1

Concocting 500 Virtual Linux Servers in Three Days

A need arose for 500 identical Linux systems on System z for the purpose of performance testing. An aggressive goal of three days was planned for this work. Creating these systems iteratively 500 times would have been prohibitively time consuming and repetitive, thus leading to errors. Therefore, automation of this process was natural.

This paper is a description of the automation process. The approach taken followed these overall steps:

All of the code used in this paper is listed in section 1.9, "Source code" on page 14.

Though the sections describe 250 Linux systems, two LPARs were utilized. 250 virtual Linux servers were created on the first LPAR, then the process was repeated on the second LPAR.

1.1 Preparing the z/VM systems

Sufficient resources, especially disk and memory were procured on each of two LPARs. The disk space was approximately 525GB:

z/VM System	21GB (3 3390-s)
For Spool space	14GB (2 3390-9s)
For Paging	140GB (20 3390-9s)
For Linux	350GB (8 3390s with 64,554 cylinders)

Memory:

Central	14GB
Expanded	2GB

z/VM 5.4 was installed then DirMaint and RACF were configured. DirMaint was set up with a disk pool named USERDATA and was configured in such a way that when a new user ID is created, it is given the proper RACF permissions.

Initially, the first LPAR had only 2GB of central storage (memory). After approaching 200 running Linux systems, the system's performance degraded significantly. The **INDICATE LOAD** command showed the following:

```
==> ind
AVGPROC-094% 04
XSTORE-001798/SEC MIGRATE-0473/SEC
MDC READS-000001/SEC WRITES-000001/SEC HIT RATIO-059%
PAGING-3794/SEC STEAL-005%
Q0-00016(00004)                          DORMANT-00162
Q1-00022(00002)           E1-00022(00014)
Q2-00041(00011) EXPAN-002 E2-00008(00004)
Q3-00128(00046) EXPAN-002 E3-00000(00000)

PROC 0000-094% CP       PROC 0001-094% CP
PROC 0002-094% CP       PROC 0003-094% CP

LIMITED-00000
```

The system was brought down and the LPAR was upgraded to 14GB/2GB (central/expanded) of memory. Then the LPAR behaved reasonably on a z990 with four CPUs.

1.2 Creating a common 191 disk

On the MAINT user ID a new user ID named LNXMAINT is created. It is given a 300 cylinder 192 disk. This disk will become the read-only 191 disk for each of the 250 user IDs. A small 20 cylinder 191 disk is also created for LNXMAINT's PROFILE EXEC.

Following is the initial LNXMAINT DIRECT file:

```
USER LNXMAINT XXXX 64M 128M BEG
   INCLUDE TCPCMSU
   LINK TCPMAINT 0592 0592 RR
```

Following are the **DIRM ADD** and **AMDISK** commands to create the user ID:

```
==> dirm add lnxmaint
DVHXMT1191I Your ADD request has been sent for processing.
...
DVHREQ2289I Your ADD request for LNXMAINT at * has completed; with RC
DVHREQ2289I = 0.
==> dirm for lnxmaint amd 191 xxxx autog 20 userdata mr
DVHXMT1191I Your AMDISK request has been sent for processing.
...
 DVHREQ2289I Your AMDISK request for LNXMAINT at * has completed; with RC
 DVHREQ2289I = 0.
==> dirm for lnxmaint amd 192 xxxx autog 300 userdata mr
 DVHREQ2288I Your AMDISK request for LNXMAINT at * has been accepted.
...
```

```
    DVHREQ2289I Your AMDISK request for LNXMAINT at * has completed; with RC
DVHREQ2289I = 0.
```

The new user ID is logged onto and the two new disks are formatted for CMS files:

```
==> format 191 a
DMSFOR603R FORMAT will erase all files on disk A(191). Do you wish to continue?
Enter 1 (YES) or 0 (NO).
1
DMSFOR605R Enter disk label:
lxm191
DMSFOR733I Formatting disk A

DMSFOR732I 20 cylinders formatted on A(191)
==> format 192 b
DMSFOR603R FORMAT will erase all files on disk B(192). Do you wish to continue?
Enter 1 (YES) or 0 (NO).
1
DMSFOR605R Enter disk label:
lxm192
DMSFOR733I Formatting disk B
DMSFOR732I 300 cylinders formatted on B(192)
```

A small PROFILE EXEC is created on the A disk:

```
/* PROFILE EXEC for LNXMAINT */
'CP SET RUN ON'
'CP SET PF11 RETRIEVE FORWARD'
'CP SET PF12 RETRIEVE'
'ACC 592 C'
```

The LNXMAINT 192 disk is initially populated with six files from another z/VM system. The SLES10S2 KERNEL and SLES10S2 INITRD files are binary data so they are moved last with FTP using subcommands of **TYPE E** and **MODE B** before copying. It is confirmed that these files are in Fixed 80-byte record format after they arrive.

From the LNXMAINT user ID, the six files are shown via the **FILELIST** command:

```
==> filel * * d
SLES10S2 EXEC     D1 V         68          9          1 10/29/08 10:00:43
SLES10S2 INITRD   D1 F         80     106677       2084 10/29/08  9:32:25
SLES10S2 KERNEL   D1 F         80      89515       1231 10/29/08  9:29:58
SWAPGEN  EXEC     D1 V         72        358          5 10/29/08  9:27:38
PROFILE  EXEC     D1 V         63         17          1 10/29/08  9:24:30
ZVL281   PARM-S10 D1 F         80          9          1 10/31/08 14:28:01
```

These six files are needed for the following reasons:

1. SLES10S2 EXEC - A small EXEC to start the SLES 10 SP2 install process:

```
/* EXEC to punch SLES-10 install system to reader and IPL from it */
'CP SPOOL PUN *'
'CP CLOSE RDR'
'PUR RDR ALL'
'PUN SLES10S2     KERNEL  * (NOH'
'PUN' USERID() 'PARM-S10  * (NOH'
'PUN SLES10S2     INITRD  * (NOH'
'CH RDR ALL KEEP'
'IPL 00C CLEAR'
```

2. SLES10S2 INITRD - The SLES 10 SP2 initial RAMdisk
3. SLES10S2 KERNEL - The SLES 10 SP2 kernel

4. SWAPGEN EXEC - An EXEC to create Linux swap spaces from VDISK. See: http://www.sinenomine.net/products/vm/swapgen

5. PROFILE EXEC - The common profile all Linux systems will run when CMS is IPLed:

```
/* PROFILE EXEC for Linux virtual servers */
'CP SET RUN ON'
'CP SET PF11 RETRIEVE FORWARD'
'CP SET PF12 RETRIEVE'
'ACC 592 C'
'SWAPGEN 300 524288'  /* create a 256M VDISK disk swap space */
'SWAPGEN 301 1048576' /* create a 512M VDISK disk swap space */
'PIPE CP QUERY' userid() '| var user'
parse value user with id . dsc .
if (dsc = 'DSC') then /* user is disconnected */
  'CP IPL 100'
else /* user is interactive -> prompt */
 do
  say 'Do you want to IPL Linux from minidisk 100? y/n'
  parse upper pull answer .
  if (answer = 'Y') then 'CP IPL 100'
 end /* else */
```

   The PROFILE EXEC common to all Linux systems creates two swap spaces using VDISK of sizes 256MB and 512MB at virtual addresses 300 and 301. It then determines if the logon is interactive or disconnected. If disconnected (true the majority of the time), Linux is automatically IPLed from minidisk 100.

