>
<td>QEQ1CHIN</td>
<td>CSQXDISP</td>
</tr>
<tr>
<td>EDUC</td>
<td>36</td>
<td>QEQ1CHIN</td>
<td>CSQXRCTL</td>
</tr>
<tr>
<td>EDUC</td>
<td>37</td>
<td>QEQ1CHIN</td>
<td>CSQXTNSV</td>
</tr>
<tr>
<td>EDUC</td>
<td>16777254</td>
<td>NASUHUBP</td>
<td>TUSSKERN</td>
</tr>
<tr>
<td>EDUC</td>
<td>39</td>
<td>INETD6</td>
<td>/usr/sbin/inetd</td>
</tr>
</tbody>
</table>
Once you know how and from where the process can be restarted, you need to assess how you will terminate the process:

**COMMAND:**

1. Show Threads in Process
2. Show Threads in Jobname
3. TMVS Job I/O Analysis
4. TMVS Job Wait Analysis
5. TMVS Job Monitor
6. Kill this Process (Notify)
7. Kill this Process (Force)
8. Cancel this Jobname

If an application spawns multiple unrelated processes, you can terminate a process in one of two ways:

- Cancel a process with NOTIFY (clean shutdown)
- Cancel a process with FORCE (KILL mode).

If you shutdown a process cleanly, it will signal the application that started the process, indicating that a shutdown has occurred. This will allow application recovery processing to ensure that no data is lost. Before you roll an application into production, you need to make sure that the application architects have planned for process recovery upon detection of a manual or abnormal shutdown of a spawned process.

If a process will not end with a KILL NOTIFY, you may then need to issue the FORCE mode KILL to end a process. This will require manual intervention to ensure that data is not lost, and that the process is restarted in the appropriate manner. As stated before, you will want to know that applications architects have planned for processes to be
KILLED, and understand their requirements for data and processing recovery.

Because all processes run as part of an OS/390 address space, it is worth considering whether or not to KILL processes, or whether to cancel the associated OS/390 address space itself and allow OS/390 recovery to handle data and processing integrity issues. Too often people have killed processes, only to eventually have to terminate the OS/390 application itself, and usually after a number of data integrity issues have occurred, which must then be manually recovered. Once again, the application architects will know whether individual processes can be safely terminated, or whether it would be better to terminate an entire application suite and allow outside procedures to ensure data recovery.

CONCLUSIONS

Because it is not entirely Unix, and not entirely OS/390, USS can be a daunting addition to our normal workload. Having a monitor to consolidate information into a familiar interface can simplify the task of managing USS. There are three monitors on the market today, which can help you understand and manage the USS processing your site will undertake. So, choose the monitor with which you are most familiar, and which provides the best mix of functionality and depth.

With IBM’s implementation of several critical system tasks that depend on USS resources, the visibility of problems in this new environment will rapidly spread within an organization. It will become imperative to know when problems occur, and to be seen to be solving them before the phone calls start.

Remember, USS is implemented as part of the OS/390 system. So resources utilized by USS processes will also count towards Service Units charged against IBM’s new Variable Workload License Charge (VWLC). Like any other subsystem, if resources are consumed by run-away tasks, this can lead to automatic upgrade charges at inopportune times.

Aaron Cain  
Staff Analyst, Landmark Systems (UK)  
© Xephon 2001
The effects of implementing snapshot copy

Batch is still the largest consumer of system resources in many installations because batch is inherently less expensive than other types of data processing (because of its lower overhead). Batch has always been cheaper and it is expected to continue to be so. Therefore you can always expect to have a significant batch workload, despite the fact that many batch applications are being ported (or moved) to on-line systems.

Batch is used to update, back-up, and report from large databases of corporate data centres. This kind of batch work exists along with the on-line systems. It must do its job with no or minimal impact or disruption of on-line applications. In the past decade, a noticeable improvement has been observed in the area of data back-up and database maintenance. Numerous utilities are now available that can dramatically enhance standard on-line system operations.

Batch work is largely confined to certain periods (windows) when on-line systems are not being heavily used. Since many installations are moving toward continuous operations (24x7), on-line applications must stay up and running as long as possible and thus the batch windows will shrink in order to lessen their impact on on-line systems. There are several methods and approaches available for cutting down the batch window. The methods usually advocated by the hardware and software vendors include the acquisition of more powerful processors, more storage, new I/O subsystems, new software tools, etc. However, the most effective way to cut down the batch window is to optimize I/O.

In the past, if batch jobs were taking longer to complete, it was usually because of insufficient resources (memory, I/O, CPU, and so forth). In addition to that, poor batch service was usually connected with JCL errors, badly-tuned VSAM files, dataset contention, and inefficient use of resources and tools available. However, poor design of the batch job, or job stream, can be the culprit.

The remainder of this article focuses on results achieved by a sample tuning exercise. Prompted by the increasing number of data checks on tape drives (which increased the cost of hardware maintenance
considerably) as well as operators’ pleas to “do something to free us from tape mounts”. The production batch streams were scanned to ascertain whether tapes could be eliminated.

The review revealed that jobs submitted by the production department frequently required a tape to be allocated, thus causing increases in the jobs elapsed time. Unfortunately, SMF does not record any duration for tape mount time (waiting for the operator to mount the tape) but provides only a count of mounts for each job step. This screening of production jobs also clearly showed that many tape datasets were in fact (unreliable) back-up copies taken before or after the execution of the main job streams. By analysing tape use, it was discovered that it was possible to reduce unnecessary demand on tapes, since it was deemed that this way of using the back-ups was inappropriate. Therefore, all of the steps that contained the UNIT=TAPE (or similar) were redesigned to use an on-the-spot snapshot copy of the desired dataset taken immediately preceding this step. Quite naturally, it was left to DFHSM to manage these snapshot copies.

The sample set of reports shown below is a summary of what was achieved by eliminating tapes from production batch. By comparing two weeks of SMF job-related records covering the same period (18 – 31 October) from the last two years, we noticed a total workload increase of 33.06%. The workload mix has changed a little: production

<table>
<thead>
<tr>
<th>Batch jobs %</th>
<th>STC %</th>
<th>TSO %</th>
</tr>
</thead>
<tbody>
<tr>
<td>Year 2000</td>
<td>70.21</td>
<td>18.80</td>
</tr>
<tr>
<td>Year 2001</td>
<td>69.63</td>
<td>23.30</td>
</tr>
</tbody>
</table>

Year 2000 batch mix: 52.02% production + 47.98% test

Year 2001 batch mix: 57.86% production + 42.14% test

Figure 1: The workload mix
jobs have increased while TSO usage has decreased. It should be noted that the increase in production batch was due to implementing completely new applications into production and the redesign of production job streams had no any effect in increasing job percentage. Figure 1 provides a summary of the workload mix. A summary of basic indicators of production batch jobs is given in Figure 2.

![Table](data:image/png;base64,iVBORw0KGgoAAAANSUhEUgAAAgAAAAAQA...)

**Figure 2: Overall review for production batch jobs only**

Figure 3 provides a summary of the basic statistical indicators for batch jobs and production jobs. The key point is that the average elapsed time for production batch dropped by 31.68% while the total number of production batch jobs went up by 46.79%.

![Table](data:image/png;base64,iVBORw0KGgoAAAANSUhEUgAAAgAAAAAQA...)

**Figure 3: Basic batch statistics in 2000 compared to 2001**
The tape mounts were tabulated, as shown in Figure 4.

<table>
<thead>
<tr>
<th></th>
<th>Jobs not using tapes</th>
<th>Jobs not tapes</th>
<th>TOTAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>2000</td>
<td>2985 / 70.04</td>
<td>1277 / 29.96</td>
<td>4262</td>
</tr>
<tr>
<td>2001</td>
<td>5496 / 87.85</td>
<td>760 / 12.15</td>
<td>6256</td>
</tr>
</tbody>
</table>

*Figure 4: Tape mounts for production jobs by group*

This table clearly shows that tape mounts were dramatically cut down. In the year 2000 approximately every third production job requested the tape to be mounted; this year only every eighth production job uses the tape.

If we turn our attention to the elapsed time of the jobs (shown in Figure 5) we can spot little change in distribution. Some would say, “What was the effect of eliminating the tapes from production jobs except in doing a favour for the operators?”

<table>
<thead>
<tr>
<th></th>
<th>UP TO 5’</th>
<th>UP TO 10’</th>
<th>UP TO 20’</th>
<th>UP TO 45’</th>
<th>&gt; 45’</th>
<th>TOTAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>2000</td>
<td>3649 / 85.62%</td>
<td>262 / 6.15%</td>
<td>159 / 3.75%</td>
<td>130 / 3.05%</td>
<td>62 / 1.45%</td>
<td>4262</td>
</tr>
<tr>
<td>2001</td>
<td>5536 / 88.49%</td>
<td>354 / 5.66%</td>
<td>178 / 2.85%</td>
<td>166 / 2.65%</td>
<td>22 / 0.3%</td>
<td>6256</td>
</tr>
</tbody>
</table>

*Figure 5: Duration of production jobs*

The answer to this question is straightforward and can be seen in Figure 6, which shows the duration of production jobs after normalization to Year 2000 data.
This table clearly shows that the number of production batch jobs completing within a timeframe of five minutes increased by 51% while the number of long-running batch jobs (with a duration of more than 45 minutes) was reduced by 65% compared with the previous year.


Mile D Pekic  
Systems Programmer  
Postal Savings Bank (Yugoslavia)  
© Xephon 2001

### Sending e-mails from a mainframe

The use of e-mail is ubiquitous within the office environment. In this article we will consider how e-mails can be used to alert sysprogs about the status of programs or jobs. There are many situations where we need to be alerted when something happens on our system. For example:

- **System jobs**: we can be alerted to possible problems, such as insufficient free space on VSAM datasets, bad response times for tasks, etc.

- **Application programs**: it can be very helpful if an application programmer can follow sudden changes, like the volume of input datasets, or the presence of errors, etc.
- **On-line transactions**: it is useful to know when on-line transactions abend.

There are several ways to send e-mail from a mainframe:

- Using IEBGENER
- Using IDCAMS and REPRO
- Using SMTPNOTE as a stand-alone command or in an application program.

The general idea is to change our JCL, to check the return code after every significant step, and, if it is required, send an e-mail:

```plaintext
//APPLPGM JOB MSGCLASS=Z,CLASS=B,NOTIFY=&SYSUID
/****************************************************************************
** SENDING E-MAILS FROM JCL IN CASE OF ERROR IN PREVIOUS STEP
****************************************************************************
//STEPGM EXEC PGM=PGMNAME,REGION=4800K
//STEPLIB DD DSN=APPHLQ.PROD.LOADLIB,DISP=SHR
//INPUT1 DD DSN=APPHLQ.INPUT1.FILE,DISP=SHR
//INPUT2 DD DSN=APPHLQ.INPUT2.FILE,DISP=SHR
//ERRORS DD DSN=APPHLQ.ERRORS.FILE,DISP=SHR
/*
//IFSTEP IF (STEPGM.RC>0) THEN

// SENDMAIL step goes here

//IFSTEP ENDIF
/*
```

**USING IEBGENER**

If you need to send e-mails as quickly as possible, you can use IEBGENER. It is good for sending short messages because you can write it on the spot. For example:

```plaintext
//SENDMAIL JOB MSGCLASS=Z,CLASS=B,NOTIFY=&SYSUID
/****************************************************************************
** SENDING E-MAIL FROM HOST USING IEBGENER
****************************************************************************
//SENDMAIL EXEC PGM=IEBGENER
//SYSUT1 DD *
HELO SMTP_SERVER
MAIL FROM:<senderid@company.com>
RCPT TO: <emailid1@client1.com>
RCPT TO: <emailid2@client2.com>
DATA
SUBJECT: MESSAGE TITLE
```
You can divide your message into two parts, message header and message body, and put each part in a separate dataset. For example, you can have SYSUT1 defined as the concatenation of two files:

```
//SENDMAIL JOB MSGCLASS=Z,CLASS=B,NOTIFY=&SYSUID
//**************************************************
//  * SENDING MAIL FROM HOST USING IEBGENER
//  **************************************************
//SENDMAIL EXEC PGM=IEBGENER
//SYSUT1 DD DSN=APPHLQ.HEADER.MSG,DISP=SHR
//SYSUT2 DD DSN=APPHLQ.BODY.MSG,DISP=SHR
//SYSUT2 DD SYSOUT=(B,SMTP)
//SYSPRINT DD SYSOUT=*  
//SYSIN DD DUMMY
//*

APPHLQ.HEADER.MSG has a description of who is the sender, who are recipients, and the subject of the message. If we need to send e-mails to some addresses frequently, we can create a PDS library and members with predefined descriptions, like:

```
USERID.HEADERS.MSG(CLIENT1)
APPHLQ.BODY.MSG has the message itself.

In both cases the description of SYSUT2 in IEBGENER utility is:
```
//SYSUT2 DD SYSOUT=(B,SMTP)
```

where: B =SYSOUT class;SMTP= SMTP address space name for external writer.

The only requirement is that the SMTP task is running under your TSO session. The weakness of IEBGENER is that we need to use as many JCL steps as we have different messages to send. It is not possible to send several different messages at once, using the same IEBGENER utility.

