Introduction to z/VM Hands-on Lab

Martha McConaghy
Marist College

SHARE Orlando
August 2011

Basic Purpose

- Provide “hands-on” experience with basic z/VM administration tasks
- Reinforce concepts introduced in SHARE sessions throughout the week
- Introduce z/VM to people new to system
- Refresh knowledge for people who have been away for awhile.

Lab Specifics

- Divided into 4 sections:
  - CP Administration tasks
  - CMS/System Administration tasks
  - Directory Administration tasks
  - Service Machine Automation
- Two sessions, with SHARE break in between
- Special “second level” VM systems will be used
  - differ slightly from standard VM installation

Exercises in Lab

- Exercises will cover some of the following:
  - Common system administrator tasks such as adding PAGE space to the system, customizing system configuration and modifying the CP directory.
  - Use of common tools such as Xedit (system editor), TRACK, DIRMAINT (CP directory maintenance) and Programmable Operator (PROP).
- Lab text includes explanations for each command. Examples of tasks are included in the appendices.
- Glossary of terms used in lab also available.
- Lab aides are here to help too.

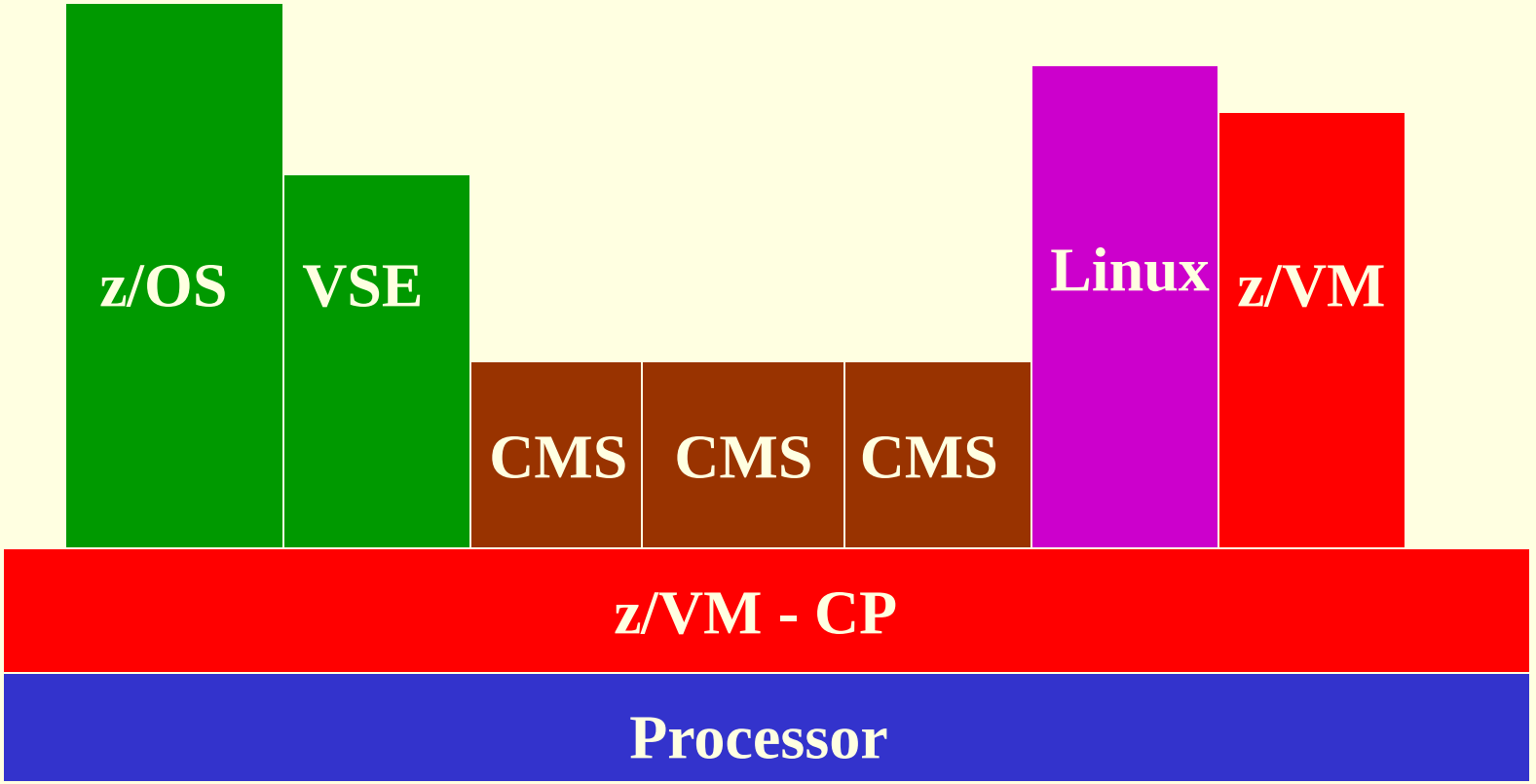
z/VM Basics

- z/VM operating system comprised of 2 main components:
  - CP (hypervisor)
  - CMS (file system, editor, programming tools, etc.)
- Other components used for specific purposes:
  - GCS (run environment for certain products)
  - TCPIP (provides network connectivity)
  - VM SES/E (system maintenance)
  - AVS (APPC/VM VTAM support)
  - TSAF
  - Various priced support features