6. ZVL282 PARM-S10 - A SLES 10 parameter file populated with the correct network information:

```
ramdisk_size=65536 root=/dev/ram1 ro init=/linuxrc TERM=dumb
HostIP=9.12.29.235  Hostname=zvl78001.pdl.pok.ibm.com
Gateway=9.12.29.1 Netmask=255.255.255.0
Broadcast=9.12.29.255 Layer2=0
ReadChannel=0.0.0600  WriteChannel=0.0.0601  DataChannel=0.0.0602
Nameserver=9.12.16.2 Portname=whatever
Install=nfs://9.12.33.17/install/suse/sles10/sp2/s390x
UseVNC=1  VNCPassword=123456
InstNetDev=osa OsaInterface=qdio OsaMedium=eth Manual=0
```

   The contents of this file are briefly described:

   - The first line is the default shipped with the SLES distribution.
   - The second line sets the IP address and host name.
   - The third and fourth and sixth lines set the other standard network data. Layer2=0 means Linux will be connected to a layer 3 VSWITCH, which is the default value when it is created on z/VM.
   - The fifth line sets the virtual network interface card (NIC) device addresses. This corresponds to a NICDEF 600 statement in the Linux user ID's directory entry.
   - The seventh line sets the location of the SLES 10 SP2 install server and to use the NFS protocol.
   - The eighth line specifies that VNC will be used for the graphical portion of the Linux installation.
   - The last line is standard for OSA, VSWITCH or Guest LAN connectivity.

With these six files created, LNXMAINT is logged off and MAINT is logged back on.

1.3 Creating a common PROFILE for the Linux user IDs

A user directory PROFILE named LNXDFLT is created. A PROFILE stores directory statements that will be common to many user IDs.

The LNXDFLT DIRECT file is created via **XEDIT**:

```
==> x lnxdflt direct
PROFILE LNXDFLT
  CPU 00 BASE
  CPU 01
  IPL CMS
  MACHINE ESA 4
  CONSOLE 0009 3215 T
  NICDEF 0600 TYPE QDIO LAN SYSTEM <vsw1>
  SPOOL 000C 2540 READER *
  SPOOL 000D 2540 PUNCH A
  SPOOL 000E 1403 A
  LINK MAINT 019O 019O RR
  LINK MAINT 019D 019D RR
  LINK MAINT 019E 019E RR
  LINK LNXMAINT 0192 0191 RR
  LINK TCPMAINT 0592 0592 RR
```

Note: **<vsw1>** is in angle brackets because the VSWITCH previously created on z/VM is named VSW1. If your VSWITCH has a different name, replace it in **<vsw1>**, but don't use the angle brackets.

The LNXDFLT PROFILE is now created with the **DIRM ADD** command:

```
==> dirm add lnxdflt
DVHXMT1191I Your ADD request has been sent for processing.
...
DVHREQ2289I Your ADD request for LNXDFLT at * has completed; with RC
DVHREQ2289I = 0.
```

This profile will be the common user directory statements to all Linux user IDs. Most statements are standard, but the NICDEF 0600 statement is important. This gives each Linux user ID a virtual network interface card at addresses 600-602. Recall that these virtual addresses are specified in the SLES 10 parameter file. It attaches the other end of the virtual network cable to the VSWITCH named VSW1. The existence of this virtual switch can be verified with the **QUERY VSWITCH** command:

```
==> q vswitch vsw1
VSWITCH SYSTEM vsw1      Type: VSWITCH Connected: 249  Maxconn: INFINITE
  PERSISTENT  RESTRICTED    NONROUTER                 Accounting: OFF
  VLAN Unaware
  MAC address: 02-2B-9A-00-00-03
  State: Ready
  IPTimeout: 5         QueueStorage: 8
  Portname: CHPID10    RDEV: 1A00.P00 Controller: DTCVSW2  VDEV:  1A00
```

The keyword **NONROUTER** in the output of **QUERY VSWITCH** is important. This implies the default type of VSWITCH which is layer 3. If the VSWITCH is layer 2, it will show the keyword **ETHERNET**.

1.4 Creating a golden Linux image

From MAINT a new user ID is created for the first Linux to be manually installed onto. It is given a 2016 cylinder minidisk at virtual address 100. The large volume 3390s have 64554 cylinders, so 32 Linux systems, each with one 2106 cylinder disk, can fit and will occupy all but 42 cylinders. Eight of these volumes will allow for up to 256 user IDs. A Linux system with only a single 2016 cylinder minidisk for a root file system is quite a small amount of disk space. However, because the Linux systems were only used for performance testing, the small disk size was acceptable.

In this example, the golden image is installed onto the user ID named ZVL281. Choosing a different naming convention will require changes to the sample code.

Following is the directory entry:

```
==> x zvl281 direct
USER ZVL281 xxxxxx 256M 1G   G
    INCLUDE LNXDFLT
```

The first Linux user ID is created with the **DIRM ADD** command. Then the **DIRM AMDisk** command adds a minidisk at virtual address 100, obtaining the disk space from the group named **userdata**.

```
==> dirm add zvl281
PUN FILE 0042 SENT TO   DIRMAINT RDR AS  0022 RECS 0009 CPY  001 O NOHOLD NOKEEP
DVHXMT1191I Your ADD request has been sent for processing.
Ready; T=0.03/0.03 14:10:48
 DVHREQ2288I Your ADD request for ZVL281 at * has been accepted.
 DVHBIU3450I The source for directory entry ZVL281 has been updated.
 DVHBIU3424I The next ONLINE will take place immediately.
 DVHDRC3451I The next ONLINE will take place via delta object directory.
 DVHBIU3428I Changes made to directory entry ZVL281 have been placed
 DVHBIU3428I online.
 DVHREQ2289I Your ADD request for ZVL281 at * has completed; with RC = 0.
==> dirm for zvl281   amd 100 xxxx autog 2016 userdata mr
 DVHXMT1191I Your AMDISK request has been sent for processing.
 DVHREQ2288I Your AMDISK request for ZVL281 at * has been accepted.
 DVHSCU3541I Work unit 31141137 has been built and queued for processing.
 DVHSHN3541I Processing work unit 31141137 as MAINT from ZLV8231,
 DVHSHN3541I notifying MAINT at ZLV8231, request 18 for ZVL281 sysaffin
 DVHSHN3541I *; to: AMDISK 0100 XXXX AUTOG 2016 USERDATA MR
 DVHBIU3450I The source for directory entry ZVL281 has been updated.
 DVHBIU3424I The next ONLINE will take place immediately.
 DVHDRC3451I The next ONLINE will take place via delta object directory.
 DVHBIU3428I Changes made to directory entry ZVL281 have been placed
 DVHBIU3428I online.
 DVHSHN3430I AMDISK operation for ZVL281 address 0100 has finished
 DVHSHN3430I (WUCF 31141137).
 DVHREQ2289I Your AMDISK request for ZVL281 at * has completed; with RC =
 DVHREQ2289I 0.
```