**USING IDCAMS AND REPRO**

Another way to send e-mail is to use IDCAMS and the REPRO command:
SENDMAIL JOB MSGCLASS=Z,CLASS=B,NOTIFY=&SYSUID

(StringUtils)

/* SENDING ONE MAIL USING IDCAMS UTILITY
(String)

STEP20 EXEC PGM=IDCAMS
(Paragraph)
//MAIL DD DSN=APPHLQ.HEADER.MSG,DISP=OLD
// DD DSN=APPHLQ.BODY.MSG,DISP=OLD
//OUT DD SYSOUT=(B,SMTP)
//SYSPRINT DD SYSOUT=* 
//SYSIN DD *
(Paragraph)
  REPRO INFILE(MAIL) OUTFILE(OUT)
 */

APPHLQ.HEADER.MSG and APPHLQ.BODY.MSG are the same as in the previous example for the IEBGENER utility. This approach is better if you have to send many different e-mails at once. This can be done as follows:

SENDMAIL JOB MSGCLASS=Z,CLASS=B,NOTIFY=&SYSUID

(Sprite)
/* SENDING MULTIPLE MAILS USING IDCAMS UTILITY
(Sprite)

STEP20 EXEC PGM=IDCAMS
(Sprite)
//MAIL1 DD DSN=APPHLQ.HEADER1.MSG,DISP=OLD
// DD DSN=APPHLQ.BODY1.MSG,DISP=OLD
//MAIL2 DD DSN=APPHLQ.HEADER2.MSG,DISP=OLD
// DD DSN=APPHLQ.BODY2.MSG,DISP=OLD
//MAIL3 DD DSN=APPHLQ.HEADER3.MSG,DISP=OLD
// DD DSN=APPHLQ.BODY3.MSG,DISP=OLD
//MAIL4 DD DSN=APPHLQ.HEADER4.MSG,DISP=OLD
// DD DSN=APPHLQ.BODY4.MSG,DISP=OLD
//OUT1 DD SYSOUT=(B,SMTP)
//OUT2 DD SYSOUT=(B,SMTP)
//OUT3 DD SYSOUT=(B,SMTP)
//OUT4 DD SYSOUT=(B,SMTP)
//SYSPRINT DD SYSOUT=* 
//SYSIN DD *
(Sprite)
  REPRO INFILE(MAIL1) OUTFILE(OUT1)
  REPRO INFILE(MAIL2) OUTFILE(OUT2)
  REPRO INFILE(MAIL3) OUTFILE(OUT3)
  REPRO INFILE(MAIL4) OUTFILE(OUT4)
 */

USING SMTPNOTE

Another way to send e-mail is using the SMTPNOTE REXX EXEC, which resides in one of SYS1 libraries: SYS1.OEM.CLIST. SMTPNOTE has parameters, like:
• FROM – e-mail sender
• TO – e-mail recipient
• SU – title of the message (subject)
• DA – data (the name of dataset with message body)
• CC – carbon-copy recipient
• BATCH – signal to REXX to set an error rather than to prompt for any missing parameters and looks like:

SMTPNOTE FROM(SENDERID@COMPANY.COM) TO(EMAILID@CLIENT.COM)
SU('Title of the message') DA('APPHLQ.SMTP.MESSAGE')

Unfortunately, SMTPNOTE needs the message dataset to be completely free, so no other job or user may have it allocated. Not even SHR use is acceptable. This may require a front-end process that copies the input data to another temporary file that then goes into SMTPNOTE. SMTPNOTE can be used in two ways, as standalone REXX in JCLs and as part of application program, where we can send e-mails directly from the program.

SMTPNOTE used as standalone
The JCL for using SMTPNOTE stand-alone would look like:

```
//
// previous steps
//
//IFSTEP   IF (LASTSTEP.RC>Ø) THEN
//TONOTE   EXEC PGM=IKJEFTØ1
//SYSPROC  DD DSN= SYS1.OEM.CLIST,DISP=SHR
//SYSTSPRT DD SYSOUT=*  
//SYSTSIN  DD *  
PROFILE NOPREFIX  
%SMTMPNOTE TO(EMAILID@CLIENT.COM) -  
SU('*** STEPNAME - COND>Ø ***') DA('APPHLQ.ERRORS.FILE')
//IFSTEP ENDIF
//*
```

SMTPNOTE used in application program
A useful attribute of the SMTPNOTE command is that you can call it from inside any application program. Because it is a REXX EXEC, you need to use the TSO/E service facility IKJEFTSR (or its alias
TSOLNK), which allows the use of commands, programs, CLISTs, and REXX EXECs from within application programs. The only restriction is that you need to execute your application program from within the TSO/E environment. It means that you need to use a program interface TMP (Terminal Monitor Program) like IKJEFT01, IKJEFT1A, or IKJEFT1B. Here is the example of a complete program written in PL/I for sending e-mails from a mainframe. What the program does is to send an e-mail to some e-mail address, including the content of the input record:

```
SENDMAIL: PROC OPTIONS(MAIN) REORDER;

DCL LENGTH   BUILTIN;
DCL PLIRETV   BUILTIN;
DCL ADDR     BUILTIN;

DCL TEMPIN   FILE RECORD INPUT;
DCL TEMPF   FILE RECORD OUTPUT;
DCL SYSPRINT OUTPUT;

DCL REcin   CHAR(80);
DCL 1 EMAIL,
  2 FROM CHAR(50) VAR,
  2 TO   CHAR(50) VAR,
  2 CC   CHAR(50) VAR,
  2 SUBJECT CHAR(50) VAR,
  2 DATASET CHAR(50) VAR,
  2 RC    BIN FIXED(31),
  2 ABEND BIN FIXED(31);

DCL 1 CMD,
  2 CMDLINE CHAR(300) VAR,
  2 CMDTYPE CHAR(1),
  2 RC    BIN FIXED(31),
  2 ABEND BIN FIXED(31);

DCL LINE     CHAR(80);
DCL CMDTYPE2 CHAR(1) INIT('02'X);
DCL CMDTYPE5 CHAR(1) INIT('05'X);
DCL QUOTE CHAR(1) INIT('7D'X);
DCL I      BIN FIXED(15);
DCL EOF    BIT(1) INIT('0'B);

ON ENDFILE(TEMPIN) EOF='1'B;

EMAIL.FROM='SENDERID@COMPANY.COM';
EMAIL.SUBJECT='E-mail from application program';
EMAIL.DATASET=QUOTE||'TSOTEMP.TXT'||QUOTE;
```
READ FILE(TEMPIN) INTO(RECIN);
DO WHILE(~EOF):

OPEN FILE(TEMPF);
LINE='FIRST LINE OF MESSAGE';
WRITE FILE(TEMPF) FROM(LINE);
LINE='SECOND LINE OF MESSAGE';
WRITE FILE(TEMPF) FROM(LINE);
LINE='THIRD LINE OF MESSAGE';
WRITE FILE(TEMPF) FROM(LINE);
CLOSE FILE(TEMPF);

IF INDEX(RECIN,'@')=Ø
THEN DO;
    LINE='WRONG TOUSER: '||TOUSER;
    PUT SKIP DATA(LINE);
    GOTO NEXTTOUSER;
END;
DO I=1 TO 8Ø WHILE(SUBSTR(TOUSER,I,1)¬=' ');
END;
EMAIL.TO=SUBSTR(RECIN,1,I-1);

CMDLINE='SMTPNOTE SU(' || E-MAIL.SUBJECT || ') TO(' || E-MAIL.TO
    || ')' DA(' || E-MAIL.DATASET || ') FROM(' || E-MAIL.FROM
    || ')' NOCC BATCH';
CMD.CMDTYPE=CMDTYPE5;
CMD.RC=Ø;
CMD.ABEND=Ø;
CALL TSOCMD(CMD);
SELECT (CMD.RC);
WHEN (Ø)
    PUT SKIP LIST('E-MAIL SENT SUCCESSFULLY');
WHEN (4)
    PUT SKIP LIST('ERROR IN TSOLNK',CMD.ABEND);
WHEN (8)
    PUT SKIP LIST('ERROR IN STMPNOTE',CMD.ABEND);
OTHERWISE;
END;
READ FILE(TEMPIN) INTO(RECIN);
ENDDO;

TSOCMD: PROC(CMD);

DCL 1 CMD,
    2 CMDLINE CHAR(300) VAR,
    2 CMDTYPE CHAR(1),
    2 RC BIN FIXED(31),
    2 ABEND BIN FIXED(31);

DCL TSOLNK ENTRY(}
DCL CHAR1 CHAR(1);
DCL BIT8 BIT(8) BASED(ADDR(CHAR1));
DCL 1 PARM1,
   2 PARM11BIN FIXED(15,Ø),
   2 PARM13BIT(8),
   2 PARM14BIT(8):
DCL PARM2 CHAR(300) VAR;
DCL PARM3 BIN FIXED(31,Ø);
DCL PARM4 BIN FIXED(31,Ø);
DCL PARM5 BIN FIXED(31,Ø);
DCL PARM6 BIN FIXED(31,Ø);

RC=Ø;
ABEND=Ø;

PARM11=Ø;
PARM13='ØØØØØØØ1'B;
CHAR1 =CMDTYPE;
PARM14=BIT8;
PARM2 =CMDLINE;
PARM3 =LENGTH(PARM2);
CALL TSOLNK(PARM1,PARM2,PARM3,PARM4,PARM5,PARM6);

SELECT (PLIRETV);
 WHEN (Ø);
 WHEN (4)
   DO;
      RC=8;
      ABEND=PARM6;
   END;
 WHEN (12)
   DO;
      RC=8;
      ABEND=PARM5;
   END;
 WHEN (20,24)
   DO;
      RC=4;
      ABEND=PARM5;
   END;
The main program must be run through the IKJEFT01 program using a call statement in this JCL like:

```
//SENDMAIL JOB MSGCLASS=Z,CLASS=B,NOTIFY=&SYSUID
//***************************************************************************
//*  SENDING E-MAILS FROM PL/1 PROGRAM
//***************************************************************************
//EXECMAIL EXEC PGM=IKJEFT01
//STEPLIB DD DSN=APPHLQ.UNIT.LOADLIB,DISP=SHR
//SYSPROC DD DSN=SYS1.OEM.CLIST,DISP=SHR
//SYSTSPRT DD SYSOUT=* 
//CEEDUMP DD SYSOUT=* 
//PLIDUMP DD SYSOUT=* 
//SYSPRINT DD SYSOUT=* 
//SYSUDUMP DD SYSOUT=* 
//SYSSIN DD DUMMY 
//TEMPF DD DSN=APPHLQ.REPORT.FILE,DISP=OLD 
//SYSTIN DD * 
  CALL 'APPHLQ.UNIT.LOADLIB(SENDMAIL)'
//*
```

As you can see, JCL must have the related DSN in SYSPROC if you try to invoke some CLIST or REXX procedure. Here we use STMPNOTE REXX EXEC, which resides in the SYS1.OEM.CLIST dataset library. The main program must be linked with the following DSN in SYSLIB; SYS1.LPALIB, where TSOLNK resides.

CONCLUSION

It can be very useful to have e-mail alerts about your system jobs. But, you do not need to have e-mails sent to your desktop. You can now send short e-mails to a mobile phone, using, for example, the ICQ SMS service, which allows you to send short messages (up to 160 characters). All you need to do is send an e-mail to the following address: phone_number@ICQSMS.COM. The phone number consists of three parts: the country code, area or mobile service code, and the
mobile phone number. This service is free. So if your mobile phone provider is on the ICQ list you can have your e-mail even before your operations realized that your program is in trouble.

Predrag Jovanovic  
Project Developer  
Pinkerton Computers Consultants Inc (USA)  © Xephon 2001

z/OS migration considerations

There has been some debate recently among users who wonder if they should install OS/390 Version 2 Release 10 prior to installing z/OS. Obviously this is dependent on the requirements and configuration of each site, but we recently made the move, and found that in our shop the move to OS/390 Version 2 Release 10 was more advantageous than moving straight to z/OS Version 1 Release 1.

This was because when running on a z900, z/OS requires the use of z/Architecture mode with its 64-bit real addressing. We had a large installed base of in-house Assembler language programs that were dependent on real storage addresses, but it is possible that some shops might have vendor products with this kind of dependency.

We found that in our situation it was preferable to use Release 10 as the operating system when the z900 was first deployed, rather than having z/OS running. This was because in Release 10 it is possible to switch between ESA/390 (31-bit) and z/Architecture (64-bit) real addressing to test our programs. You cannot switch addressing modes with z/OS. We simply installed OS/390 Version 2 Release 10, and ran a 64-bit LPAR on the z900 for testing. This does not affect normal applications or products because these are 64-bit real addresses, not the virtual addresses used by most applications.

Systems Programmer (USA)  © Xephon 2001
A C preprocessor

THE PROBLEM

If you do not work in a country that uses the LATIN1 code page, you could encounter certain problems during coding. You will certainly face problems when coding the C programming language. Square brackets [] and the negation sign | are very important elements in C programs. In our code page these characters are represented in a manner that differs from what the C compiler would expect. One of the solutions is the use of trigraphs:

```plaintext
??( [ ??) ] ??!
```

However C programs written with trigraphs are not clear for reading and are complicated by transferring to different platforms. Solutions using #define statements in C programs are not suitable for this problem. The reason is that the #define statement does a general change, which is not appropriate when we want to use brackets in constants and comments.

A SOLUTION

We wrote a small preprocessor that resolves this problem by translating critical characters to characters that the compiler would expect. The preprocessor changes characters selectively, but only when they belong to statements, not when they are parts of constants or comments.

Note: the preprocessor is written in C. You have to compile and link it with the original procedure and then modify the existing procedure by adding the step for executing the preprocessor.