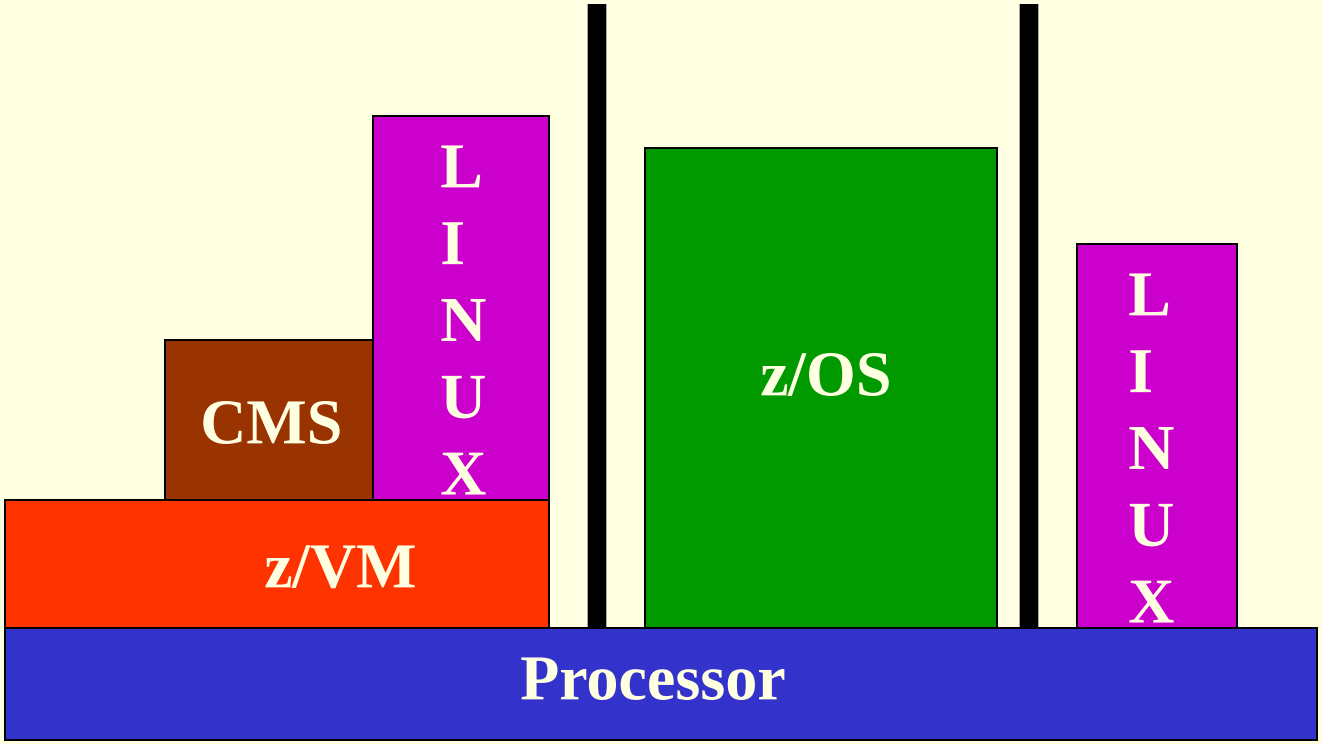
“Virtual Machine”

- **Hypervisor** - provides virtual hardware environment for other operating systems
- **Control Program (CP)** - component of z/VM that provides hypervisor function. It controls hardware resources and shares them among users. Environment is virtualized for each user.
- **Virtual Machine** - individual environment provided by CP to each user.