Output from **DIRM** commands are watched for **RC = 0** messages.

The new user ID is logged on to be sure there are no error messages, especially regarding disks and network. Access to the VSWITCH is handled automatically on this system with RACF. On systems without RACF, access for each user ID to the VSWITCH must be explicitly given. For example:

```
==> set vswitch vsw1 grant zvl281
Command complete
```

Because this access also has to be set at IPL time, the **SET VSWITCH** commands for all user IDs are often also added to AUTOLOG1's PROFILE EXEC.

1.4.1 Installing SLES 10 SP2 on the golden image

This section does not give step-by-step instructions on how to install Linux under z/VM. Rather, it gives high level steps:

- A parameter file is created on the LNXMAINT 192 disk (see 1.2, "Creating a common 191 disk" on page 2)
- Memory (storage) is reset to 512 MB (**DEF STOR 512M**)
- The install process is started

For a detailed description of installing and customizing SLES 10 SP2, see the IBM Redbook *z/VM and Linux on IBM System z The Virtualization Cookbook for SLES 10 SP2*, on the Web at:

http://www.redbooks.ibm.com/abstracts/sg247493.html?Open

The Redbook describes many different file systems that utilize logical volumes. In this example only a single, much smaller, root file system is created.

The golden image SLES 10 SP2 is installed as follows:

- One root file system, of type ext3, is created from minidisk 100. The 2016 cylinders result in about 1.4GB of ext3-formatted disk space.
- On the window "*Create a Primary Partition on /dev/dasda*", the **Fstab Options** button is clicked. In the *Fstab options* window, select the **Device path** radio button in the *Mount in /etc/fstab by* radio group. This prevents the disk ID for minidisk 100 from being used in /etc/fstab which makes cloning very difficult.
- The swap spaces at virtual addresses 300 and 301 are utilized.
- The software chosen is only the **Base** and **32-bit compatibility** groups. This will result in Linux systems that are slightly under 1GB.

1.4.2 Customizing the SLES 10 SP2 golden image

After the system is installed, some small customization steps are performed.

- The default IPL menu timeout is decreased from 10 to 3 seconds in /etc/zipl.conf.
- The vmpoff=logoff parameter is added to /etc/zipl.conf.
- The cmsfs RPM is added with YaST so CMS files can be read from each user ID's 191 disk.

1.4.3 Modifying the clones

To clone the golden image, networking values (especially the IP address and DNS host name) on the cloned images will need to be modified. A Linux bash script named **boot.findself** is written for this purpose. It is stored in the startup directory /etc/init.d and set to start in the boot run level. After the golden image is copied to another User ID's minidisks, the clone is IPLed, and the boot.findself script runs before the network starts. The values are modified and the clone comes up with the correct IP address.

Obtaining the code

The **boot.findself** script can be obtained from the following Web site:

http://www.vm.ibm.com/devpages/mikemac/

In addition to this script, all the REXX EXECs and XEDIT macros used in this paper are also available. The code for the script, REXX EXECs and XEDIT macros are listed in the following sections:

- 1.9.1, “The boot.findself Linux script” on page 14
- 1.9.2, “The MK1LNXID EXEC” on page 18
- 1.9.3, “The CHUSERID XEDIT macro” on page 18
- 1.9.4, “The MKLNXIDS EXEC” on page 19
- 1.9.5, “The MKPARMS EXEC” on page 19
- 1.9.6, “The CHANGEIP XEDIT macro” on page 20
- 1.9.7, “The COPYMDSK EXEC” on page 20
- 1.9.8, “The CLON1LNX EXEC” on page 21
- 1.9.9, “The CLONLNXS EXEC” on page 21

Enabling the boot.findself script

The **boot.findself** script is copied to the /etc/init.d/ directory (with **wget** in this example, but there are other methods will work).

```
# cd /etc/init.d
# wget http://www.vm.ibm.com/devpages/mikemac/boot.findself
```

The script is set to be executable with the **chmod** command:

```
# chmod +x boot.findself
```

The script is set to run at boot time with the **chkconfig** command:

```
# chkconfig boot.findself on
```

The **chkconfig --list** command verifies that it is set on in the boot run level:

```
# chkconfig --list boot.findself
boot.findself            0:off  1:off  2:off  3:off  4:off  5:off  6:off  B:on
```

1.4.4 Verifying the golden image

An SSH session as root is started with the newly installed and configured system. Some settings are queried and configuration changes are verified.

The /etc/fstab file is observed to verify the root file system is referenced by path:

```
# cat /etc/fstab
/dev/disk/by-path/ccw-0.0.0100-part1 /    ext3       acl,user_xattr        1 1
/dev/dasdb1          swap                 swap       defaults              0 0
/dev/dasdc1          swap                 swap       defaults              0 0
...
```

The **df -h** command shows that the root file system occupies about 910 MB and is 69% full:

```
# df -h
Filesystem            Size  Used Avail Use% Mounted on
/dev/dasda1           1.4G  910M  415M  69% /
udev                  121M   56K  121M   1% /dev
```

The contents of the /proc/sys/kernel/hz_timer file verify that the “timer pop” is off, which is now the default value with SLES 10 SP2. This is important for performance with so many Linux systems running:

```
# cat /proc/sys/kernel/hz_timer
0
```

It is verified that the CMSFS package is installed:

```
# rpm -qa | grep cmsfs
cmsfs-1.1.8-3.2
```

The system is now shutdown:

```
# shutdown -h now

Broadcast message from root (pts/0) (Sun Nov  2 08:44:55 2008):

The system is going down for system halt NOW!
```

The golden image is now ready to be cloned.

1.5 Creating 250 z/VM user IDs

Now 250 z/VM user IDs are needed. Think of this as creating 250 virtual PCs with memory, disk and network resources.