CPREPROC

```c
#include <stdio.h>
#include <string.h>
#define TRUE     1
#define FALSE    0
main()
{
```
FILE *fp1, *fp2;
char InStr[225], *pCh;
int feof(FILE *fp1);
int Noch = 225, i, j;
int indNotComment, indNotConstant;

fp1=fopen("DD:SYSIN","r");
if (fp1 == NULL) printf("Error opening SYSIN !");

fp2=fopen("DD:SYSOUT","w");
if (fp2 == NULL) printf("Error opening SYSOUT !");

fgets(InStr,Noch,fp1);
indNotComment = indNotConstant = TRUE;

while( ) feof(fp1))
{
  for (i=0, pCh = (char *)InStr; *pCh != '\0';pCh++)
    switch(*pCh)
    {
      case 0x4A: if (indNotComment && indNotConstant)
        *pCh = 0xad; /* [ -> i */
          break; /* [ -> i */
      case 0x5A: if (indNotComment && indNotConstant)
        *pCh = 0xbd; /* ] -> u */
          break; /* ] -> u */
      case 0x4F: if (indNotComment && indNotConstant)
        *pCh = 0xaa; /* ! -> ] */
          break; /* ! -> ] */
      case '\': if (indNotConstant)
        pCh++ ;
        break;
      case '\\': if (indNotConstant)
        pCh++ ;
        break;
      case '"': if (indNotComment)
        indNotConstant = 1 - indNotConstant;
        break;
      case '/': if (indNotConstant && pCh[1] == '*')
        {
          pCh++;
          indNotComment = FALSE;
        }
        break;
      case '*': if (indNotConstant && indNotComment &&
        pCh[1] == '/')
        {
          pCh++;
          indNotComment = TRUE;
        }
        break;
    }
JOB TO COMPILE AND LINK THE C PREPROCESSOR

//SYSTMØ2L   JOB MSGCLASS=X,MSGLEVEL=(1,0),NOTIFY=&SYSUID,CLASS=A
//COMPRESL EXEC EDCCL,CPARM='OBJ,SOURCE,OFFSET',
//         INFILE=userid.USER.C(CPREPROC)
//LKED.SYSLMOD  DD DSN=SYS1.LINKLIB(CPREPROC),DISP=SHR

EDCCL1 FOR COMPILING AND LINKING

//********************************************************************
//*                                                                  *
//*  COMPILE AND LINK EDIT A C PROGRAM                               *
//*                                                                  *
//*  OS/39Ø C/C++                                                    *
//*                                                                  *
//********************************************************************
//*
//EDCCL PROC  INFILE=,                   < INPUT ... REQUIRED
//  CREGSIZ='4M',                        < COMPILER REGION SIZE
//  CRUN=,                               < COMPILER RUNTIME OPTIONS
//  CPARM='SOURCE',                      < COMPILER OPTIONS
//  CPARM2=,                             < COMPILER OPTIONS
//  CPARM3=,                             < COMPILER OPTIONS
//  SYSLBLK='32ØØ',                      < BLOCKSIZE FOR &&LOADSET
//  LIBPRFX='CEE',                       < PREFIX FOR LIBRARY DSN
//  LNGPRFX='CBC',                       < PREFIX FOR LANGUAGE DSN
//  CLANG='EDCMSGE', < NOT USED IN THIS RELEASE. KEPT FOR COMPATIBILITY
//  LREGSIZ='1Ø24K',                     < LINK EDIT REGION SIZE
//  LPARM='AMODE=31,MAP',                < LINK EDIT OPTIONS
//  DCBØØ='(RECFM=FB,LRECL=8Ø,BLKSIZE=32ØØ)',      <DCB FOR LRECL 8Ø
//  DCB32ØØ='(RECFM=FB,LRECL=32ØØ,BLKSIZE=128ØØ)', <DCB FOR LRECL 32ØØ
//  OUTFILE='&&GSET(GO),DISP=(MOD,PASS),UNIT=VIO,SPACE=(TRK,(7,7,1))',
//  TUNIT='VIO'                          < UNIT FOR TEMPORARY FILES
//*-------------------------------------------------------------------
//*  PREPROCESOR STEP:
//*-------------------------------------------------------------------
//CONVERZ EXEC PGM=CPREPROC
//STELIB  DD  DSNAME=SYS1.LINKLIB,DISP=SHR
//SYSPRINT DD  SYSOUT=*                  
//SYSIN    DD  DSNAME=&INFILE,DISP=SHR
//SYSOUT   DD  DSNAME=&&CKONV,DISP=(NEW,PASS),
You will have to implement similar changes in other procedures that are used on your installation (EDCC, EDCCLG, etc).

Emina Spasic and Dragan Nikolic
Systems Programmers
Postal Savings Bank (Yugoslavia) © Xephon 2001
A REXX program to initialize DASD

For those who work in storage administration, DASD initialization is something that has to be done every once in a while, but, hopefully, only a few volumes at a time. However, last time I was faced with the need to initialize some DASD, I was asked for 50. Not that initializing DASD is difficult, but getting 50 free addresses, and running through all the steps needed, for each one of those volumes, is a pain. So, as usually happens when I am faced with a boring task, I wrote a program to do it. The best part is that this program will work for any number of volumes. In order to be flexible, and easy to use, I decided to use an ISPF panel to obtain the values to work with:

```
COMMAND ==> 
```

```
DISK INIT
Free volume prefix .........................: 
Prefix for initialization ..................: 
Device Type to use .......................: 
    DASD Model (if applicable) ..........: 
Number of Volumes to initialize ..........: 
SMS Volumes (Y/N) .....................: 
JOBNAME to use in JOBS .................: 

ENTER to Execute
END  to Cancel
```

In order for this program to be useful at your shop, without any changes, your available volumes for initialization must be on-line, and must begin with a common and unique prefix. In our case, that prefix is FR: all our free volumes are FRxxxx, hence the default used.

This is the way it works: you specify, via the ISPF panel, the DASD prefix to be searched for free volumes, the prefix with which to initialize the new volumes, the device type to use (and model, if you are using a 3380/3390), how many volumes to initialize, and whether they are to be SMS managed. I wrote code to validate only 3380 and 3390 device types, but any other kind of DASD can be added without trouble. As this program will submit four jobs, the JOBNAME to use
will have to be specified in here as well. The defaults are assumed prior to the ISPF panel invocation, and, in some cases, are systems dependent, because we have different environments in our different systems. After you specify the values to use, the program will execute an IDCAMS DCOLLECT, in order to obtain the volume information to work with. This is done in foreground and, if there are many volumes on-line that match the specified prefixes, it may take some time. For this to work, IDCAMS must be defined as an authorized program in IKJTSOxx AUTHCMD NAMES to run under TSO.

DCOLLECT will produce two output files, one with the free volumes, the other with the DASD that match the new volume prefix. A few validations will be done, based on the contents of this two files, after which, if everything comes out all right, four jobs will be submitted, with TYPRUN=HOLD. For this kind of task, I always check each job, prior to executing it, via SDSF and an SJ. It is too easy for something to be painfully wrong.

The first job will use an ICEGENER to submit a ‘RO *ALL, VARY addr, OFFLINE’ for each volume. In a sysplex environment, this will route the VARY command to all the systems. If you are not in a sysplex environment, take out the ‘RO *ALL’. However, you will have to devise a way to do the VARY ONLINE on the other systems that share the device. The third one will do the ONLINE. The submitted JCL will look like this, and the third JCL will differ only in the OFFLINE keyword, and in the description:

```plaintext
//jobname JOB (ACCT#),'PUT VOLUMES OFFLINE',
//       MSGLEVEL=(1,1),
//       TYPRUN=HOLD,
//       CLASS=W,MSGCLASS=X
//*
//PUT#OFF EXEC PGM=ICEGENER
//SYSPRINT DD SYSOUT=*
//SYSUT2 DD SYSOUT=(*,INTRDR)
//SYSIN DD DUMMY
//SYSUT1 DD DATA,DLM='££'
/*$VS,'RO *ALL,V xxxx,OFFLINE'
££
/*
```

The second job will run ICKDSF and the INITs for all the volumes, with VERIFY(VOLSER) and, if the volumes are to be SMS-managed, with the STORAGEGROUP keyword as well. The VTOC and
INDEXED VTOC size are device type – and model-dependent, and hard-coded in the program, within a SELECT, so it will be easy to accommodate new/old types. The submitted JCL will look like this:

```bash
//jobname JOB (ACCT#),'INIT VOLUMES OFFLINE',
// MSGLEVEL=(1,1),
// TYPRUN=HOLD,
// CLASS=W,MSGCLASS=X
/**
//INIT#OFF EXEC PGM=ICKDSF,PARM='NOREPLYU'
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
   INIT UNIT(xxxx) INDEX(Ø,1,Ø14) VTOC(1,Ø,Ø6Ø) -
   VERIFY(FRxxxx) VOLID(prfØ1) STORAGEGROUP
/**

The fourth job will be generated only if the volume is to be SMS-managed, and it will create the VVDS. If you want the VVDS to be created, even for non-SMS volumes, you will have to remove an IF, in the GENERATE_JOBS procedure. This submitted JCL will look like this:

```bash
//jobname JOB (ACCT#),'CREATE VVDS',
// MSGLEVEL=(1,1),
// TYPRUN=HOLD,
// CLASS=W,MSGCLASS=X
/**
//CRE#VVDS EXEC PGM=IDCAMS,REGION=4M
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
   DEFINE CLUSTER (NAME(SYS1.VVDS.Vprfnnn) -
                   VOL(prfnnn) NONINDEXED TRK(45 Ø))
/**
```

Some validations are made at the panel level:

- Volume prefixes length (must be at least one character).
- Device type, and model (3380/J/K 3390/1/2/3). Others can be easily added.
- Number of volumes to initialize (must be numeric and greater than zero).
- SMS managed (must be Y or N).
- JOBNAME length (must be 8).
This is the DISKINIT panel definition:

)ATTR
   # TYPE(INPUT) INTENS(LOW) COLOR(TURQ) CAPS(ON) PADC('_')
| @ TYPE(TEXT) INTENS(LOW) COLOR(BLUE) SKIP(ON)
+ @ TYPE(TEXT) INTENS(HIGH) COLOR(YELLOW) SKIP(ON)
£ @ TYPE(TEXT) INTENS(HIGH) COLOR(TURQ) SKIP(ON)
)

@COMMAND ======#ZCMD

+ DISK INIT

Free volume prefix .........................:    #Z  |
Prefix for initialization ..................:    #Z  |
Device type to use ........................:    #Z  |
   DASD Model (id applicable) ............:    #Z  |
Number of Volumes to initialize ..........:    #Z  |
   SMS Volumes £(Y/N)......................:    #Z  |
   JOBNAME to use in JOBS .................:    #Z  |

   @ENTER|to Execute
   @END  |to Cancel
)

.INIT
   .ZVARS='(VOLPREF NEWPREF TYPE MODEL HOWMANY SMS JOBNAME)'
)

PROC
   VER (&VOLPREF,LEN,GE,1)
   VER (&NEWPREF,LEN,GE,1)
   VER (&TYPE,NONBLANK,LIST,338Ø,339Ø)
   IF  (VER(&TYPE,LIST,338Ø))
      VER(&MODEL,NONBLANK,LIST,J,K)
   ELSE
      IF  (VER(&TYPE,LIST,339Ø))
         VER(&MODEL,NONBLANK,LIST,1,2,3)
      ELSE
         &MODEL = ' '
   END
   VER (&HOWMANY,NONBLANK,NUM)
   VER (&SMS,NONBLANK,LIST,S,N)
   VER (&JOBNAME,LEN,EQ,8)
   &PFKEY = .PFKEY
)

In order to obtain free gaps in the volume numeration, the program will order the second output file, generated by DCOLLECT, by VOLSER, using an EDIT macro (X$INDK$X) in the beginning of the GET_NEW_VOL procedure. It is a very simple macro:
If there are enough volumes to satisfy your request, the INIT related jobs will be generated and submitted, and a LOG record will be formatted for each DASD volume initialized. These records will be added to a LOG file (which will be created in the first use of this program), and will be recorded in a temporary file, which will be browsed, prior to program termination, so you will have a first visual information of the volumes selected:

BROWSE userid.D01241.T121231.VIEW.LOG Line 0000000 Col 001 000
Command ---> Scroll ---> CSR

**************************** Top of Data ****************************
FRxxxx init as newprf on yyyymmdd - hh:mm:ss by RACF user name
**************************** Bottom of Data ****************************

The program is shown below:

/* REXX */
address "ISREDIT"
"macro"
"sort 25 3Ø"
"save"
"end"
return
/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

howmany=1
volpref="FR"       /* Free Volume PREFIX */
newpref="CPG"      /* New Volume PREFIX */
devtype=339Ø       /* Default Device Type */
model=3            /* Default type model */
sms="Y"            /* SMS init is the default */
jobname=userid()"I"  /* JOB NAME to use */
acctnum="ACCT#"    /* ACCOUNT info to use */
job_class="A"      /* JOB CLASS to use */
msg_class="X"      /* MSG CLASS to use */
log_hlq="PDOIDM"   /* LOG FILE High Level Qualifier */
zedlmsg=""
/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */
/* Invoke ISPF Panel to specify invocation values */
/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */
do z=1
    address "ISPEXEC" "control errors return"
    address "ISPEXEC" "display panel(diskinit)"
    select
*/
when wordpos (pfkey,"PFØ3 PF15 PFØ4 PF16")>Ø then
do
  zedlmsg=zedlmsg,
  "You terminated the DISKINIT process, by",
  "pressing the PFØ3/15 or PFØ4/16 Key"
leave z
end
otherwise
  l=6-length (newpref)
  limit=copies (9,1)
  if limit<howmany then
do
  zedlmsg=zedlmsg,
  "You specified more volumes to initialize",
  "("howmany") than it is possible with prefix",
  newpref "("limit")"
end
else
do
  call alloc_files
leave z
end
end
if zedlmsg="" then
do
  address "ISPEXEC" "SETMSG MSG(ISRZØØ1)"
  zedlmsg=""
end
if zedlmsg="" then
do
  address "ISPEXEC" "SETMSG MSG(ISRZØØ1)"
end
return
/* - - - - - - - - - - - - - */
alloc_files:
/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */
/* ALLOCATE Work files                                         */
/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */
"alloc f(sysprint) shr reuse dummy"
hlqs=userid(),              /* high level qualifiers for work files */
  /*".D"date("J"),*/
  /*".T"space(translate(time(),".\","."),Ø)
out_dsn_1=""hlqs".OUTFILE.#Ø1" /* Work File for Free Volumes */
out_dsn_2=""hlqs".OUTFILE.#Ø2" /* Work file for New Volumes */

"alloc f(dd#1) new dsorg(PS) recfm(V B) lrecl(454)",
  "da(out_dsn_1") space (1Ø 5) tracks release"
if rc=Ø then
do
  "alloc f(dd#2) new dsorg(PS) recfm(V B) lrecl(454)",

"da("out_dsn_2") space (10 5) tracks release"
if rc=Ø then
do
  in_dsn=""hlqs".SYSIN"
  "alloc f(SYSIN) new dsorg(PS) recfm(F B) lrecl(80)"
  "da("in_dsn") space (1 1) tracks release reuse"
  if rc=Ø then
do
    call format_sysin
    "alloc f(sysin) shr reuse da(*)"
  end
else
do
  zedlmsg=zedlmsg,
  "Error ("rc") on the ALLOC for "in_dsn"
  "free f("dd#1","dd#2")"
end
else
do
  zedlmsg=zedlmsg,
  "Error ("rc") on the ALLOC for "out_dsn_2"
  "free f("dd#1")"
end
else
do
  zedlmsg=zedlmsg,
  "Error ("rc") on the ALLOC for "out_dsn_1"
end
"alloc f(sysprint) shr reuse da(*)"
return

/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */
format_sysin:
/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */
/* Format DCOLLECT SYSIN and call IDCAMS                        */
/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */
queue" DCOLLECT OFILE("dd#1") VOL("volpref"*) NODATAINFO"
queue" DCOLLECT OFILE("dd#2") VOL("newpref"*) NODATAINFO"
"execio "queued()" diskw SYSIN (finis)"
if rc=Ø then
do
  "alloc f(sysin) reuse old da("in_dsn") delete"
  "CALL *(IDCAMS)"
  if rc=Ø then
do
    "alloc f("dd#1") old da("out_dsn_1") delete"
    "execio * diskr "dd#1" (finis stem volume_info.)"
    if rc=Ø then
do
      if volume_info.Ø>Ø then
do
call process_data
else
do
    zedlmsg=zedlmsg,
    "No volumes "volpref, 
    "* were obtained"
end
else
do
    zedlmsg=zedlmsg,
    "Error ("rc") on "out_dsn_1" READ"
end
"free f("dd#1")"
else
do
    zedlmsg=zedlmsg,
    "Error ("rc") during DCOLLECT execution.".
    "Process state unknown."
end
else
do
    zedlmsg=zedlmsg,
    "Error ("rc") on the WRITE for "in_dsn
    "dropbuf"
end
return
/* - - - - - - - - - - - - - */
process_data:
if volume_info.Ø<howmany then
    do
        zedlmsg=zedlmsg,
        "Not enough volumes "volpref"* to initialize",
        "You asked for "howmany", but only "volume_info.Ø,
        "are available"
    end
else
    do
        /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */
        /* Set VTOC, VTOCIX an VVDS sizes, depending on DASD device */
        /* and model type */
        /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */
        select
            when devtype=339Ø then
                do
                    if model=3 then
                        do
                            vtoc="1,0.060"
                            vti=0.1.014"
                        end
                    end
                end
            else
                do
                    zedlmsg=zedlmsg,
                    "Error ("rc") on the WRITE for "in_dsn
                    "dropbuf"
                end
            end
        end
end
when devtype=3380 then
do
  if model="K" then
do
    vtoc="1.0.060"
    vtix="0.1.014"
    vvds="45 Ø"
  else
    vtoc="1.0.045"
    vtix="0.1.014"
    vvds="30 Ø"
  end
  end
otherwise
  nop
end

/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */
/* Process DCOLLECT obtained information for free volumes    */
/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */
do a=1 to volume_info.Ø
  parse value volume_info.a with 25 volid,
                      31 ..
                      45 tot_cap,
                      49 ..
                      69 dev_type,
                      77 dev_num,
                      79 .
  dev_type=strip(dev_type)
  dev_num =c2x(dev_num)
  tot_cap =c2d(tot_cap)
/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */
/* Set Model Type, based on DASD total capacity           */
/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */
select
  when dev_type="3390" then
do
    select
      when tot_cap>2771500 then
        model_a="9"
      when tot_cap>1847600 then
      end
    end
  end
end

model_a="3"
when tot_cap> 923800 then
  model_a="2"
otherwise
  model_a="1"
end
end
when dev_type="3380" then
do
  select
  when tot_cap>1230900 then
    model_a="K"
  when tot_cap> 615400 then
    model_a="E"
otherwise
    model_a="J"
  end
end
otherwise
  model_a=""
end
/* - - - - - - - - - - - - - - - - - - - - - - - - - - - */
/* Is this DASD of the same type & model as specified? */
/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */
if  devtype=dev_type & model=model_a then
do
  o=o+1
  addr.o  =dev_num
  volid.o =volid
  if  o=how_many then
    leave a
  end
/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */
/* Do we have as many free DASD as requested? */
/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */
select
  when o=Ø then
    do
      zedlmsg=zedlmsg,
      "You asked for "howmany devtype"/"model, volumes, but there are none available"
    end
  when o<howmany then
    do
      zedlmsg=zedlmsg,
      "You asked for "howmany devtype"/"model, volumes, but there are only "o" available"
    end
  otherwise
    call get_new_vol
end
get_new_vol:
/* Process DCOLLECT obtained information for new volumes */
drop volume_info.
    /* Edit work file and order by VOLSER */
    address "ISPEXEC" "edit dataset("out_dsn_2") macro(x$indk$x)"
    "alloc f("dd#2") old da("out_dsn_2") delete"
    "execio * diskr "dd#2" (finis stem volume_info.)"
if rc=Ø then
    do
        first=Ø
        /* Are there any volumes with the specified prefix? */
        if volume_info.Ø>Ø then
            do a=1 to volume_info.Ø
                parse value volume_info.a with 25 nnn 31.
                nnn=right(nnn,1)
                /* Is the suffix a valid whole number? */
                if datatype(nnn,"W") then
                    do
                        if nnn-first>=howmany+1 then
                            do
                                leave a
                                end
                            else
                                do
                                    first=nnn
                                    end
                                end
                            else
                                nop
                            end
                end
                /* Is the available range big enough? */
                if first+howmany>limit then
                    do
                        zedlmsg=zedlmsg,
                        "The number of volumes to initialize ("howmany")",
                        "exceeds the available range : "first+1" to "limit
                    end
                else
                    do
                        /* Are there any volumes with the specified prefix? */
                        if volume_info.Ø>Ø then
                            do a=1 to volume_info.Ø
                                parse value volume_info.a with 25 nnn 31.
                                nnn=right(nnn,1)
                                /* Is the suffix a valid whole number? */
                                if datatype(nnn,"W") then
                                    do
                                        if nnn-first>=howmany+1 then
                                            do
                                                leave a
                                                end
                                            else
                                                do
                                                    first=nnn
                                                    end
                                                end
                                            else
                                                nop
                                            end
                                end
                                /* Is the available range big enough? */
                                if first+howmany>limit then
                                    do
                                        zedlmsg=zedlmsg,
                                        "The number of volumes to initialize ("howmany")",
                                        "exceeds the available range : "first+1" to "limit
                                    end
                                else
                                    do
                                    first=Ø
                                    end
                                end
                            end
                        end
                    end
                end
            end
        else
            nop
        end
    end
end
return
/* - - - - - - - - - - - - */
do
    call generate_jobs
end
else
    do
        zedlmsg=zedlmsg,
            "Error ("rc") on "out_dsn_2" READ"
    end
"free f("dd#2")"
return
/* - - - - - - - - - - - - - */
generate_jobs:
/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */
/* Format, and SUBMIT, the INIT process related JOBs */
/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */
name=name() /* get RACF user name */
do a=1 to howmany
    new_value=first+a
    new_vol.a=newpref||right(new_value,1."Ø")
    log_line.a=volid.a init as "new_vol.a, 
        "on "date("S")": "time()" by "name"
end
call put_offline /* format PUT OFFLINE JOB */
call init_offline /* format INIT OFFLINE JOB */
call put_online /* format PUT ONLINE JOB */
ddname="O"time("S")
"alloc f("ddname") writer(intrdr) sysout(A) LRECL(8Ø) RECFM(F)"
"execio "off_line.Ø" diskw "ddname" (stem off_line.)"
"execio "init_off.Ø" diskw "ddname" (stem init_off.)"
"execio "on_line.Ø" diskw "ddname" (stem on_line.)"
if sms="Y" then
do
    job#=4
    call create_vvds /* format CREATE VVDS JOB */
        "execio "cr_vvds.Ø" diskw "ddname" (stem cr_vvds.)"
end
else
do
    job#=3
end
"execio Ø diskw "ddname" (finis)"
"free f("ddname")"
if howmany=1 then
do
    zedlmsg=zedlmsg,