Basic VM Configuration



LPAR Environment



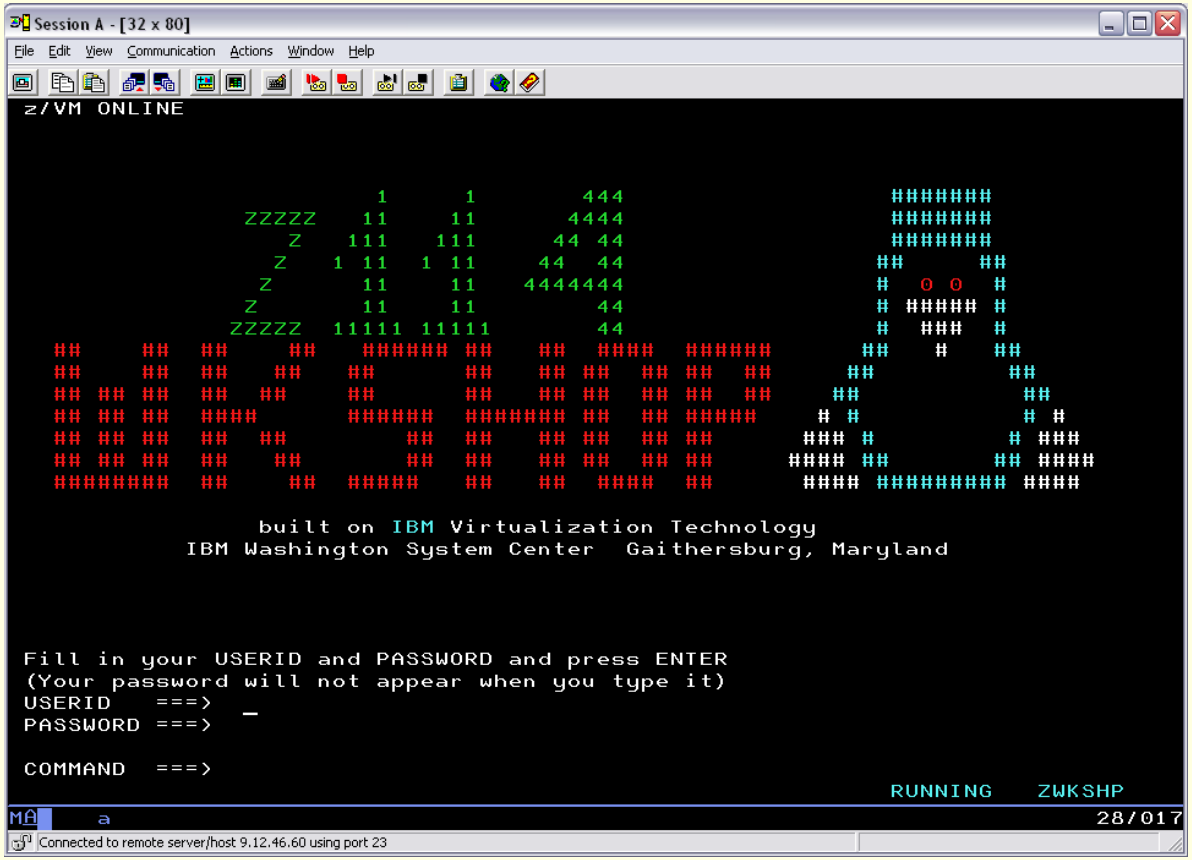
1st vs. 2nd Level

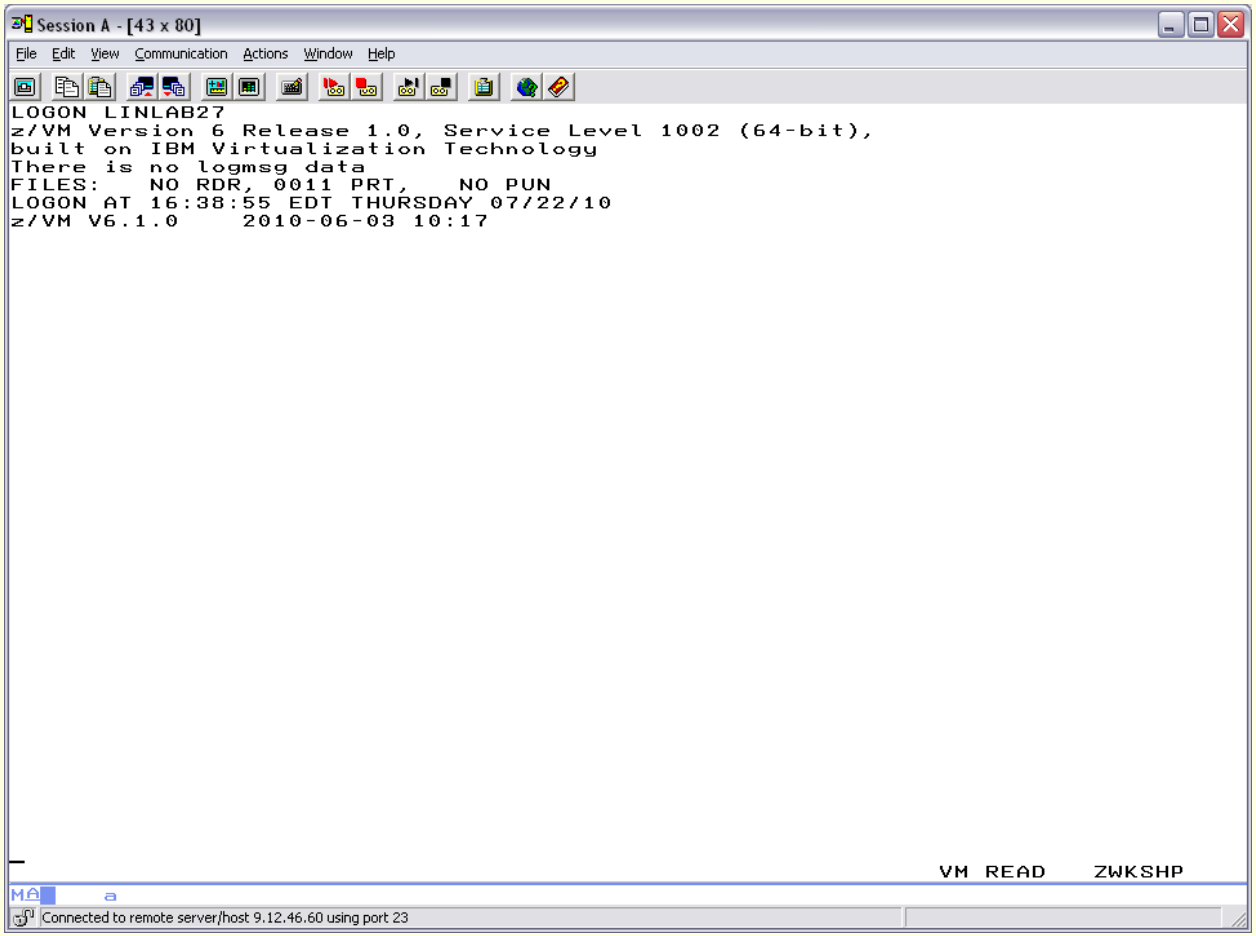
- z/VM running on the hardware whether in LPAR or basic mode, is 1st level VM.
- z/VM running as a guest under z/VM is 2nd level VM.
- A guest system, such as Linux, running under 2nd level VM is considered 3rd level.
- A 2nd level VM system is independent of the 1st level system.
- Uses:
  - test environment for upgrades or major changes
  - training environment for operators, system programmers, etc.
  - migrate older applications to new hardware with little or no disruption.

Getting Started

With the Lab…..

Finally!