A REXX EXEC named **MK1LNXID EXEC** and an XEDIT macro named **CHUSERID XEDIT** are written to copy the ZVL281 DIRECT file and modify the file name and the user ID. For the code listing and details, see 1.9.2, “The MK1LNXID EXEC” on page 18 and 1.9.3, “The CHUSERID XEDIT macro” on page 18.

All EXECs and XEDIT macros are stored on the MAINT 191 disk.

A single user ID named ZVL282 is created with the **MK1LNXID EXEC**:

```
==> mk1lnxid zvl282
Creating the user ID zvl282
DMSXCG517I 1 occurrence(s) changed on 1 line(s)
DVHELD1190I Command EXECLOAD complete; RC= 0.
PUN FILE 0569 SENT TO   DIRMAINT RDR AS  0582 RECS 0009 CPY  001 0 NOHOLD NOKEEP
*SMSG MAINT DVHXMT1191I REQUEST=592 RTN=DVHXMT MSG=1191 FMT=01 SUBS= ADD
*SMSG DIRMAINT DVHREQ2288I REQUEST=592 RTN=DVHREQ MSG=2288 FMT=01 SUBS= ADD ZVL2
8252 * MAINT
*SMSG DIRMAINT DVHBIU3450I REQUEST=592 RTN=DVHBIU MSG=3450 FMT=01 SUBS= zvl282
*SMSG DIRMAINT DVHBIU3424I REQUEST=592 RTN=DVHBIU MSG=3424 FMT=01 SUBS= zvl282
*SMSG DIRMAINT DVHDRC3451I REQUEST=592 RTN=DVHDRC MSG=3451 FMT=01 SUBS=
*SMSG DIRMAINT DVHBIU3428I REQUEST=592 RTN=DVHBIU MSG=3428 FMT=01 SUBS= zvl282
*SMSG DIRMAINT DVHREQ2289I REQUEST=592 RTN=DVHREQ MSG=2289 FMT=01 SUBS= 0 ADD ZV
L28252 * MAINT ZLV8231
*SMSG MAINT DVHXMT1191I REQUEST=593 RTN=DVHXMT MSG=1191 FMT=01 SUBS= AMDISK
*SMSG DIRMAINT DVHREQ2288I REQUEST=593 RTN=DVHREQ MSG=2288 FMT=01 SUBS= AMDISK Z
VL28252 * MAINT
*SMSG DIRMAINT DVHSCU3541I REQUEST=593 RTN=DVHSCU MSG=3541 FMT=03 SUBS= 12115001
*SMSG DIRMAINT DVHSHN3541I REQUEST=593 RTN=DVHSHN MSG=3541 FMT=04 SUBS= 12115001
 MAINT ZLV8231 MAINT ZLV8231 593 zvl282 * AMDISK 0100 XXXX AUTOG 2016 USERDATA
 MR
*SMSG DIRMAINT DVHBIU3450I REQUEST=593 RTN=DVHBIU MSG=3450 FMT=01 SUBS= zvl282
*SMSG DIRMAINT DVHBIU3424I REQUEST=593 RTN=DVHBIU MSG=3424 FMT=01 SUBS= zvl282
```

```
*SMSG DIRMAINT DVHDRC3451I REQUEST=593 RTN=DVHDRC MSG=3451 FMT=01 SUBS=
*SMSG DIRMAINT DVHBIU3428I REQUEST=593 RTN=DVHBIU MSG=3428 FMT=01 SUBS= zvl282
*SMSG DIRMAINT DVHSHN3430I REQUEST=593 RTN=DVHSHN MSG=3430 FMT=01 SUBS= AMDISK Z
VL28252 0100 12115001
*SMSG DIRMAINT DVHREQ2289I REQUEST=593 RTN=DVHREQ MSG=2289 FMT=01 SUBS= 0 AMDISK
 zvl282 * MAINT ZLV8231
DVHELD1190I Command EXECDROP complete; RC= 0.
```

Watch for return codes of 0 from the DirMaint commands. Logon to the user ID to verify it has been created successfully.

1.5.1 Creating the remaining user IDs

Now that one user ID has been created with **MK1LNXID**, another EXEC named **MKLNXIDS** is written to create the remaining 249 user IDs. See 1.9.4, “The MKLNXIDS EXEC” on page 19 for a listing of the source code and a more detailed description.

This EXEC ran for about 25 minutes:

```
==> mklnxids
...
```

When completed there are 250 user IDs, all with a 2016 cylinder 100 disk, 256 MB of memory and a virtual NIC starting at address 600.

1.6 Creating 250 parameter files

Now that there are 250 new user IDs, each one needs a corresponding parameter file on the `LNXMAINT 192` disk. These files will contain networking and other information. The IP address and the DNS host name will have to be set correctly for each virtual server.

A REXX EXEC named **MKPARMS** and an XEDIT macro named **CHANGEIP** are written. See section 1.9.5, “The MKPARMS EXEC” on page 19 and 1.9.6, “The CHANGEIP XEDIT macro” on page 20 for a complete listing of the code and a more detailed description.

The new **MKPARMS EXEC** is run:

```
==> mkparms
DASD 1192 LINKED R/W; R/O BY   248 USERS
DMSXCG517I 1 occurrence(s) changed on 1 line(s)
DMSXCG517I 1 occurrence(s) changed on 1 line(s)
...
```

After clearing the screen a number of times, the EXEC creates the parameter files in a matter of seconds.

To review, the following should be true:

- One Linux system has been installed manually - the *golden image*.
- 250 new user IDs have been created
- Corresponding parameter files for each of the user IDs have been created
- The golden image is primed with a script to personalize itself.

The next step is to clone Linux systems.

1.7 Cloning 250 Linux virtual servers

The golden image can now be copied 250 times with either **DDR** or **FLASHCOPY**. First, a REXX EXEC to copy a single minidisk, **COPYMDSK**, is written. For a listing of the source code, see section 1.9.7, “The COPYMDSK EXEC” on page 20.

This EXEC takes two arguments: the virtual addresses of the source and target minidisks. It first tries to copy the disk with **FLASHCOPY**. If this fails for any reason, it falls back to **DDR**.

1.7.1 Cloning one Linux

Now a REXX EXEC to clone one Linux, **CLON1LNX**, is written that will utilize the **COPYMDSK EXEC**. For a listing of the source code, see section 1.9.8, “The CLON1LNX EXEC” on page 21.