        job# Jobs ("jobname"), to Initialize "new_vol.1, 
        "have been submitted"
end
else
do
zedlmsg=zedlmsg,
    job#" Jobs ("jobname"), to Initialize "new_vol.1" to",
    new_vol.howmany", have been submitted"
end
call log_down
return
/* - - - - - - - - - - - - - */
put_offline:
/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */
/* Format the VARY OFFLINE JCL */
/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */
queue"//"jobname" JOB ("acctnum"), 'PUT VOLUMES OFFLINE',"
queue"//           MSGLEVEL=(1,1),"
queue"//           TYPRUN=HOLD,NOTIFY=&SYSUID,"
queue"//           CLASS="job_class", MSGCLASS="msg_class"
queue"//"
queue"//"PUT#OFF EXEC PGM=ICEGENER"
queue"//"SYSPRINT DD SYSOUT=""
queue"//"SYSUT2 DD SYSOUT=(*,INTRDR)"
queue"//"SYSIN DD DUMMY"
queue"//"SYSUT1 DD DATA,DLM='££'
do  a=1 to howmany
    queue"/*$VS,'RO *ALL.V "addr.a",OFFLINE'"
end
queue"££"
queue"/*
do  a=1 to queued()
    pull off_line.a
end
off_line.Ø=a-1
return
/* - - - - - - - - - - - - - */
init_offline:
/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */
/* Format the INIT OFFLINE JCL */
/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */
queue"//"jobname" JOB ("acctnum"), 'INIT VOLUMES OFFLINE',"
queue"//           MSGLEVEL=(1,1),"
queue"//           TYPRUN=HOLD,NOTIFY=&SYSUID,"
queue"//           CLASS="job_class", MSGCLASS="msg_class"
queue"//"
queue"//"INIT#OFF EXEC PGM=ICKDSF,PARM='NOREPLYU'"
queue"//"SYSPRINT DD SYSOUT=""
queue"//"SYSIN DD *"
if  sms="Y" then
    opts="STORAGEGROUP"
else
    opts=""
do  a=1 to howmany
    queue" INIT UNIT("addr.a") INDEX("vtix") VTOC("vtoc") -"
    queue" VERIFY("valid.a") VOLID("new_vol.a") "opts
end
queue"/*"
queue"/*"
do  a=1 to queued()
   pull init_off.a
end
init_off.Ø=a-1
return
/* - - - - - - - - - - - - - */
put_online:
/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */
/* Format the VARY ONLINE JCL */
queue"*/"jobname" JOB ("acctnum"),'PUT VOLUMES ONLINE',"
queue"// MSGLEVEL=(1,1),"
queue"// TYPRUN=HOLD,NOTIFY=&SYSUID,"
queue"// CLASS="job_class",MSGCLASS="msg_class"
queue"*/"*"
queue"//"jobname" JOB ("acctnum"),'PUT VOLUMES ONLINE',"
queue"// MSGLEVEL=(1,1),"
queue"// TYPRUN=HOLD,NOTIFY=&SYSUID,"
queue"// CLASS="job_class",MSGCLASS="msg_class"
queue"*/"*"
queue"//PUT#ON EXEC PGM=ICEGENER"
queue"//SYSPRINT DD SYSOUT=""
queue"//SYSUT2 DD SYSOUT=(*,INTRDR)"
queue"//SYSIN DD DUMMY"
queue"//SYSUT1 DD DATA,DLM='££'"
do  a=1 to howmany
   queue"/*$VS,'RO *ALL.V "addr.a",ONLINE'"
end
queue"££"
queue"/*"
do  a=1 to queued()
   pull on_line.a
end
on_line.Ø=a-1
return
/* - - - - - - - - - - - - - */
create_vvds:
/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */
/* Format the CREATE VVDS JCL */
queue"*/"jobname" JOB ("acctnum"),'CREATE VVDS',"
queue"// MSGLEVEL=(1,1),"
queue"// TYPRUN=HOLD,NOTIFY=&SYSUID,"
queue"// CLASS="job_class",MSGCLASS="msg_class"
queue"*/"*"
queue"//CRE#VVDS EXEC PGM=IDCAMS,REGION=4M"
queue"//SYSPRINT DD SYSOUT=""
queue"//SYSIN DD *"
do  a=1 to howmany
   queue" DEFINE CLUSTER (NAME(SYS1.VVDS.V"new_vol.a") -"
      queue" VOL("new_vol.a") NONINDEXED TRK("vvds")"
end
queue"/*"
do  a=1 to queued()
pull cr_vvds.a
cr_vvds.Ø=a-1
return
/* - - - - - - - - - - - - - */
name: procedure
/* - - - - - - - - - - - - - */
/* Get user name from RACF profile */
/* - - - - - - - - - - - - - */
x=outtrap(user.,"NOCONCAT")
"LU "userid()
x=outtrap("OFF")
parse value user.1 with "NAME="who"OWNER="
return strip(name)
/* - - - - - - - - - - - - - */
log_down:
/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */
/* Write the INIT information on the LOG file */
/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */
dd="O"time("S")
log_dsn=""log_hlq".##LOG.INITDASD"
sysmsg=sysdsn(log_dsn)
if  sysmsg="OK" then
  do
    "alloc f("dd") mod reuse da("log_dsn")"
  end
else
  do
    "alloc f("dd") new reuse da("log_dsn")",
    "lrecl (254) recfm(V B) space (1Ø 5) tracks"
  end
if  rc¬Ø then
  do
    zedlmsg=zedlmsg,
    "Error ("rc") on the ALLOC for "log_dsn
  end
else
  do
    "execio "howmany" diskw "dd" (finis stem log_line.)"
    if  rc¬Ø then
      do
        zedlmsg=zedlmsg,
        "Error ("rc") on the WRITE for "log_dsn
      end
    "Free f("dd")"
  end
/* - - - - - - - - - - - - - */
/* View this execution LOG, using a temporary work file */
/* - - - - - - - - - - - - - */
dd="W"time("S")
view_log=""hlqs".VIEW.LOG"
"alloc f("dd") new reuse da("view_log")",
    "lrecl (254) recfm(V B) space (1Ø 5) tracks delete"
if rc\=0 then
do a=1 to how_many
    say log_line.a
end
else
do
    "execio "howmany" diskw "dd" (finis stem log_line.)"
    if rc=0 then
        do
            address "ISPEXEC" "control errors return"
            address "ISPEXEC" "browse dataset("view_log")"
        end
        else
            do a=1 to how_many
                say log_line.a
            end
    "Free f("dd")"
end
return
/* - - - - - - - - - - - */

Systems Programmer (UK) © Xephon 2001

Comprehensive dynamic allocation facilitation

The accompanying ALLOC and ALLOCPL macros offer a means of requesting dynamic allocation with the most frequently used parameters, as well as permitting reentrant and register-addressed parameter implementations. Complete instructions are included as comments within the macros.

ALLOC

&NAME ALLOC &RENT=, 'YES' IF REENTRY DESIRED *
    &VERB='AL', DEFAULT IS TO ALLOCATE *
    &DDNAM=, DDNAME *
    &DSNAM=, DATA SET NAME *
    &MEMBR=, PDS MEMBER *
    &STATS=, INITIAL STATUS *
    &NDISP=, NORMAL DISPOSITION *
    &CDISP=, CONDITIONAL DISPOSITION *
    &TRK=, 'YES' FOR TRACK ALLOCATION *
    &CYL=, 'YES' FOR CYL. ALLOCATION *
    &PRIME=, PRIMARY SPACE QUANTITY *
&SECOND=, SECONDARY SPACE QUANTITY *
&DIR=, NO. OF DIRECTORY BLOCKS *
&RLSE=, 'YES' TO RLSE. UNUSED SPC. *
&VLSER=, VOLUME SERIAL NUMBER *
&UNIT=, UNITNAME *
&SYSOU=, SYSOUT CLASS *
&SPGNM=, SYSOUT PROGRAM NAME *
&SFMNO=, SYSOUT FORM NUMBER *
&OUTPT=, 'OUTPUT' STMT. REFERENCE *
&CLOSE=, 'YES' TO FREE AT CLOSE *
&SUSER=, SYSOUT REMOTE USER *
&BLKSZ=, BLOCKSIZE *
&DSORG=, DATA SET ORGANIZATION *
&LRECL=, LOGICAL RECORD LENGTH *
&RECFM=, RECORD FORMAT *
&RTDDN=, 'YES' TO RETURN THE DDNAME *
&UNALC= 'YES' TO FORCE UNALLOCATE

PRINT OFF

***********************************************************************
***********************************************************************
*  NAME: ALLOC                                                        *
*  DESCRIPTION:                                                       *
*  STORE VALUES INTO A DYNAMIC ALLOCATION (SVC 99) PARAMETER LIST     *
*  MAPPED BY THE ALLOCPL USER MACRO, AND ISSUE SVC 99.                *
*  MACRO VARIABLE VALUES:                                             *
*  TEXT UNIT VALUES MAY BE SPECIFIED IN ONE OF THREE WAYS:            *
*  1 AS A LITERAL IN QUOTES, IN WHICH CASE EITHER THE LITERAL        *
*     CHARACTER VALUE OR AN APPROPRIATELY TRANSLATED HEX VALUE        *
*     IS USED. FOR EXAMPLE, DDNAM='WHATEVER' RESULTS IN A             *
*     TEXT UNIT WHICH PLUGS IN WHATEVER AS THE DDNAME.                *
*     SPECIFYING TRK='10' RESULTS IN A TEXT UNIT IN WHICH THE         *
*     HEX EQUIVALENT OF DECIMAL 10 IS SUBSTITUTED.                    *
*  2 AS A NONQUOTED NAME, WHICH IS ASSUMED TO BE A LABEL NAME AND      *
*     HENCE AN ADDRESS. THE CONTENTS OF THIS ADDRESS MUST BE A        *
*     FULLY AND COMPLETELY FORMATTED TEXT UNIT.                       *
*  3 AS A REGISTER VALUE IN PARENTHESES. THE REGISTER MUST CONTAIN    *
*     THE ADDRESS OF A FULLY AND COMPLETELY FORMATTED TEXT UNIT.      *
*  &RENT IS AN ASSEMBLY TIME VARIABLE AND CAN ONLY BE SPECIFIED        *
*  AS A LITERAL OR OMITTED. &VERB SPECIFIES THE VERB CODE; IF CODED   *
*  IN ADDRESS OR REGISTER NOTATION IT MUST POINT TO A ONE-BYTE FIELD   *
*  CONTAINING THE CODE IN HEX.                                        *
*  ACCEPTABLE LITERAL VALUES:
*  RENT='YES'  VERB='AL'|'UN' DEFAULT IS 'AL'
*  DDNAM='ANY VALID DDNAME'
*  DSNAM='ANY VALID DATA SET NAME'

* MEMBR='ANY VALID PDS MEMBER NAME'
* STATS='NEW'|'OLD'|'SHR'
* NDISP='KEEP'|'DELETE'|'CATLG'
* CDISP='KEEP'|'DELETE'|'CATLG'
* TRK='YES'
* CYL='YES'
* PRIME='ANY VALID NUMERIC SPECIFICATION'
* SECND='ANY VALID NUMERIC SPECIFICATION'
* DIR='ANY VALID NUMERIC SPECIFICATION'
* RLSE='YES'
* VLSER='ANY VALID VOLUME SERIAL NUMBER'
* UNIT='ANY VALID UNITNAME'
* SYSOU='ANY VALID SYSOUT CLASS'
* SPGNM='ANY VALID SYSOUT PROGRAM NAME'
* SUSER='ANY VALID SYSOUT REMOTE USER'
* SFMNO='ANY VALID SYSOUT FORM NAME/NUMBER'
* BLKSIZE='ANY VALID NUMERIC SPECIFICATION'
* CLOSE='YES'
* DSORG='PS'|'PO'|'DA'
* LRECL='ANY VALID NUMERIC SPECIFICATION'
* RECFM='ANY VALID COMBINATION' E.G. F,FB,V,VBA...
* RTDDN='YES'
* UNALC='YES'
* OUTPUT:
  A PARAMETER LIST READY FOR INPUT TO SVC 99.

* SAMPLE INVOCATIONS:
  * ALLOC DSNAM='A.B',STATS='SHR'
  * ALLOC VERB='UN',DSNAM='A.B'
  * ALLOC RENT='YES',DSNAM='A.B.C.D.E',UNIT='SYSALLDA',TRK='YES',
    PRIME='1',SECND='2',DDNAM='TEST1'
  * ALLOC DSNAM=DSNAMADR,STATS=(R8)

* NOTES:
  1 THE 'ALLOCPL' USER MACRO AND 'IEFZB4DØ' AND IEFZB4D2' SYSTEM
     MACROS MUST BE SPECIFIED IN CONJUNCTION WITH THIS MACRO.
  2 IF REENTRANCY IS DESIRED, SPECIFY 'RENT=YES' ON THIS MACRO
     AND CODE THE 'ALLOCPL' MACRO WITHIN A DSECT.
  3 CHECK ADDITIONAL NOTES IN THE 'ALLOCPL' MACRO.

***********************************************************************
PRINT ON
&NAME \ DS ØH
GBLA &ALCMTUN,&ALCMTUL
LCLA &I1,&I2,&STRINGK
LCLC &CHARS

&I1 \ SETA Ø \ NUMBER OF TEXT UNITS
&I2 \ SETA Ø \ LENGTH OF TEXT UNITS
AIF ('&RENT' EQ '').RENT NON-REENTRANT VERSION
AIF ('&RENT' EQ 'YES').RENTY REENTRANT VERSION
MNOTE 8,'INVALID ''RENT'' SPECIFICATION'

© 2001. Xephon UK telephone 01635 33848, fax 01635 38345. USA telephone (303) 410 9344, fax (303) 438 0290.
AGO .RENTE

.RENTY ANOP
LA R1,ALCRB          PLUG VALUES
ST R1,ALCRBPTR
OI ALCRBPTR,X'80'
MVC ALCRB(8),=X'14000000000000'
LA R1,ALCTXTPP
ST R1,ALCTXTPP
MVC ALCFLAG2,=AL4(Ø)

.RENTE ANOP
AIF ('&VERB'(1,1) EQ '(').VERBRG  REGISTER
AIF ('&VERB'(1,1) EQ '****').VERBST  STRING
MVC ALCVERB,&VERB  ADDRESS
AGO .VERBE

.VERBRG ANOP
&STRINGK SETA K'&VERB-2  NO. OF CHARACTERS
&CHARS SETC '&VERB'(2,&STRINGK)
MVC ALCVERB,Ø(&CHARS)
AGO .VERBE

.VERBST ANOP
AIF ('&VERB' EQ '***AL***').VERBSET
AIF ('&VERB' EQ '***UN***').VERBSET
AIF ('&VERB' EQ '***CC***').VERBSET
AIF ('&VERB' EQ '***DC***').VERBSET
AIF ('&VERB' EQ '***RI***').VERBSET
AIF ('&VERB' EQ '***DN***').VERBSET
AIF ('&VERB' EQ '***IN***').VERBSET
MNOTE 8,'INVALID VERB SPECIFICATION'
AGO .VERBE

.Verbset ANOP
&STRINGK SETA K'&VERB-2  NO. OF CHARACTERS
&CHARS SETC '&VERB'(2,&STRINGK)
MVC ALCVERB,S99VRB&CHARS
AGO .VERBE

.Verb ANOP
LA R1,ALCTUS  POINT TO TU AREA

.DDNAME AIF ('&DDNAME' EQ ' ').DDNAME
&I1 SETA &I1+1  INCREMENT TU NUMBER
AIF ('&DDNAME'(1,1) EQ '(').DDNAMRG  REGISTER
AIF ('&DDNAME'(1,1) EQ '****').DDNAMST STRING
LA RØ,&DDNAME  ADDRESS
ST RØ,ALCTUA&I1
AGO .DDNAME

.DDNAME ANOP
&STRINGK SETA K'&DDNAME-2
&CHARS SETC '&DDNAME'(2,&STRINGK)
ST &CHARS,ALCTUA&I1
AGO .DDNAME

.DDNAME ANOP
&STRINGK SETA K'&DDNAME-2
ST R1,ALCTUA&I1  STORE TU ADDRESS
MVC Ø(2,R1),=AL2(DALDDNAM) KEY
MVC 2(2,R1),=AL2(1) NUMBER
MVC 4(2,R1),=AL2(&STRINGK) LENGTH
&CHARS SETC 'DDNAM' (2,&STRINGK) GET THE ACTUAL STRING
MVC 6(&STRINGK,R1),=C'&CHARS' PARM
&I2 SETA &I2+6+&STRINGK LENGTH OF THIS TU
LA R1,(6+&STRINGK)(R1) UPDATE TU ADDRESS

&DDNAM ANOP
&DSNAM AIF ('&DSNAM' EQ '').DSNAME
&I1 SETA &I1+1 INCREMENT TU NUMBER
AIF ('&DSNAM'(1,1) EQ ' ').DSNAMRG REGISTER
AIF ('&DSNAM'(1,1) EQ ' ').DSNAMST STRING
LA R0,&DSNAM ADDRESS
ST R0,ALCTUA&I1
AGO .DSNAME

&DSNAMRG ANOP
&STRINGK SETA K'&DSNAM-2
&CHARS SETC 'DSNAM' (2,&STRINGK)
ST &CHARS,ALCTUA&I1
AGO .DSNAME

&DSNAMST ANOP
&STRINGK SETA K'&DSNAM-2
ST R1,ALCTUA&I1 STORE TU ADDRESS
MVC Ø(2,R1),=AL2(DALDSNAM) KEY
MVC 2(2,R1),=AL2(1) NUMBER
MVC 4(2,R1),=AL2(&STRINGK) LENGTH
&CHARS SETC 'DSNAM' (2,&STRINGK) GET THE ACTUAL STRING
MVC 6(&STRINGK,R1),=C'&CHARS' PARM
&I2 SETA &I2+6+&STRINGK LENGTH OF THIS TU
LA R1,(6+&STRINGK)(R1) UPDATE TU ADDRESS

&DSNAME ANOP
&MEMBR AIF ('&MEMBR' EQ '').MEMBRE
&I1 SETA &I1+1 INCREMENT TU NUMBER
AIF ('&MEMBR'(1,1) EQ ' ').MEMBRRG REGISTER
AIF ('&MEMBR'(1,1) EQ ' ').MEMBRST STRING
LA R0,&MEMBR ADDRESS
ST R0,ALCTUA&I1
AGO .MEMBRE

&MEMBRRG ANOP
&STRINGK SETA K'&MEMBR-2
&CHARS SETC 'MEMBR' (2,&STRINGK)
ST &CHARS,ALCTUA&I1
AGO .MEMBRE

&MEMBRST ANOP
&STRINGK SETA K'&MEMBR-2
ST R1,ALCTUA&I1 STORE TU ADDRESS
MVC Ø(2,R1),=AL2(DALMEMBR) KEY
MVC 2(2,R1),=AL2(1) NUMBER
MVC 4(2,R1),=AL2(&STRINGK) LENGTH
&CHARS SETC 'MEMBR' (2,&STRINGK) GET THE ACTUAL STRING
MVC 6(&STRINGK,R1),=C'&CHARS'       PARM
&I2 SETA &I2+6+&STRINGK LENGTH OF THIS TU
LA R1,(6+&STRINGK)(R1) UPDATE TU ADDRESS
.MEMBRE ANOP
.STATS AIF ('&STATS' EQ '').STATSE
&I1 SETA &I1+1 INCREMENT TU NUMBER
AIF ('&STATS'(1,1) EQ '(').STATSRG REGISTER
AIF ('&STATS'(1,1) EQ '****').STATSS STRING
LA RØ,&STATS ADDRESS
ST RØ,ALCTUA&I1
AGO .STATSE

.STATSRG ANOP
&STRINGK SETA K'&STATS-2
&CHARS SETC '&STATS'(2,&STRINGK)
ST &CHARS,ALCTUA&I1
AGO .STATSE

.STATSS ANOP
ST R1,ALCTUA&I1 STORE TU ADDRESS
MVC Ø(2,R1),=AL2(DALSTATS) KEY
MVC 2(2,R1),=AL2(1) NUMBER
MVC 4(2,R1),=AL2(1) LENGTH
AIF ('&STATS' EQ '***OLD***').STATSO
AIF ('&STATS' EQ '***MOD***').STATSM
AIF ('&STATS' EQ '***NEW***').STATSN
AIF ('&STATS' EQ '***SHR***').STATSS
MNOTE 8,'INVALID "STATS" SPECIFICATION'
AGO .STATSE

.STATSO ANOP
MVI 6(R1),X'Ø1'
AGO .STATSU1

.STATSM ANOP
MVI 6(R1),X'Ø2'
AGO .STATSU1

.STATSN ANOP
MVI 6(R1),X'Ø4'
AGO .STATSU1

.STATSS ANOP
MVI 6(R1),X'Ø8'
AGO .STATSU1

.STATSU1 ANOP
&I2 SETA &I2+7 LENGTH OF THIS TU
LA R1,7(R1) UPDATE TU ADDRESS

.STATSE ANOP
.NDISP AIF ('&NDISP' EQ '').NDISPE
&I1 SETA &I1+1 INCREMENT TU NUMBER
AIF ('&NDISP'(1,1) EQ '(').NDISPRG REGISTER
AIF ('&NDISP'(1,1) EQ '****').NDISPSST STRING
LA RØ,&NDISP ADDRESS
ST RØ,ALCTUA&I1
AGO .NDISPE
...
AIF ('&CDISP' EQ '''CATLG''').CDISPC
AIF ('&CDISP' EQ '''DELETE''').CDISPD
AIF ('&CDISP' EQ '''KEEP''').CDISPK

MNOTE 8,'INVALID "CDISP" SPECIFICATION'
AGO .CDISPE

.CDISPU ANOP
MVI 6(R1),X'Ø1'
AGO .CDISPU1

.CDISPC ANOP
MVI 6(R1),X'Ø2'
AGO .CDISPU1

.CDISPD ANOP
MVI 6(R1),X'Ø4'
AGO .CDISPU1

.CDISPK ANOP
MVI 6(R1),X'Ø8'
AGO .CDISPU1

.CDISPU1 ANOP
&I2 SETA &I2+7          LENGTH OF THIS TU
   LA R1,7(R1)          UPDATE TU ADDRESS

.CDISPE ANOP

.UNIT AIF ('&UNIT' EQ '').UNITE
&I1 SETA &I1+1          INCREMENT TU NUMBER
AIF ('&UNIT'(1,1) EQ ').UNITRG REGISTER
AIF ('&UNIT'(1,'"'1) EQ '''').UNITST STRING
   LA R0,&UNIT          ADDRESS
   ST R0,ALCTUA&I1
   AGO .UNITE

.UNITRG ANOP
&STRINGK SETA K'&UNIT-2
&CHARS SETC '&UNIT'(2,&STRINGK)