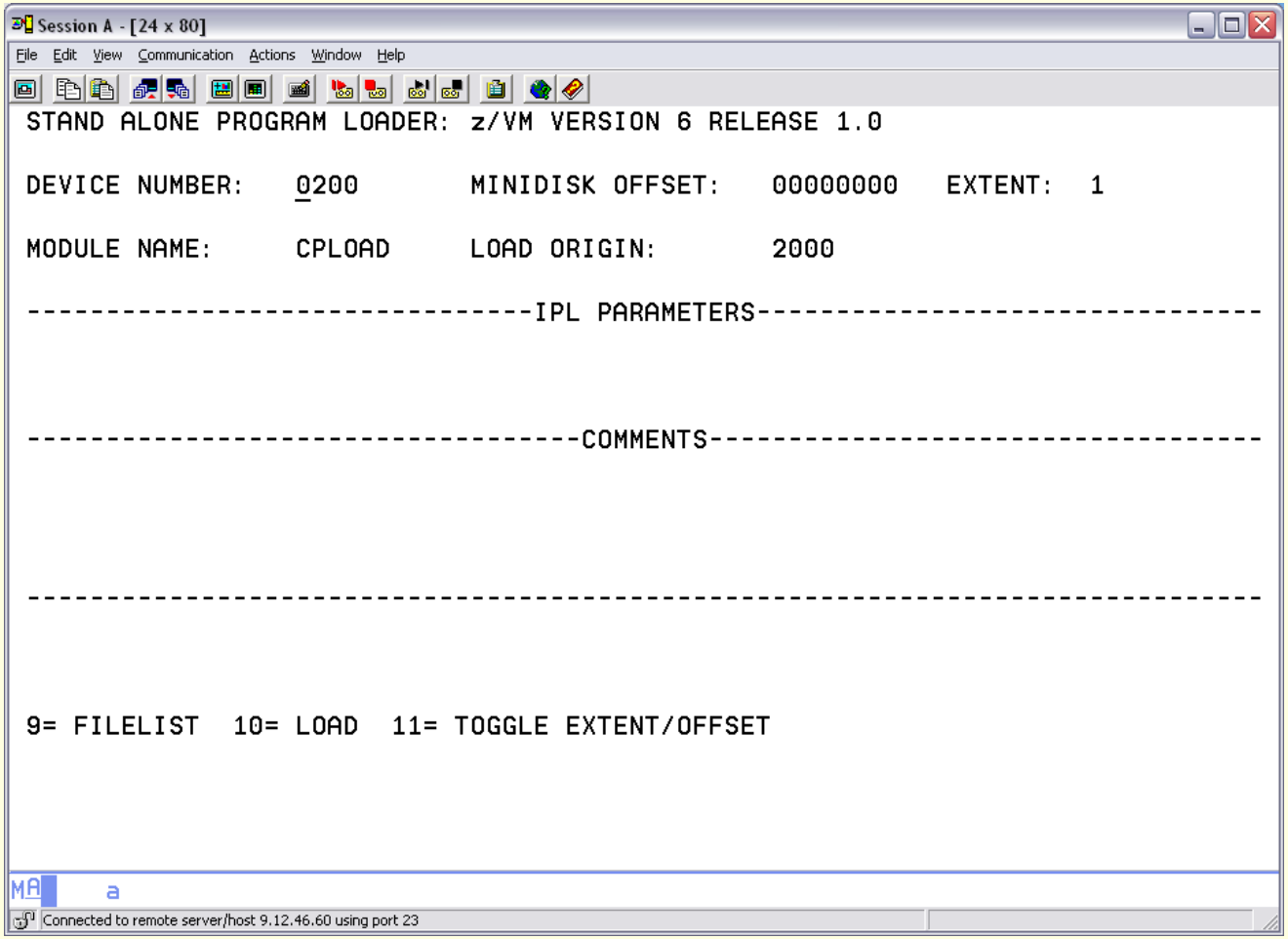

Session A - [32 x 80]
File Edit View Communication Actions Window Help
z/VM ONLINE
built on IBM Virtualization Technology
IBM Washington System Center Gaithersburg, Maryland
Fill in your USERID and PASSWORD and press ENTER
(Your password will not appear when you type it)
USERID ===>
PASSWORD ===>
COMMAND ===>
RUNNING ZWKSHP
MA a
28/017
Connected to remote server/host 9.12.46.60 using port 23