One Linux is cloned with the **CLON1LNX EXEC** to the user ID ZVL282:

```
==> clon1lnx zvl282
HCPCQU045E ZVL281 not logged on
HCPCQU045E ZVL282 not logged on

Copying mindisk 100 from zvl281 to zvl282

Copying minidisk 1100 to 2100 ...
HCPNFC332E Invalid control unit type - 1100
FLASHCOPY failed, falling back to DDR ...
z/VM DASD DUMP/RESTORE PROGRAM
HCPDDR696I VOLID READ IS 0X0100
HCPDDR697I NO VOL1 LABEL FOUND
COPYING   0X0100
COPYING DATA  11/02/08 AT 14.31.46  GMT FROM 0X0100
INPUT CYLINDER EXTENTS      OUTPUT CYLINDER EXTENTS
      START       STOP         START        STOP
          0       2015             0        2015
END OF COPY
END OF JOB
Return value = 0
DASD 1100 DETACHED
DASD 2100 DETACHED
```

In this example, **FLASHCOPY** command in the **COPYMDSK** EXEC fails, but **DDR** succeeds.

1.7.2 Logging onto the new Linux interactively

Logon to the newly cloned user ID. If RACF is running, you may be asked to reset the password. Press **Enter** at the VM Read prompt. The **PROFILE EXEC** on the LNXMAINT 192 disk is invoked and two VDISK swap spaces at virtual addresses 300 and 301 are created.

```
LOGON ZVL282
RPIMGR042I PASSWORD EXPIRED

To change your password - enter: nnn/nnn  where nnn = new password
or,
enter LOGOFF to cancel

 ICH70001I ZVL282 LAST ACCESS AT **:**:** ON ****, **** **,****
 HCPRPW004I Password changed
00: NIC 0600 is created; devices 0600-0602 defined
00: z/VM Version 5 Release 4.0, Service Level 0801 (64-bit),
```

```
00: built on IBM Virtualization Technology
00: There is no logmsg data
00: FILES: 0001 RDR,   NO PRT,   NO PUN
00: LOGON AT 11:08:09 EDT SUNDAY 11/02/08
z/VM V5.4.0    2008-10-24 11:04

DMSACP723I A (191) R/O
DMSACP723I C (592) R/O
DIAG swap disk defined at virtual address 300 (64989 4K pages of swap space)
DIAG swap disk defined at virtual address 301 (129981 4K pages of swap space)
Do you want to IPL Linux from minidisk 100? y/n
y
```

The letter **y** is typed at the prompt to IPL from minidisk 100. Linux should now boot. If it does not then confirm that the disk has been copied.

```
00: zIPL v1.6.3 interactive boot menu
00:
00:  0. default (ipl)
00:
00:  1. ipl
00:  2. Failsafe
00:
00: Note: VM users please use '#cp vi vmsg <number> <kernel-parameters>'
00:
00: Please choose (default will boot in 3 seconds):
00: Booting default (ipl)...
Linux version 2.6.16.60-0.21-default (geeko@buildhost) (gcc version 4.1.2 200701
15 (SUSE Linux)) #1 SMP Tue May 6 12:41:02 UTC 2008
We are running under VM (64 bit mode)
Detected 2 CPU's
Boot cpu address  0
Built 1 zonelists
Kernel command line: root=/dev/disk/by-path/ccw-0.0.0100-part1 vmpoff=LOGOFF TER
M=dumb BOOT_IMAGE=0
...
```

Three changes to a default SLES 10 SP2 system are initially observed. The first and the third are optional, the second is critical:

1. A reduced boot wait time of 3 seconds
2. The root file system being reference by path (**/dev/disk/by-path/ccw-0.0.0100-part1**)

   Note: if the disk is referenced by disk ID, it will fail here. The quickest way to rectify this is to reinstall the golden image and remember to click the **Fstab options** button.
3. The `vmpoff=LOGOFF` parameter is in `/etc/zipl.conf`.

The boot messages are observed carefully to confirm that the **`boot.findself`** script is run

```
...
Can't determine current runlevel
..done
Executing boot.findself
Modifying (escaped) 9\.12\.29\.235 to 172.16.231.2 and other IP info
Activating remaining swap-devices in /etc/fstab...
...
```

If the **`boot.findself`** script does not run, confirm that it exists in `/etc/init.d`, is set to be executable, and has been turned on by **`chkconfig`**. A number of errors were initially encountered in this script. To aid in debugging, a **`-x`** suffix is appended to the first line of the script on the golden image. This turns on the shell trace function.

```
# head -1 /etc/init.d/boot.findself
#!/bin/bash -x
```

The golden image is shut down, another Linux cloned and started interactively. With tracing on, much more information about what is happening in the code can be observed as the new clone boots. The **-x** argument is removed after the script is debugged.

1.7.3 Cloning the remaining Linux servers

Now that an EXEC exists to clone one Linux, it is fairly easy to repeat the process. A REXX EXEC, **CLONLNXS**, is written to loop 249 more times, from 3 to 251, to clone the virtual servers. For a listing of the source code, see section 1.9.9, “The CLONLNXS EXEC” on page 21.

The EXEC is invoked:

```
==> clonlnxs
...
```

It is fairly simple and virtually no error detection or correction code exists. It is very long running. The test system had no **FLASHCOPY** feature, but did have some fairly fast disks. So the clones were initially coming about two minutes apart but this interval continued to increase up to four minutes as the z/VM system became more pressured. This resulted in the total running time being more than 12 hours.

The EXEC is run on MAINT. To avoid the need to clear the screen many times over the course of many hours, the user ID is disconnected via the **#CP DISC** command. The creation of Linux systems is observed by simply **ping**ing the new virtual servers as they arrive on the network.

1.8 Creating 250 more Linux virtual servers

The title of this paper states 500 Linux systems, but the creation of only 250 is discussed. This is because two sets of 250 systems were created on two different z/VM LPARs. The second z/VM system is similarly configured.

A golden images exists on the first LPAR. The entire process could be repeated on the second LPAR to include installing Linux manually. However, that would require configuring two golden images identically which would be more likely to lead to error. So a method of copying the golden image from one LPAR to the second is chosen.

If the DASD on one system can also be accessed from the second, then this is a possibility and **DDR** could again be used. Because the two LPARs in this example do not have access to each other’s disks, the **PIPEDDR** command is used. It is not standard with z/VM, but is available on the Web at:

http://www.vm.ibm.com/download/packages/descript.cgi?PIPEDDR

First a user ID on the second LPAR is created. In this example it is ZVL231. The user ID is logged onto. Start **PIPEDDR** on the target side with the **restore** and **listen** parameters so the target user ID is listening. Note the port number that gets used (**1033** in this example):

```
==> pipeddr restore * 100 (listen
Any data on ZVL291 0100 will be erased.
Continue? (Y/N)
y
Connecting to TCP/IP.
Waiting for connection on port 1033
... (prompt will stop here until data is sent)
```

```
Sending user is ZVL281 at ZVL2831
Receiving data from zlv2831.pdl.pok.ibm.com
815 MB received.
Data restored successfully.
```

Now go to the source z/VM system and logon to the user ID with the golden image, ZVL281 in this example. Invoke the **PIPEDDR** command with the **dump** parameter and the TCP/IP address of the source z/VM (**9.12.28.231**) as well as the port number that the target system is listening on (**1033**):

```
==> pipeddr dump * 100 9.12.28.231 1033
Dumping disk ZVL281 0100 to 9.12.31.78
...
-- All data sent to ZVL281 AT ZLV8231 --
815 MB transmitted.
```

The golden image is now copied to the second LPAR.