   ST &CHARS,ALCTUA&I1
   AGO .UNITE

.UNITST ANOP
&STRINGK SETA K'&UNIT-2
   ST R1,ALCTUA&I1       STORE TU ADDRESS
   MVC Ø(2,R1),=AL2(DALUNIT) KEY
   MVC 2(2,R1),=AL2(1)    NUMBER
   MVC 4(2,R1),=AL2(&STRINGK) LENGTH
   &CHARS SETC '&UNIT'(2,&STRINGK) GET THE ACTUAL STRING
   MVC 6(&STRINGK,R1),=C'&CHARS'  PARM
   &I2 SETA &I2+6+&STRINGK LENGTH OF THIS TU
   LA R1,(6+&STRINGK)(R1) UPDATE TU ADDRESS

.UNITE ANOP
.SYSOU AIF ('&SYSOU' EQ ' ').SYSOUE
&I1 SETA &I1+1          INCREMENT TU NUMBER
AIF ('&SYSOU'(1,1) EQ ').SYSOURG REGISTER
AIF ('&SYSOU'(1,1) EQ '''').SYSOUST STRING
   LA R0,&SYSOU          ADDRESS
   ST R0,ALCTUA&I1
AGO .SYSOUE

&STRINGK SETA K'&SYSOU-2
&CHARS SETC '&SYSOU'(2,&STRINGK)
ST &CHARS,ALCTUA&I1
AGO .SYSOUE

&STRINGK SETA K'&SYSOU-2
ST R1,ALCTUA&I1
MVC Ø(2,R1),=AL2(DALSYSOU) KEY
MVC 2(2,R1),=AL2(1) NUMBER
MVC 4(2,R1),=AL2(&STRINGK) LENGTH
&CHARS SETC '&SYSOU'(2,&STRINGK) GET THE ACTUAL STRING
MVC 6(&STRINGK,R1),='&CHARS' PARM
&I2 SETA &I2+6+&STRINGK LENGTH OF THIS TU
LA R1,(6+&STRINGK)(R1) UPDATE TU ADDRESS

&SYSOUE ANOP
.SPGNM AIF ('&SPGNM' EQ '').SPGNME
&I1 SETA &I1+1 INCREMENT TU NUMBER
AIF ('&SPGNM'(1,1) EQ '(').SPGNMRG REGISTER
AIF ('&SPGNM'(1,1) EQ '''').SPGNMST STRING
LA RØ,&SPGNM ADDRESS
ST RØ,ALCTUA&I1
AGO .SPGNME

&SPGNMRG ANOP
&STRINGK SETA K'&SPGNM-2
&CHARS SETC '&SPGNM'(2,&STRINGK)
ST &CHARS,ALCTUA&I1
AGO .SPGNME

&SPGNMST ANOP
&STRINGK SETA K'&SPGNM-2
ST R1,ALCTUA&I1
MVC Ø(2,R1),=AL2(DALSPGNM) KEY
MVC 2(2,R1),=AL2(1) NUMBER
MVC 4(2,R1),=AL2(&STRINGK) LENGTH
&CHARS SETC '&SPGNM'(2,&STRINGK) GET THE ACTUAL STRING
MVC 6(&STRINGK,R1),='&CHARS' PARM
&I2 SETA &I2+6+&STRINGK LENGTH OF THIS TU
LA R1,(6+&STRINGK)(R1) UPDATE TU ADDRESS

&SPGNME ANOP
.CLOSE AIF ('&CLOSE' EQ '').CLOSEE
&I1 SETA &I1+1 INCREMENT TU NUMBER
AIF ('&CLOSE'(1,1) EQ '(').CLOSERG REGISTER
AIF ('&CLOSE'(1,1) EQ '''').CLOSEST LITERAL
LA RØ,&CLOSE ADDRESS
ST RØ,ALCTUA&I1
AGO .CLOSEE

.CLOSERG ANOP
&STRINGK SETA K'&CLOSE-2
&CHARS SETC '&CLOSE'(2,&STRINGK)
ST &CHARS,ALCTUA&I1
AGO .CLOSEE

.CLOSEST ANOP
AIF ('&CLOSE' EQ '''YES''').CLOSEY
MNOTE 8,"INVALID "CLOSE" SPECIFICATION"
AGO .CLOSEE

.CLOSEY ANOP

&STRINGK SETA Ø
ST R1,ALCTUA&I1                     STORE TU ADDRESS
MVC Ø(2,R1),=AL2(DALCLOSE)           KEY
MVC 2(2,R1),=AL2(Ø)                  NUMBER
MVC 4(2,R1),=AL2(Ø)                  LENGTH
&I2 SETA &I2+4                            LENGTH OF THIS TU
LA R1,(4)(R1)                       UPDATE TU ADDRESS

.SUSER AIF ('&SUSER' EQ '').SUSERE
&I1 SETA &I1+1                            INCREMENT TU NUMBER
AIF ('&SUSER'(1,1) EQ '').SUSERRG REGISTER
AIF ('&SUSER'(1,1) EQ '****').SUSERST STRING
LA RØ,&SUSER                        ADDRESS
ST RØ,ALCTUA&I1                     STORE TU ADDRESS
AGO .SUSERE

.SUSERRG ANOP
&STRINGK SETA K'&SUSER-2
&CHARS SETC '&SUSER'(2,&STRINGK)
ST &CHARS,ALCTUA&I1
AGO .SUSERE

.SUSERST ANOP
&STRINGK SETA K'&SUSER-2
ST R1,ALCTUA&I1                     STORE TU ADDRESS
MVC Ø(2,R1),=AL2(DALSUSER)           KEY
MVC 2(2,R1),=AL2(1)                  NUMBER
MVC 4(2,R1),=AL2(2&STRINGK)          LENGTH
&CHARS SETC '&SUSER'(2,&STRINGK)      GET THE ACTUAL STRING
MVC 6(&STRINGK,R1),=C'&CHARS'        PARM
&I2 SETA &I2+6+&STRINGK              LENGTH OF THIS TU
LA R1,(6+&STRINGK)(R1)              UPDATE TU ADDRESS

.SUSERE ANOP

.SFMNO AIF ('&SFMLNO' EQ '').SFMLNOE
&I1 SETA &I1+1                            INCREMENT TU NUMBER
AIF ('&SFMLNO'(1,1) EQ ' ').SFMLNOR REGISTER
AIF ('&SFMLNO'(1,1) EQ '****').SFMLNOST STRING
LA RØ,&SFMLNO                        ADDRESS
ST RØ,ALCTUA&I1                     STORE TU ADDRESS
AGO .SFMLNOE

.SFMNORG ANOP
&STRINGK SETA K'&SFMLNO-2
&CHARS SETC '&SFMLNO'(2,&STRINGK)
ST &CHARS,ALCTUA&I1
AGO .SFMLNOE
.SFNMNST ANOP
&STRINGK SETA K'SFNMNO-2
  ST R1,ALCTUA&I1
  MVC Ø(2,R1),=AL2(DALSFMNO)
  MVC 2(2,R1),=AL2(1)
  MVC 4(2,R1),=AL2(&STRINGK)
&CHARS SETC '&SFNMNO'(2,&STRINGK)
  MVC 6(&STRINGK,R1),=C'&CHARS'
&I2 SETA &I2+6+&STRINGK
  LA R1,(6+&STRINGK)(R1)
.SFMNOE ANOP
.OUTPT AIF ('&OUTPT' EQ '').OUTPTE
  &I1 SETA &I1+1
  AIF ('&OUTPT'(1,1) EQ ' ').OUTPTRG
  AIF ('&OUTPT'(1,1) EQ '***').OUTPTST
  LA RØ,&OUTPT
  ST RØ,ALCTUA&I1
  AGO .OUTPTE
.OUTPTRG ANOP
&STRINGK SETA K'&OUTPT-2
&CHARS SETC '&OUTPT'(2,&STRINGK)
  ST &CHARS,ALCTUA&I1
  AGO .OUTPTE
.OUTPTST ANOP
&STRINGK SETA K'&OUTPT-2
  ST R1,ALCTUA&I1
  MVC Ø(2,R1),=AL2(DALOUTPT)
  MVC 2(2,R1),=AL2(1)
  MVC 4(2,R1),=AL2(&STRINGK)
&CHARS SETC '&OUTPT'(2,&STRINGK)
  MVC 6(&STRINGK,R1),=C'&CHARS'
&I2 SETA &I2+6+&STRINGK
  LA R1,(6+&STRINGK)(R1)
.OUTPTE ANOP
.VLSER AIF ('&VLSER' EQ '').VLSERE
  &I1 SETA &I1+1
  AIF ('&VLSER'(1,1) EQ ' ').VLSERRG
  AIF ('&VLSER'(1,1) EQ '***').VLSERST
  LA RØ,&VLSER
  ST RØ,ALCTUA&I1
  AGO .VLSERE
.VLSERRG ANOP
&STRINGK SETA K'&VLSER-2
&CHARS SETC '&VLSER'(2,&STRINGK)
  ST &CHARS,ALCTUA&I1
  AGO .VLSERE
.VLSERST ANOP
&STRINGK SETA K'&VLSER-2
  ST R1,ALCTUA&I1
  MVC Ø(2,R1),=AL2(DALVLSER)
  KEY
MVC 2(2,R1),=AL2(1) NUMBER
MVC 4(2,R1),=AL2(&STRINGK) LENGTH
&CHARS SETC '&VLSE' (2,&STRINGK) GET THE ACTUAL STRING
MVC 6(&STRINGK,R1),=C'&CHARS' PARM
&I2 SETA &I2+6+&STRINGK LENGTH OF THIS TU
LA R1,(6+&STRINGK)(R1) UPDATE TU ADDRESS
.AVLSE ANOP
.RTDDN AIF ('&RTDDN' EQ '').RTDDNE &I1 SETA &I1+1 INCREMENT TU NUMBER
AIF ('&RTDDN' (1,1) EQ '(').RTDDNRG REGISTER
AIF ('&RTDDN' (1,1) EQ '***').RTDDNST LITERAL
LA R0,&RTDDN ADDRESS
ST R0,ALCTUA&I1
AGO .RTDDNE
.RTDDNRG ANOP
&STRINGK SETA K'&RTDDN-2
&CHARS SETC '&RTDDN'(2,&STRINGK)
ST &CHARS,ALCTUA&I1
AGO .RTDDNE
.RTDDNST ANOP
AIF ('&RTDDN' EQ '***YES***').RTDDNY MNOTE 8,'INVALID "RTDDN" SPECIFICATION'
AGO .RTDDNE
.RTDDNY ANOP
&STRINGK SETA 0
ST R1,ALCTUA&I1 STORE TU ADDRESS
MVC 0(2,R1),=AL2(DALRTDDN) KEY
MVC 2(2,R1),=AL2(1) NUMBER
MVC 4(2,R1),=AL2(8) LENGTH
&I2 SETA &I2+6+8 LENGTH OF THIS TU
LA R1,(6+8)(R1) UPDATE TU ADDRESS
.RTDDNE ANOP
.TRK AIF ('&TRK' EQ '').TRKE &I1 SETA &I1+1 INCREMENT TU NUMBER
AIF ('&TRK' (1,1) EQ '(').TRKRG REGISTER
AIF ('&TRK' (1,1) EQ '****').TRKST LITERAL
LA R0,&TRK ADDRESS
ST R0,ALCTUA&I1
AGO .TRKE
.TRKRG ANOP
&STRINGK SETA K'&TRK-2
&CHARS SETC '&TRK'(2,&STRINGK)
ST &CHARS,ALCTUA&I1
AGO .TRKE
.TRKST ANOP
AIF ('&TRK' EQ '***YES***').TRKY MNOTE 8,'INVALID "TRK" SPECIFICATION'
AGO .TRKE
.TRKY ANOP
&STRINGK SETA 0
```
ST R1,ALCTUA&I1
MVC Ø(2,R1),=AL2(DALTRK)
MVC 2(2,R1),=AL2(Ø)
&I2 SETA &I2+4
LA R1,(4)(R1)
.TRKE ANOP
.CYL AIF ('&CYL' EQ ' ').CYLE
&I1 SETA &I1+1
AIF ('&CYL'(1,1) EQ '(').CYLRG
AIF ('&CYL'(1,1) EQ ' ' ').CYLST
LA RØ,&CYL
ST RØ,ALCTUA&I1
AGO .CYLE
.CYLRG ANOP
&STRINGK SETA K'&CYL-2
&CHARS SETC '&CYL'(2,&STRINGK)
ST &CHARS,ALCTUA&I1
AGO .CYLE
.CYLST ANOP
AIF ('&CYL' EQ 'YES').CYLY
MNOTE 8,'INVALID ''CYL'' SPECIFICATION'
AGO .CYLE
.CYLY ANOP
&STRINGK SETA Ø
ST R1,ALCTUA&I1
MVC Ø(2,R1),=AL2(DALCYL)
MVC 2(2,R1),=AL2(Ø)
&I2 SETA &I2+4
LA R1,(4)(R1)
.UNALC AIF ('&UNALC' EQ ' ').UNALCE
&I1 SETA &I1+1
AIF ('&UNALC'(1,1) EQ '(').UNALCRG
AIF ('&UNALC'(1,1) EQ ' ' ').UNALCST
LA RØ,&UNALC
ST RØ,ALCTUA&I1
AGO .UNALCE
.UNALCRG ANOP
&STRINGK SETA K'&UNALC-2
&CHARS SETC '&UNALC'(2,&STRINGK)
ST &CHARS,ALCTUA&I1
AGO .UNALCE
.UNALCST ANOP
AIF ('&UNALC' EQ 'YES').UNALCY
MNOTE 8,'INVALID ''UNALC'' SPECIFICATION'
AGO .UNALCE
.UNALCY ANOP
&STRINGK SETA Ø
ST R1,ALCTUA&I1
MVC Ø(2,R1),=AL2(DUNUNALC)
```
MVC 2(2,R1),=AL2(Ø)       NUMBER
&I2 SETA &I2+4             LENGTH OF THIS TU
LA R1,(4)(R1)             UPDATE TU ADDRESS
.UNALCE ANOP

.PRIME AIF ('&PRIME' EQ '').PRIMEE
&I1 SETA &I1+1             INCREMENT TU NUMBER
AIF ('&PRIME'(1,1) EQ '(').PRIMERG REGISTER
AIF ('&PRIME'(1,1) EQ '****').PRIMEST STRING
LA RØ,&PRIME              ADDRESS
ST RØ,ALCTUA&I1
AGO .PRIMEE

.PRIMERG ANOP
&STRINGK SETA K'&PRIME-2
&CHARS SETC '&PRIME'(2,&STRINGK)
ST &CHARS,ALCTUA&I1
AGO .PRIMEE

.PRIMEST ANOP
&STRINGK SETA K'&PRIME-2
ST R1,ALCTUA&I1           STORE TU ADDRESS
MVC Ø(2,R1),=AL2(DALPRIME) KEY
MVC 2(2,R1),=AL2(1)       NUMBER
MVC 4(2,R1),=AL2(3)       LENGTH
&CHARS SETC '&PRIME'(2,&STRINGK) GET THE ACTUAL STRING
MVC 6(3,R1),=AL3(&CHARS)  PARM
&I2 SETA &I2+6+3          LENGTH OF THIS TU
LA R1,(6+3)(R1)           UPDATE TU ADDRESS

.PRIMEE ANOP

.SECND AIF ('&SECND' EQ '').SECNDE
&I1 SETA &I1+1             INCREMENT TU NUMBER
AIF ('&SECND'(1,1) EQ '(').SECNDRG REGISTER
AIF ('&SECND'(1,1) EQ '****').SECNDST STRING
LA RØ,&SECND              ADDRESS
ST RØ,ALCTUA&I1
AGO .SECNDE

.SECNDRG ANOP
&STRINGK SETA K'&SECND-2
&CHARS SETC '&SECND'(2,&STRINGK)
ST &CHARS,ALCTUA&I1
AGO .SECNDE

.SECNDST ANOP
&STRINGK SETA K'&SECND-2
ST R1,ALCTUA&I1           STORE TU ADDRESS
MVC Ø(2,R1),=AL2(DALSECND) KEY
MVC 2(2,R1),=AL2(1)       NUMBER
MVC 4(2,R1),=AL2(3)       LENGTH
&CHARS SETC '&SECND'(2,&STRINGK) GET THE ACTUAL STRING
MVC 6(3,R1),=AL3(&CHARS)  PARM
&I2 SETA &I2+6+3          LENGTH OF THIS TU
LA R1,(6+3)(R1)           UPDATE TU ADDRESS

.SECNDE ANOP
.DIR  AIF  ('&DIR' EQ '').DIRE
&I1  SETA  &I1+1                    INCREMENT TU NUMBER
AIF  ('&DIR'(1,1) EQ ' ').DIRRG  REGISTER
AIF  ('&DIR'(1,1) EQ ' *').DIRST  STRING
LA  RØ,&DIR                      ADDRESS
ST  RØ,ALCTUA&I1
AGO  .DIRE

.DIRRG  ANOP
&STRINGK  SETA  K'&DIR-2
&CHARS  SETC  '&DIR'(2,&STRINGK)
ST  &CHARS,ALCTUA&I1
AGO  .DIRE

.DIRST  ANOP
&STRINGK  SETA  K'&DIR-2
ST  R1,ALCTUA&I1
MVC  Ø(2,R1),=AL2(DALDIR)     KEY
MVC  2(2,R1),=AL2(1)         NUMBER
MVC  4(2,R1),=AL2(3)         LENGTH
&CHARS  SETC  '&DIR'(2,&STRINGK)
MVC  6(3,R1),=AL3(&CHARS)    PARM
&I2  SETA  &I2+6+3               LENGTH OF THIS TU
LA  R1,(6+3)(R1)             UPDATE TU ADDRESS
AGO  .DIR

.RLSE  AIF  ('&RLSE' EQ '').RLSEE
&I1  SETA  &I1+1                    INCREMENT TU NUMBER
AIF  ('&RLSE'(1,1) EQ ' ').RLSERG  REGISTER
AIF  ('&RLSE'(1,1) EQ ' *').RLSEST  LITERAL
LA  RØ,&RLSE                    ADDRESS
ST  RØ,ALCTUA&I1
AGO  .RLSEE

.RLSERG  ANOP
&STRINGK  SETA  K'&RLSE-2
&CHARS  SETC  '&RLSE'(2,&STRINGK)
ST  &CHARS,ALCTUA&I1
AGO  .RLSEE

.RLSEST  ANOP
AIF  ('&RLSE' EQ ' *YES*').RLSEY
MNOTE  8,'INVALID 'RLSE'' SPECIFICATION'
AGO  .RLSEE

.RLSEY  ANOP
&STRINGK  SETA  Ø
ST  R1,ALCTUA&I1
MVC  Ø(2,R1),=AL2(DALRLSE)     KEY
MVC  2(2,R1),=AL2(Ø)         NUMBER
MVC  4(2,R1),=AL2(Ø)         LENGTH
&I2  SETA  &I2+4               LENGTH OF THIS TU
LA  R1,(4)(R1)             UPDATE TU ADDRESS
AGO  .RLSEE

.BLKSZ  AIF  ('&BLKSZ' EQ '').BLKSZE
&I1  SETA  &I1+1                    INCREMENT TU NUMBER
AIF ('&BLKSZ'(1,1) EQ '(').BLKSRG REGISTER
AIF ('&BLKSZ'(1,1) EQ '***').BLKSRST STRING
LA R0,&BLKSZ
ST R0,ALCTUA&I1
AGO .BLKSZE
.BLKSRG ANOP
&STRINGK SETA K'&BLKSZ-2
&CHARS SETC '&BLKSZ'(2,&STRINGK)
ST &CHARS,ALCTUA&I1
AGO .BLKSZE
.BLKSRST ANOP
&STRINGK SETA K'&BLKSZ-2
ST R1,ALCTUA&I1
MVC Ø(2,R1),=AL2(DALBLKSZ)
MVC 2(2,R1),=AL2(1)
MVC 4(2,R1),=AL2(2)
&CHARS SETC '&BLKSZ'(2,&STRINGK)
MVC 6(2,R1),=AL2(&CHARS)
&I2 SETA &I2+6+2
LA R1,(6+2)(R1)
AGO .DSORGE
.DSORGE
&STRINGK SETA K'&DSORG-2
&CHARS SETC '&DSORG'(2,&STRINGK)
ST &CHARS,ALCTUA&I1
AGO .DSORGE
.DSORGE
&STRINGK SETA K'&DSORG-2
ST R1,ALCTUA&I1
MVC Ø(2,R1),=AL2(DALDSORG)
MVC 2(2,R1),=AL2(1)
MVC 4(2,R1),=AL2(2)
AIF ('&DSORG' EQ '***PS***').DSORGPS
AIF ('&DSORG' EQ '***PO***').DSORGPO
AIF ('&DSORG' EQ '***DA***').DSORGDA
MNOTE 8,'INVALID ''DSORG'' SPECIFICATION'
AGO .DSORGE
.DSORGE
MVC 6(2,R1),=X'4ØØØ'
AGO .DSORGUI
.DSORGUI
MVC 6(2,R1),=X'Ø2ØØ'
AGO .DSORGUI
.DSORGDA ANOP
  MVC 6(2,R1),=X'2000'
  AGO .DSORGUI

&I2 SETA &I2+6+2
  LA R1,(6+2)(R1)

.LRECL AIF ('&LRECL' EQ '').LRECLE
  &I1 SETA &I1+1
  AIF ('&LRECL'(1,1) EQ '(').LRECLR
  AIF ('&LRECL'(1,1) EQ '****').LRECLST
  LA R0,&LRECL
  ST R0,ALCTUA&I1
  AGO .LRECLE

.LRECLR ANOP
  &STRINGK SETA K'&LRECL-2
  &CHARS SETC '&LRECL'(2,&STRINGK)
  ST &CHARS,ALCTUA&I1
  AGO .LRECLE

.LRECLST ANOP
  &STRINGK SETA K'&LRECL-2
  ST R1,ALCTUA&I1
  MVC Ø(2,R1),=AL2(DALLRECL)
  MVC 2(2,R1),=AL2(1)
  MVC 4(2,R1),=AL2(2)
  &CHARS SETC '&LRECL'(2,&STRINGK)
  MVC 6(2,R1),=AL2(&CHARS)
  &I2 SETA &I2+6+2
  LA R1,(6+2)(R1)

.LRECLE ANOP
  .RECFM AIF ('&RECFM' EQ '').RECFME
  &I1 SETA &I1+1
  AIF ('&RECFM'(1,1) EQ '(').RECFMR
  AIF ('&RECFM'(1,1) EQ '****').RECFMST
  LA R0,&RECFM
  ST R0,ALCTUA&I1
  AGO .RECFME

.RECFMR ANOP
  &STRINGK SETA K'&RECFM-2
  &CHARS SETC '&RECFM'(2,&STRINGK)
  ST &CHARS,ALCTUA&I1
  AGO .RECFME

.RECFMST ANOP
  &STRINGK SETA K'&RECFM-2
  ST R1,ALCTUA&I1
  MVC Ø(2,R1),=AL2(DALRECFM)
  MVC 2(2,R1),=AL2(1)
  MVC 4(2,R1),=AL2(1)
  MVI 6(R1),Ø
  AIF ('&RECFM'(2,1) EQ 'F').RECFM2F
AIF ('&RECFM'(2,1) EQ 'V').RECFM2V
AIF ('&RECFM'(2,1) EQ 'U').RECFM2U
MNOTE 8,'INVALID 'RECFM' SPECIFICATION'
AGO .RECFME

.RECFM2F ANOP
   OI  6(R1),X'80'
   AGO .RECFM2E

.RECFM2V ANOP
   OI  6(R1),X'40'
   AGO .RECFM2E

.RECFM2U ANOP
   OI  6(R1),X'C0'
   AGO .RECFM2E

.RECFM2E ANOP
   AIF ('&RECFM'(3,1) EQ 'S').RECFM3S
   AIF ('&RECFM'(3,1) EQ 'B').RECFM3B
   AIF ('&RECFM'(3,1) EQ 'A').RECFM3A
   AIF ('&RECFM'(3,1) EQ 'M').RECFM3M
   AIF ('&RECFM'(3,1) EQ '##').RECFMU1
   MNOTE 8,'INVALID 'RECFM' SPECIFICATION'
   AGO .RECFME

.RECFM3S ANOP
   OI  6(R1),X'08'
   AGO .RECFM3E

.RECFM3B ANOP
   OI  6(R1),X'10'
   AGO .RECFM3E

.RECFM3A ANOP
   OI  6(R1),X'04'
   AGO .RECFM3E

.RECFM3M ANOP
   OI  6(R1),X'02'
   AGO .RECFM3E

.RECFM3E ANOP
   AIF ('&RECFM'(4,1) EQ 'S').RECFM4S
   AIF ('&RECFM'(4,1) EQ 'A').RECFM4A
   AIF ('&RECFM'(4,1) EQ 'M').RECFM4M
   AIF ('&RECFM'(4,1) EQ '##').RECFMU1
   MNOTE 8,'INVALID 'RECFM' SPECIFICATION'
   AGO .RECFME

.RECFM4S ANOP
   OI  6(R1),X'08'
   AGO .RECFM4E

.RECFM4A ANOP
   OI  6(R1),X'04'
   AGO .RECFM4E

.RECFM4M ANOP
   OI  6(R1),X'02'
   AGO .RECFM4E

.RECFM4E ANOP
ALLOCPL

&NAME ALLOCPL
PRINT OFF

***********************************************************************
*  NAME: ALLOCPL  
*  DESCRIPTION:  
*  RESERVE STORAGE FOR, OR MAP, AN SVC99 PARAMETER LIST.  
*  
*  OUTPUT:  
*  A PARAMETER LIST READY FOR INPUT TO SVC 99.  
*  
*  NOTES:  
*  1  THE 'ALLOC' USER MACRO AND 'IEFZB4D0' AND IEFZB4D2' SYSTEM  
*  MACROS MUST BE SPECIFIED IN CONJUNCTION WITH THIS MACRO.  
*  2  THIS MACRO GENERATES LABEL NAMES SIMILAR TO THOSE OF THE  
*  'IEFZB4D0' MACRO EXCEPT THAT THE FIRST THREE CHARACTERS OF  
*  EACH NAME ARE 'ALC' RATHER THAN 'S99'.  
*  3  THE LENGTHS OF THE TEXT UNIT POINTER LIST AND TEXT UNITS  
*  ARE GOVERNED BY THE 'GBLA' VARIABLES SPECIFIED BELOW, THE  
*  VALUES OF WHICH ARE COMPUTED IN THE COURSE OF THE EXPANSION  
*  OF THE 'ALLOC' MACRO.  THESE LENGTHS ARE EQUAL TO THOSE  
*  COMPUTED BY THE 'ALLOC' MACRO EXPANSIONS WHICH GENERATE  
*  THE GREATEST LENGTHS.  ALL 'ALLOC' MACROS USE THE SAME  
*  TUPL AND TU AREAS TO CONSERVE STORAGE.  
***********************************************************************

PRINT ON
LCLA &I
GBLA &ALCMTUN,&ALCMTUL
**DYNAMIC ALLOCATION REQUEST BLOCK POINTER**