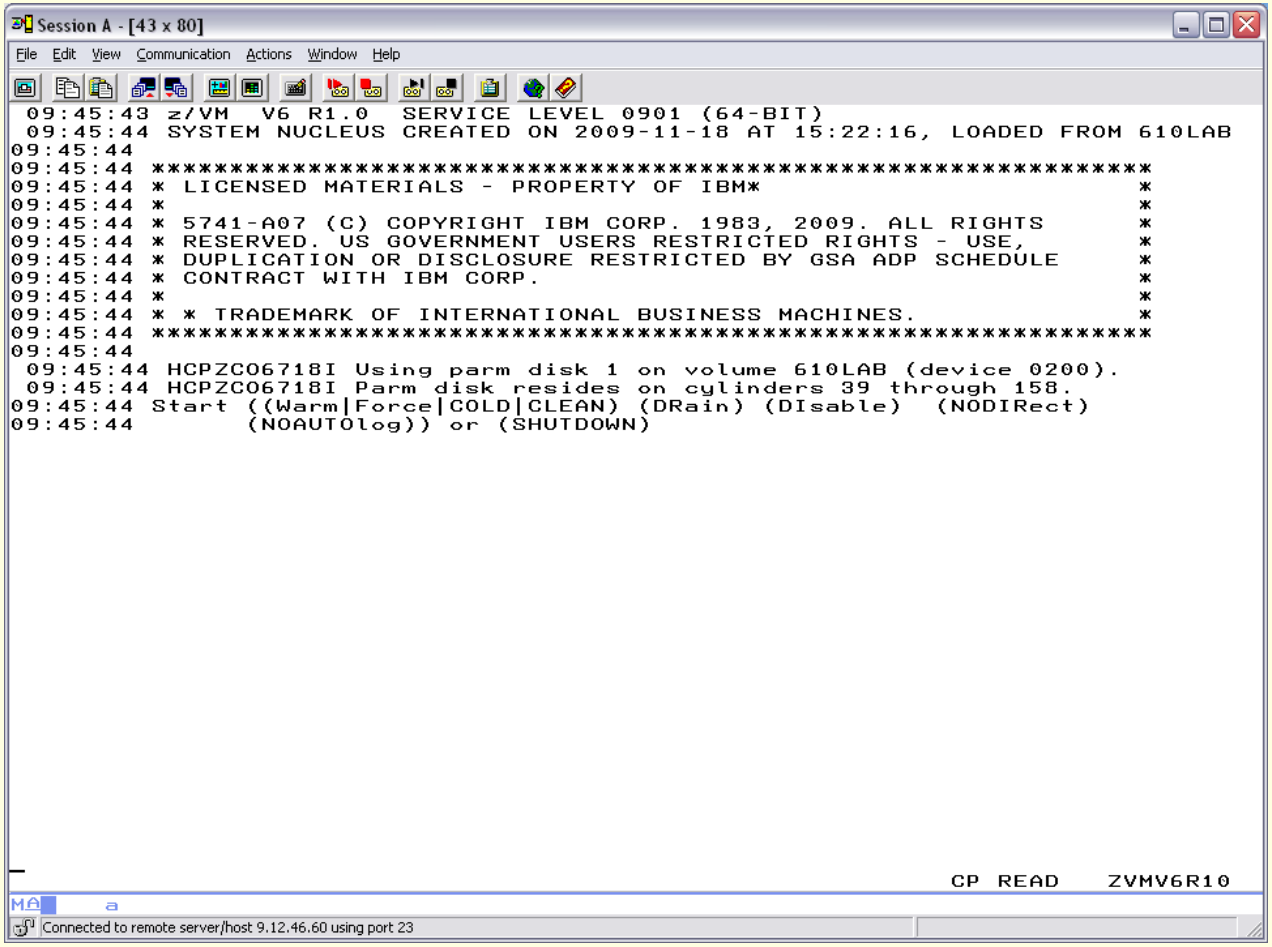

Session A - [43 x 80]
File Edit View Communication Actions Window Help
LOGON LINLAB27
z/VM Version 6 Release 1.0, Service Level 1002 (64-bit),
built on IBM Virtualization Technology
There is no logmsg data
FILES: NO RDR, 0011 PRT, NO PUN
LOGON AT 16:38:55 EDT THURSDAY 07/22/10
z/VM V6.1.0 2010-06-03 10:17
VM READ ZWKSHP
Connected to remote server/host 9.12.46.60 using port 23