1.8.1 Completing the creation of 250 more Linux virtual servers

The following high level steps were taken to create another set of 250 Linux systems on the second LPAR:

- The REXX EXECs and the XEDIT macros were copied to the second system (MAINT 191).
- The new user IDs were created with **MKUSERS**.
- The new parameter files were created with **MKPARMS** using the TCP/IP information for the second set of Linux systems (172.16.31/24 in this example).
- One Linux was cloned with **CLON1LNX**. It is booted interactively, while watching for errors.
- After that was successful, 249 more systems were cloned with **CLONLNXS**. Again, this step required more than 12 hours.

1.9 Source code

All of the source code used to create the 500 virtual servers is in this section. The sections are:

1.9.1 The boot.findself Linux script

Following is the **boot.findself** script that runs once at the startup of a new Linux system to set its network values:

```
# cat /etc/init.d/boot.findself
#!/bin/sh
#
```

```
# /etc/init.d/boot.findself
#
### BEGIN INIT INFO
# Provides:          boot.findself
# Required-Start:    boot.localfs
# Required-Start:
# Required-Stop:
# Default-Start:     B
# Default-Stop:
# Description:       upon first boot find/modify IP@ + hostname, gen SSH keys
### END INIT INFO
# --------------------------------------------------------------------------
# THE PROGRAM IS PROVIDED ON AN "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS
# OF ANY KIND, EITHER EXPRESS OR IMPLIED INCLUDING, WITHOUT LIMITATION, ANY
# WARRANTIES OR CONDITIONS OF TITLE, NON-INFRINGEMENT, MERCHANTABILITY
# OR FITNESS FOR A PARTICULAR PURPOSE.
# NEITHER RECIPIENT NOR ANY CONTRIBUTORS SHALL HAVE ANY LIABILITY FOR ANY
# DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
# (INCLUDING WITHOUT LIMITATION LOST PROFITS), HOWEVER CAUSED AND ON ANY THEORY
# OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING
# NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OR
# DISTRIBUTION OF THE PROGRAM OR THE EXERCISE OF ANY RIGHTS GRANTED
# HEREUNDER, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGES
#
#+-------------------------------------------------------------------------+
function findID()
# Get my VM user ID - don't do anything on the source ID.
#+-------------------------------------------------------------------------+
 {
  targetID=$(cat /proc/sysinfo | grep "VM00 Name" | awk '{print $3}')
  if [ "$targetID" = "$sourceID" ]; then # don't do anything
    exit
  fi
 }

#+-------------------------------------------------------------------------+
function enableAdisk()
# Enable my 191 (A) disk
#+-------------------------------------------------------------------------+
 {
  chccwdev -e 191 > /dev/null 2>&1
  rc=$?
  if [ $rc != 0 ]; then # unable to enable 191 disk
    echo "$0: Unable to enable 191, rc from chccwdev = $rc"
    exit 1
  fi
  Adisk=/dev/$(egrep '^0.0.0191' /proc/dasd/devices | awk '{print $7}')
 }

#+-------------------------------------------------------------------------+
function findSourceIP()
# Get the source IP address and hostName
#+-------------------------------------------------------------------------+
 {
  sourceParm="$sourceID.$parmType"
  cmsfslst -d $Adisk | grep $sourceID | grep $parmType > /dev/null
  rc=$?
  if [ $rc != 0 ]; then
    echo "$0: $sourceParm not found on 191 minidisk. Exiting"
    exit 2
```

```
  fi
  export local $(cmsfscat -a -d $Adisk $sourceParm)
  # ugly code follows to escape any dots (.) in the source values
  # dots are not interpreted literally in the sed regular expressions later
  sourceName=$(echo "$Hostname" | sed -e 's:\.:\\\.:g')
  sourceIP=$(echo "$HostIP" | sed -e 's:\.:\\\.:g')
  sourceHost=${Hostname%%.*}  # Chop domain name off to leave host name
  sourceGW=$(echo "$Gateway" | sed -e 's:\.:\\\.:g')
  sourceMask=$(echo "$Netmask" | sed -e 's:\.:\\\.:g')
  sourceBroadcast=$(echo "$Broadcast" | sed -e 's:\.:\\\.:g')
  sourceReaddev=$ReadChannel
  sourceDNS=$Nameserver
 }

#+-------------------------------------------------------------------------+
function findTargetIP()
# Get my new IP address and hostname
#+-------------------------------------------------------------------------+
 {
  targetParm="$targetID.$parmType"
  cmsfslst -d $Adisk | grep $targetID | grep $parmType > /dev/null
  rc=$?
  if [ $rc != 0 ]; then
    echo "$0: $targetParm not found on 191 minidisk. Exiting"
    exit 3
  fi
  export local $(cmsfscat -a -d $Adisk $targetParm)
  targetName=$Hostname
  targetIP=$HostIP
  targetHost=${Hostname%%.*}  # Chop domain name off to leave host name
  targetGW=$Gateway
  targetMask=$Netmask
  targetBroadcast=$Broadcast
  targetReaddev=$ReadChannel
  targetDNS=$Nameserver
 }

#+-------------------------------------------------------------------------+
function modifyIP()
# Modify IP and DNS info in the following files:
#   /etc/HOSTNAME
#   /etc/hosts
#   /etc/sysconfig/network/routes
#   /etc/resolv.conf
#   /etc/sysconfig/network/ifcfg-qeth-bus-ccw-$ReadChannel
#+-------------------------------------------------------------------------+
 {
  echo "Modifying (escaped) $sourceIP to $targetIP and other IP info"
  ethOfile="/etc/sysconfig/network/ifcfg-qeth-bus-ccw-$ReadChannel"
  sed --in-place -e "s/$sourceName/$targetName/g" /etc/HOSTNAME
  sed --in-place -e "s/$sourceHost/$targetHost/g" \
                 -e "s/$sourceIP/$targetIP/g" /etc/hosts
  sed --in-place -e "s/$sourceIP/$targetIP/g" $ethOfile
  sed --in-place -e "s/$sourceGW/$targetGW/g" /etc/sysconfig/network/routes
  sed --in-place -e "s/$sourceIP/$targetIP/g" \
                 -e "s/$sourceMask/$targetMask/g" \
                 -e "s/$sourceBroadcast/$targetBroadcast/g" \
                    /etc/sysconfig/network/ifcfg-qeth-bus-ccw-$targetReaddev
  sed --in-place -e "s/$sourceDNS/$targetDNS/g" /etc/resolv.conf
  hostname $targetHost
```

```
 }

#+-----------------------------------------------------------------------------+
function genSSHkeys()
# Regenerate a set of SSH keys
#+-----------------------------------------------------------------------------+
 {
  rm /etc/ssh/ssh_host_*
  ssh-keygen -t rsa -N "" -q -f /etc/ssh/ssh_host_rsa_key
  ssh-keygen -t dsa -N "" -q -f /etc/ssh/ssh_host_dsa_key
  ssh-keygen -t rsa1 -N "" -q -f /etc/ssh/ssh_host_key
 }

# main()
# global variables
sourceID="ZVL281"       # VM user ID where first Linux was installed
parmType="PARM-S1O"     # File type of parameter file on 191 disk

# function calls
echo "Executing boot.findself"
findID
enableAdisk
findSourceIP
findTargetIP
modifyIP
genSSHkeys
chkconfig boot.findself off  # run only once => turn self off
```