```
&NAME DS ØF             FULLWORD ALIGNMENT
ALCRBPTR DC X'8Ø',AL3(ALCRB) REQUEST BLOCK POINTER
```

**DYNAMIC ALLOCATION REQUEST BLOCK**

```
ALCRB DS ØF             REQUEST BLOCK
ALCRBLS DC AL1(2Ø)     LENGTH OF REQUEST BLOCK
ALCVERB DC AL1(ØØ)     VERB CODE
ALCFLAG1 DS ØAL2       FLAGS
ALCFLAG11 DC AL1(ØØ)   FIRST FLAGS BYTE
ALCFLAG12 DC AL1(ØØ)   SECOND BYTE OF FLAGS
ALCRSC DS ØAL4         REASON CODE FIELDS
ALCERROR DC AL2(ØØ)    ERROR REASON CODE
ALCINFO DC AL2(ØØ)     INFORMATION REASON CODE
ALCTXTPP DC A(ALCTXTPL) ADDR OF LIST OF TEXT UNIT PTRS
ALCRSVØ1 DS F          RESERVED
ALCFLAGG2 DS ØAL4      FLAGS FOR AUTHORIZED FUNCTIONS
ALCFLAG21 DC AL1(ØØ)   FIRST BYTE OF FLAGS
ALCFLAG22 DS AL1(ØØ)   SECOND BYTE OF FLAGS
ALCFLAG23 DS AL1(ØØ)   THIRD BYTE OF FLAGS
ALCFLAG24 DS AL1(ØØ)   FOURTH BYTE OF FLAGS
```

**DYNAMIC ALLOCATION TEXT UNIT POINTER LIST**