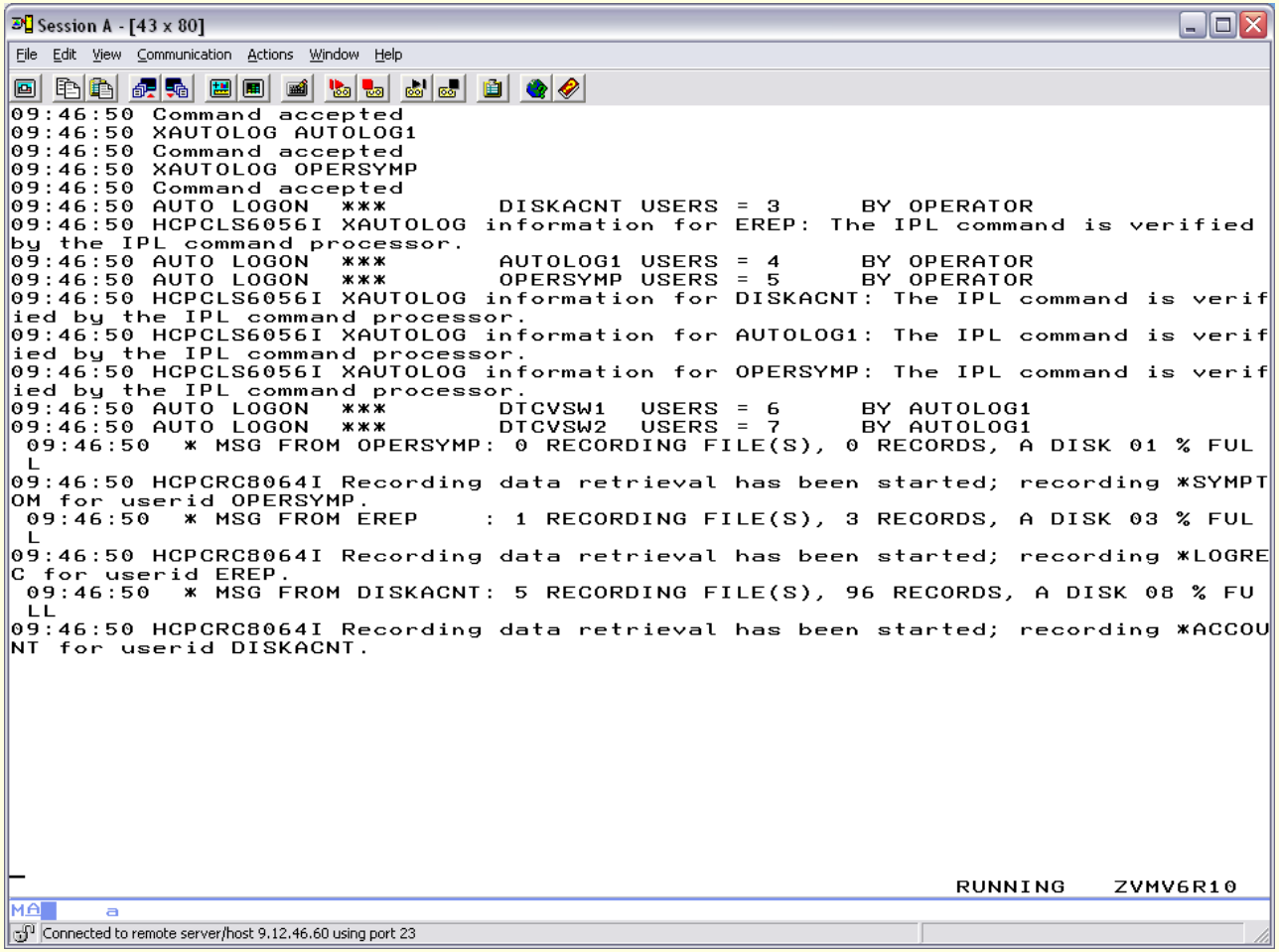
system clear {hit Enter}

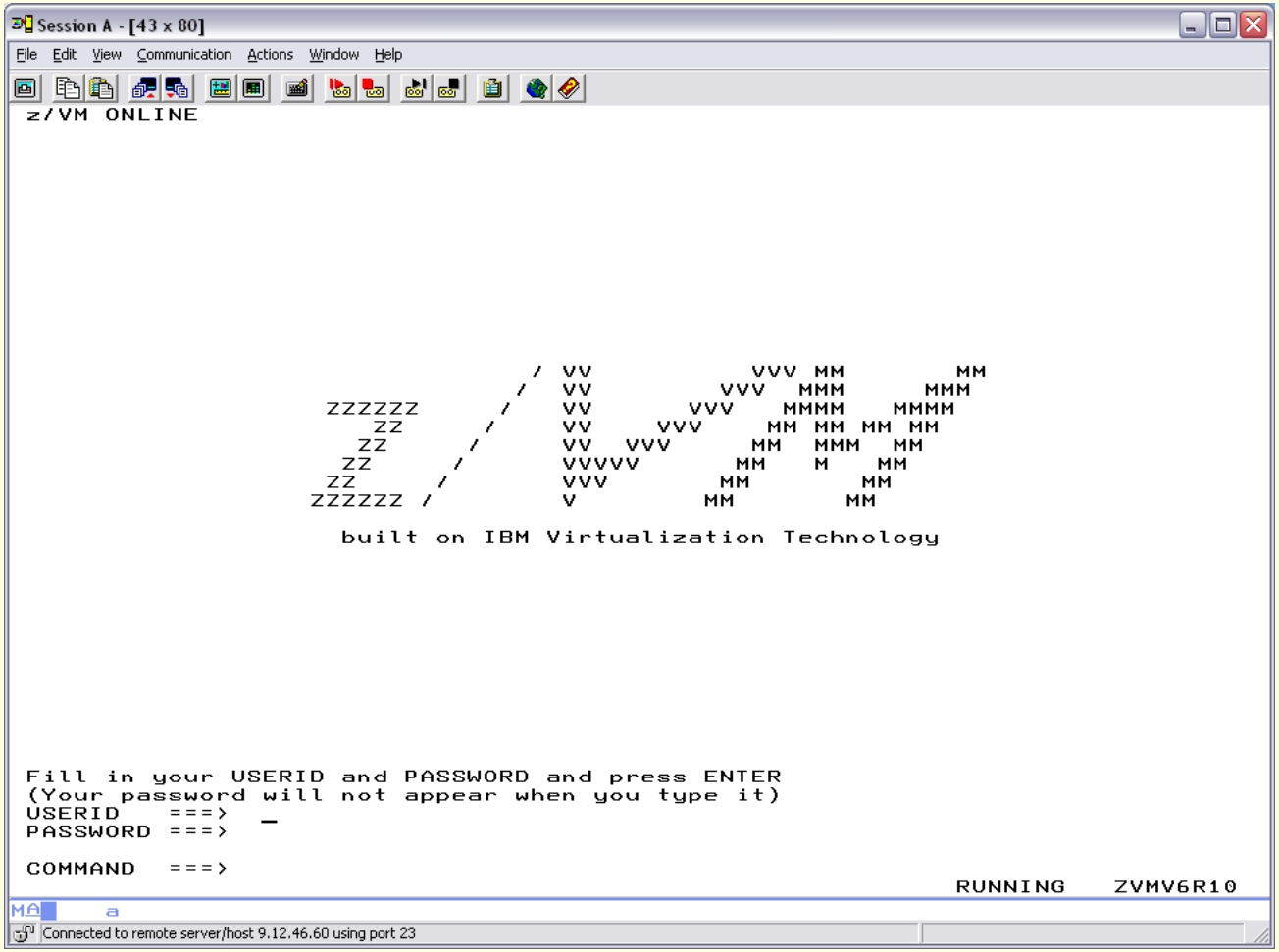