Note the first line of the main code: `sourceID="ZVL281"`. If you chose a different user ID for the golden image, this variable must be set to reflect that change.

The script calls the following functions:

findID	Determines the user ID that it is running on, If the target system equals the source system (`ZVL281`), then that means the golden image is being booted. The script just stops so nothing is changed.
enableAdisk	Use the CMS file system to enable the 191 disk which is the `LNXMAINT 192` disk. This will allow access to both the source and the target parameter files.
findSourceIP	Variables in the source parameter file are exported and copied to new variables of the form `sourceXxxx`.
findTargetIP	Variables in the target parameter file are exported and copied to new variables of the form `targetXxxx`.
modifyIP	TCP/IP information is modified in important files such as `/etc/hosts`, `/etc/HOSTNAME`, `/etc/sysconfig/network/routes`, `/etc/resolv.conf` and the file corresponding to the primary adapter, `/etc/sysconfig/network/ifcfg-qeth-bus-ccw-0.0.0600`, in this example. Note that this may not be all the files where TCP/IP information exists, but it is sufficient to modify the Linux system to communicate with.
genSSHkeys	Regenerate the SSH keys so they are not duplicates of the golden image.

The last line of the script turns itself off with the **chkconfig** command so it will not run on subsequent reboots of the modified system.

1.9.2 The MK1LNXID EXEC

Following is the **MK1LNXID EXEC** that creates a single user ID with DirMaint:

```
==> type mk1lnxid exec
/*--------------------------------------------------------------------
THE PROGRAM IS PROVIDED ON AN "AS IS" BASIS, WITHOUT WARRANTIES OR
CONDITIONS OF ANY KIND, EITHER EXPRESS OR IMPLIED INCLUDING, WITHOUT
LIMITATION, ANY WARRANTIES OR CONDITIONS OF TITLE, NON-INFRINGEMENT,
MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.
NEITHER RECIPIENT NOR ANY CONTRIBUTORS SHALL HAVE ANY LIABILITY FOR
ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
DAMAGES (INCLUDING WITHOUT LIMITATION LOST PROFITS), HOWEVER CAUSED
AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY,
OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF
THE USE OR DISTRIBUTION OF THE PROGRAM OR THE EXERCISE OF ANY RIGHTS
GRANTED HEREUNDER, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGES
--------------------------------------------------------------------*/
/* MK1LNXID EXEC: create one user ID for a Linux system */
/* Arg 1: the user ID to create                         */
parse arg userid .
sourceFile = 'zvl281 direct a'
targetFile = userid 'direct a'
say 'Creating the user ID' userid
'copy' sourceFile targetFile '(rep'
'xedit' targetfile '(profile chuserid)' userid
'EXEC DIRMAINT EXECLOAD'
'EXEC DVHSAPI ADD' userid
retVal = rc
'PIPE STEM DVHSAPI. | CONSOLE'
if (retVal \= 0) then do
  say 'return code from DIRM ADD =' retVal
  exit 1
end
'EXEC DVHSAPI FOR' userid 'amdisk 100 xxxx autog 2016 userdata mr'
retVal = rc
'PIPE STEM DVHSAPI. | CONSOLE'
if (retVal \= 0) then do
  say 'return code from DIRM AMDISK =' retVal
  exit 1
end
'EXEC DIRMAINT EXECDROP'
'RAC PERMIT lnxmaint.192 CLASS(VMMDISK) ID('||userid||') ACCESS(READ)'
exit
```

The source directory entry file is copied to a new file with the file name being that of the target user ID. In the new file, the user ID is modified by calling the **CHUSERID XEDIT** macro which modifies and saves the new file. The remaining lines do the following:

- **DVHSAPI ADD** creates each new user ID
- **DVHSAPI AMDISK** add a 2016 cylinder minidisk at virtual address 100 from the group named **USERDATA**
- **RAC PERMIT** allows the new user ID to link to the LNXMAINT 192 disk. If RACF is not running, this call is not necessary.

1.9.3 The CHUSERID XEDIT macro

Following is the **CHUSERID XEDIT** macro:

```
==> type chuserid xedit
/* CHUSERID XEDIT: change the user ID in a directory entry */
parse upper arg fn ft fm '(' options ')' userid .
'command c/USER ZVL281/USER' userid||'/ *'
'command file'
```

This macro obtains the target user ID with the **PARSE** command. The user ID is modified via the **CHANGE** (**c**) XEDIT subcommand. The modified file is saved with the **FILE** XEDIT subcommand.

1.9.4 The MKLNXIDS EXEC

Following is the **MKLNXIDS** EXEC:

```
/*-------------------------------------------------------------------
THE PROGRAM IS PROVIDED ON AN "AS IS" BASIS, WITHOUT WARRANTIES OR
...
-------------------------------------------------------------------*/
/* MKLNXIDS EXEC: create 249 ZVL28xxx user IDs */
do i = 3 to 251
  userid = 'zvl28'||i
  'mk1lnxid' userid
end
exit
```

This EXEC is a small **DO** loop that creates user IDs by calling **MK1LNXID**. The user ID of each new Linux is set with a suffix that is the last octet of the IP address that it will use.