```
ALCTXTPL DS ØF
&I       SETA Ø
.LOOP1   ANOP
&I       SETA &I+1
AIF    (&I GT &ALCMTUN).ALCMTUE
ALCTXUA&I DC A(Ø)       TEXT UNIT ADDRESS - PLUGGED BY
AGO    .LOOP1           ALLOC MACRO AT EXECUTION TIME
.ALCMTUE ANOP
```

**DYNAMIC ALLOCATION TEXT UNITS**

```
ALCMTUS DS ØF
DS &ALCMTUL.C
MEND
```

 Systems Programmer (Canada)  © Xephon 2001
Dynamic Channel-path Management

INTRODUCTION

A new component of z/OS is the Intelligent Resource Director (IRD). Within the IRD a new area of functionality is Dynamic Channel-path Management, which is designed to adjust the channel configuration dynamically in response to shifting workload patterns. The article provides a brief overview of some of the advantages we have found in having Dynamic Channel-path Management.

To run Dynamic Channel-path Management successfully you need to be running on an IBM zSeries 900, with z/OS 1.1 or higher in z/Architecture mode. Both LPAR and Basic modes of operation are supported. It is possible to share managed channels among z/OS images on the same CPC, but the images must be members of the same LPAR Cluster, so they need to be in the same Parallel Sysplex, and must all be running z/OS 1.1 or above in z/Architecture mode. WLM can be in either Goal mode or Compatibility mode. If WLM is in Goal mode, there is greater benefit from Dynamic Channel-path Management, but this is not a requirement. Dynamic Channel-path Management only supports DASD control units that operate completely non-synchronously and are attached via ESCON or FICON Bridge (FCV) channels. In addition, because Dynamic Channel-path Management works by adding paths to a control unit, the control unit obviously has to support multiple paths. This effectively limits you to DASD and tape.

WHY USE DYNAMIC CHANNEL-PATH MANAGEMENT?

The most important benefit of running Dynamic Channel-path Management is the increase in I/O performance. However, benefits can also be obtained through simplification of I/O configuration definition, a reduction in the skills required to manage z/OS, enhanced availability, and a reduction in the need for more than 256 channels. Some of these issues are considered below.
**Improved I/O performance**

Dynamic Channel-path Management works by dynamically moving the available channel bandwidth to where it is needed the most. With OS/390, users needed to balance their available channels across their I/O devices. They had to try to provide sufficient paths to handle the average load on each controller. This was quite an art, but, even so, changes in system use often meant that there were some controllers with more I/O paths available than required, while other controllers possibly had too few.

Dynamic Channel-path Management attempts to balance the responsiveness of the available channels by moving channels to the controllers that require additional bandwidth. This can be done even when the system is in WLM Compatibility mode.

Systems programmers no longer need to remember different rules of thumb about how busy you should run your channels for every channel or control unit type that is installed. Instead, you just need to monitor for the signs of channel over-utilization (high channel utilization combined with high Pend times). You can now do this quite easily with RMF, which provides a report which shows the average aggregate utilization for all managed channels. This really saves time.

**Simplified configuration**

Dynamic Channel-path Management also simplifies the task of defining your configuration. With OS/390 the process of configuration definition was quite time consuming and complicated. Operators needed to decide how many paths were required for each CU for acceptable performance. Then they needed to decide which CUs should share channels (usually by identifying ones that are busy at different times). Then it was necessary to decide which paths to use for each CU to balance utilization, then select paths that minimize points of failure. Up to eight paths had to be defined for each CU monitor and these needed to be tuned on an ongoing basis. With Dynamic Channel-path Management the configuration definition process is much simpler. Operators just need to estimate the maximum channel bandwidth required to handle the workload on the managed CUs at the peak time. Then they need to define at least two non-managed paths and the maximum number of managed paths you are likely to be needed for each CU.
**Time savings**

Because Dynamic Channel-path Management is essentially self-tuning, it is possible for systems programmers to spend much less time dealing with configuration management, monitoring, and capacity planning. However, you still need to understand the performance and availability characteristics of all your installed hardware.

**Maximize utilization of installed resources**

Dynamic Channel-path Management can increase throughput to DASD subsystems without the need for extra channels. Theoretically it is possible to increase the overall average utilization of the channels, without heavily impacting the response time for the connected subsystems.

Another advantage of Dynamic Channel-path Management is that every time you install a new device, you do not have to worry about reconfiguring your paths to match its characteristics. It should be sufficient to follow the connectivity recommendations provided in the installation documents associated with the device when you are designing the physical connectivity.

Because Dynamic Channel-path Management dynamically adjusts to changing workloads, if you happen to end up with multiple busy control units on a channel, which is something that we encounter a lot and which would have resulted in degraded performance under OS/390, Dynamic Channel-path Management reacts to the increased pend time by adding idle or less busy channels to the control units that are suffering high pend times.

**CONCLUSIONS**

Dynamic Channel-path Management is a significant new element of z/OS. Because the system can now move I/O paths dynamically to the LCUs that are experiencing channel delays, it is possible to reduce the CU/channel-level capacity planning and balancing activity that was necessary in the days of OS/390. In the next edition of *MVS Update* we will consider some of the hardware and software planning issues that are required to implement Dynamic Channel-path Management successfully.
IBM has nearly doubled the performance of its z900 and upgraded its z/OS Version 1 release 2 to extend the Intelligent Resource Director to sites running z/VM and Linux on the mainframe.

Other new and useful functionality includes the z/OS Intrusion Detection Services software. This scans incoming data, gives early warning of potential threats, and provides added protection against flooding and denial of service attacks, which are a highly prevalent security issue in the enterprise. Security is also addressed in the hardware with the PCI Cryptographic Accelerator Card, which provides SSL (Secure Sockets Layer) performance of up to 3,850 secure transactions per second.

Other z900 hardware features of interest include Capacity Upgrade on Demand for z900, which has been enhanced so that concurrent memory upgrades can be installed without disruption or changing hardware. Other Capacity back-up is designed to allow a capacity back-up (or disaster recovery) server to be returned to its normal configuration without outage or disruption when it’s no longer needed.

z/VM Release 4.2 extends virtualization capabilities to the new features of the z900, including HiperSockets, OSA Express Token-Ring, FICON, and TCP/IP. Networking and I/O enhancements include FICON Express support and OSA-Express Token Ring supporting up to 100Mb/s. A new C++ compiler and Unix file system are said to improve portability and performance of Unix applications running under z/OS.
The Advanced Software Products Group has announced the Version 5 of its MegaCryption cryptographic product for MVS. MegaCryption provides encryption/decryption, signing, and integrity-checking in one utility. It incorporates many industry-compliant algorithms: DES, Triple-DES, Blowfish, Cast, AES, DSA, MD5, SHA, etc, and is partially interoperable with PGP and GnuPG. Keys may be stored into the RACF database.

There are three new features in this version. The cryptographic hardware coprocessor is supported; this makes hardware encryption usable in areas that have been unexplored until now, such as DFSMSdss backups, and dataset sealing, etc. Speed and key security are important benefits.

The tool also supports the new 128-bit AES (Advanced Encryption Standard) for symmetric encryption. It also allows for the decryption of GnuPG-encrypted data sets under OS/390: this enables Unix or Windows systems to create encrypted datasets destined for the mainframe. GnuPG (see www.gnupg.org) is a free PGP with the same functions as PGP, but no GUI interface.

For further information contact:
ASPG, 3185 Horseshoe Drive South, Naples, Florida 34104, USA.
Tel: (941) 649 1548
Fax: (941) 649 6391
http://www.aspg.com

IBM has announced IMS V8 with enhanced IMS Database Manager (IMS DB) and IMS Transaction Manager (IMS TM). Simplified access to new and existing IMS applications and data, along with Internet access, is supported with z/OS Version 1, Release 2, IMS Connect Version 1, Release 2, VisualAge Java Enterprise Edition Version 4, IBM Developer Kit for OS/390, Java2 Technology Edition, IMS DataPropagator Version 3, Release 1, CICS Transaction Server for z/OS Version 2, and DB2 UDB for z/OS and OS/390 Version 7.

Improved systems and data management and enhanced performance and availability are also supported with IMS High Performance Change Accumulation Version 1, Release 1, IMS Network Compression Facility for z/OS, IMS Fast Path Basic Tools for z/OS, Version 1, Release 2, IMS Extended Terminal Option for z/OS, Version 2, Release 2, and IMS Parallel Reorganization Version 2, Release 1.

IMS V8 Database Manager enhancements include availability/recovery items such as IMS/DB2 coordinated disaster recovery support, Database Recovery Control (DBRC) enhancements, parallel database processing, batch Resource Recovery Service (RRS) support, Database Image Copy 2 enhancements, and Fast Path Shared Virtual Storage Option (VSO) Coupling Facility enhancements.

Contact your local IBM representative for further information:
http://www.ibm.com/ims

***