1.9.5 The MKPARMS EXEC

Following is the code for the **MKPARMS EXEC**. It copies from the first parameter file, named ZVL281 PARM-S10, in this example, and creates 249 more files incorporating the last octet of the IP dotted decimal address into the file name suffix.

```
==> type mkparms exec
/*-------------------------------------------------------------------
THE PROGRAM IS PROVIDED ON AN "AS IS" BASIS, WITHOUT WARRANTIES OR
...
-------------------------------------------------------------------*/
/* MKPARMS EXEC: copy PARM-S10 files and modify IP info */
'cp link lnxmaint 192 1192 mr'
if (rc \= 0) then do
  say 'LINK failed'
  exit 1
end
'access 1192 f'
if (rc \= 0) then do
  say 'ACCESS failed'
  exit 2
end

sourceFile = 'zvl281 parm-s10 f'
do i = 2 to 251
  targetFile = 'zvl28'||i 'parm-s10 f'
  'copy' sourceFile targetFile '(rep'
  'xedit' targetfile '(profile changeip)' i
end
exit
```

The LNXMAINT 192 disk is linked read/write (MR) and accessed as file mode F. The first parameter file, zvl281 parm-s10 f, is set as the source file. In the **DO** loop, the target file name is set based on the last octet of the IP address. The file is copied and then modified with the **CHANGEIP XEDIT** macro.

1.9.6 The CHANGEIP XEDIT macro

Following is the **CHANGEIP XEDIT** macro:

```
==> type changeip xedit
/* CHANGEIP XEDIT: change IP address and DNS name in a SLES 10 parm file */
parse upper arg fn ft fm '(' options ')' octet .
'command set stay on'
'command c/HostIP=172.16.231.2/HostIP=172.16.231.'||octet||'/ *'
'command c/zvl281/zvl28'||octet||'/ *'
'command file'
```

The parameter octet is pulled with the **PARSE** command. The XEDIT **SET STAY ON** subcommand is needed to keep the current line at the top of the file so each **CHANGE** subcommand operates on the entire file. Then the IP address and host name are modified with the XEDIT **CHANGE** subcommand. Finally the modified file is saved with the XEDIT **FILE** subcommand.

1.9.7 The COPYMDSK EXEC

This EXEC copies one minidisk to another. It first tries to use **FLASHCOPY**. If that fails, it falls back to using **DDR**.

```
==> type copymdsk exec
/*---------------------------------------------------------------------
THE PROGRAM IS PROVIDED ON AN "AS IS" BASIS, WITHOUT WARRANTIES OR
...
---------------------------------------------------------------------*/
/* COPYMDSK EXEC - copy minidisk w/FLASHCOPY, if it fails, try DDR   */
/*  Arg 1: vaddr of source minidisk                                  */
/*  Arg 2: vaddr of target minidisk                                  */
Address 'COMMAND'
Parse Arg source target .
Say
Say 'Copying minidisk' source 'to' target '...'
'CP FLASHCOPY' source '0 END' target '0 END'
If (rc \= 0) Then Do                  /* Fallback to DDR             */
  Say 'FLASHCOPY failed, falling back to DDR ...'
                                      /* Queue up DDR commands       */
  Queue 'SYSPRINT CONS'               /* Don't print to file         */
  Queue 'PROMPTS OFF'                 /* Don't ask 'Are you sure?'   */
  Queue 'IN' source '3390'            /* Input minidisk              */
  Queue 'OUT' target '3390'           /* Output minidisk             */
  Queue 'COPY ALL'                    /* Copy all contents           */
  Queue ' '                           /* Empty record ends DDR       */
  'DDR'
  retVal = rc
  End
Else retVal = rc
Say 'Return value =' retVal
Return retVal
```

1.9.8 The CLON1LNX EXEC

This EXEC <does this>

```
==> type clon1lnx exec
/*--------------------------------------------------------------------
THE PROGRAM IS PROVIDED ON AN "AS IS" BASIS, WITHOUT WARRANTIES OR
...
--------------------------------------------------------------------*/
/* CLON1LNX EXEC: clone 100 disk from zvl281 to a target user ID */
/* Arg 1: the target user ID                                     */
parse arg targetID .
sourceID = "zvl281"
call checkID sourceID
call checkID targetID
say ' '
say 'Copying mindisk 100 from' sourceID 'to' targetID
'CP LINK' sourceID '100 1100 RR'
'CP LINK' targetID '100 2100 MR'
COPYMDSK 1100 2100
'CP DETACH 1100'
'CP DETACH 2100'
exit

/*+-------------------------------------------------------------------+*/
checkID: procedure
/*| Check that a user ID is logged off                                |*/
/*|  parm 1: userID in question                                       |*/
/*+-------------------------------------------------------------------+*/
parse arg userid
'cp query' userid
retVal = rc
select
  when (retVal = 0) then do
    say 'User ID' userid 'must be logged off'
    exit 1
  end
  when (retVal = 3) then do
    say 'User ID' userid 'does not exist'
    exit 2
  end
  when (retVal = 45) then do
    /* no-op - this is expected */
  end
  otherwise
    say 'Return code of' retval 'is not expected'
    exit 3
end /* select */
return
```

This EXEC calls a procedure **checkID** for each the source and target user ID. If either does not exist or is logged on, the code stops. Otherwise the source and target minidisks are linked at virtual addresses 1100 and 2100. Then the **COPYMDSK EXEC** is called to do the work. Finally the disks are relinquished with the **DETACH** command.

1.9.9 The CLONLNXS EXEC

This EXEC <does this>

```
==> x clonlnxs exec
```

```
/*-------------------------------------------------------------------
THE PROGRAM IS PROVIDED ON AN "AS IS" BASIS, WITHOUT WARRANTIES OR
...
-------------------------------------------------------------------*/
/* CLONLNXS EXEC: clone and start 249 ZVL28xxx user IDs */
do i = 3 to 251
  userid = 'zvl28'||i
  'clon1lnx' userid
  'xautolog' userid
end
```

1.10 Conclusion

In the end the 500 Linux systems were created in slightly over three days. With the automation scripts in place, the golden image can now be modified, copied to the second LPAR and two **CLONLNXS** EXECs can be run, with 500 new virtual servers being available in less than one day.

1.10.1 Obtaining the samples

If you want to try some of the code in this paper, understand that it is not supported by IBM. This paper is on the following Web page under the **Papers** heading:

`http://www.vm.ibm.com/devpages/mikemac/`

Below the PDF of the paper are bullets for **EXECs**, **XEDIT macros** and a **Linux script** to make it easier to get copies of the code.

1.10.2 Thanks

Thanks to Bill Norton, Senthil Bakthavachalam, Mike Wilkins, Eileen Digan and Jim Switzer, all of IBM, for help with this paper.

1.10.3 Feedback

Feel free to contact the author of this paper, Michael MacIsaac, with any feedback or questions. The E-mail address for direct communication is **mikemac** at **us.ibm.com**. Also the linux-390 or IBMVM list servers can be used, and will probably be more effective in getting complete answers.