term conmode 3270 {hit Enter}

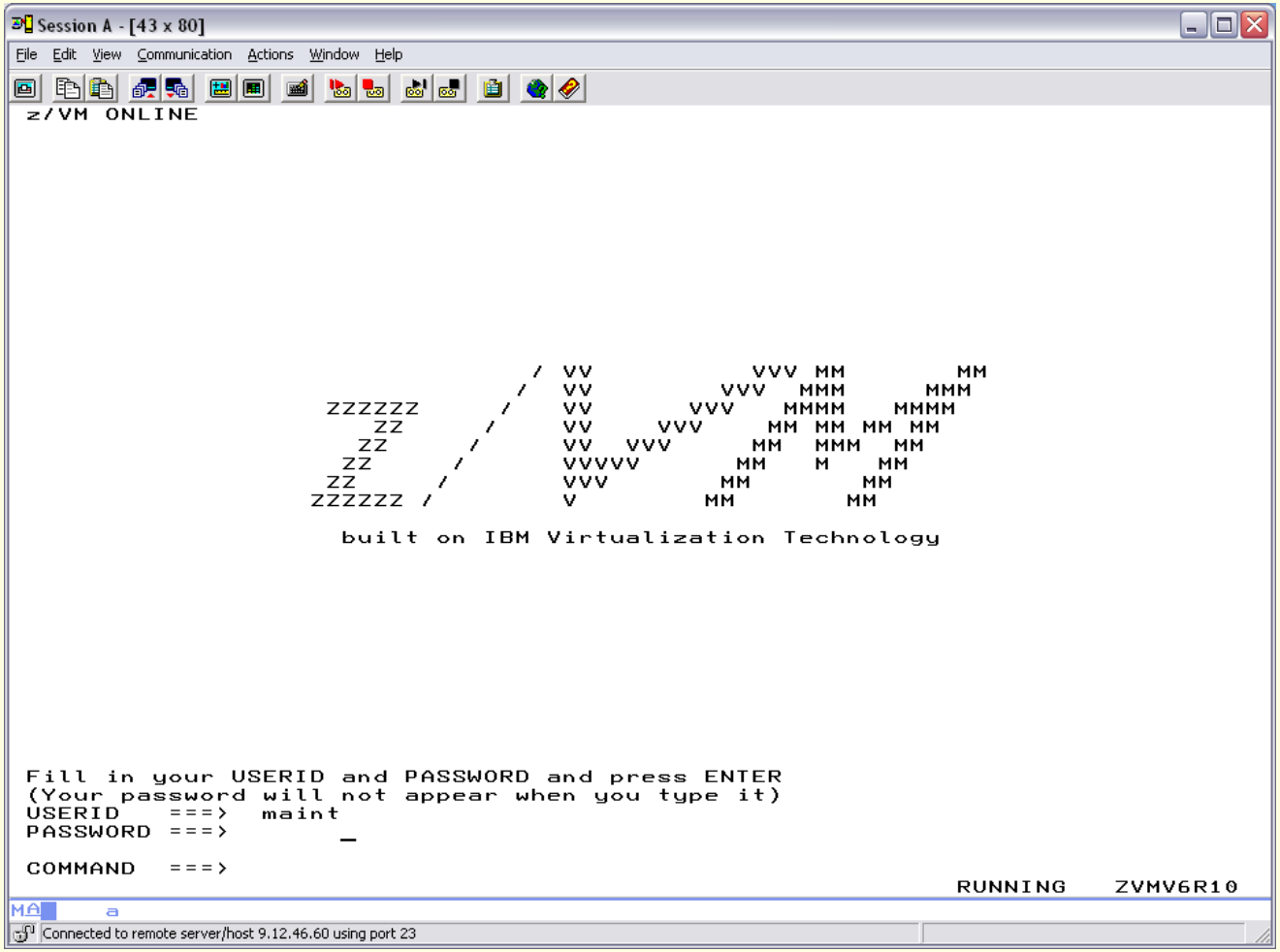
ipl 200 clear loadparm 009 {hit Enter}

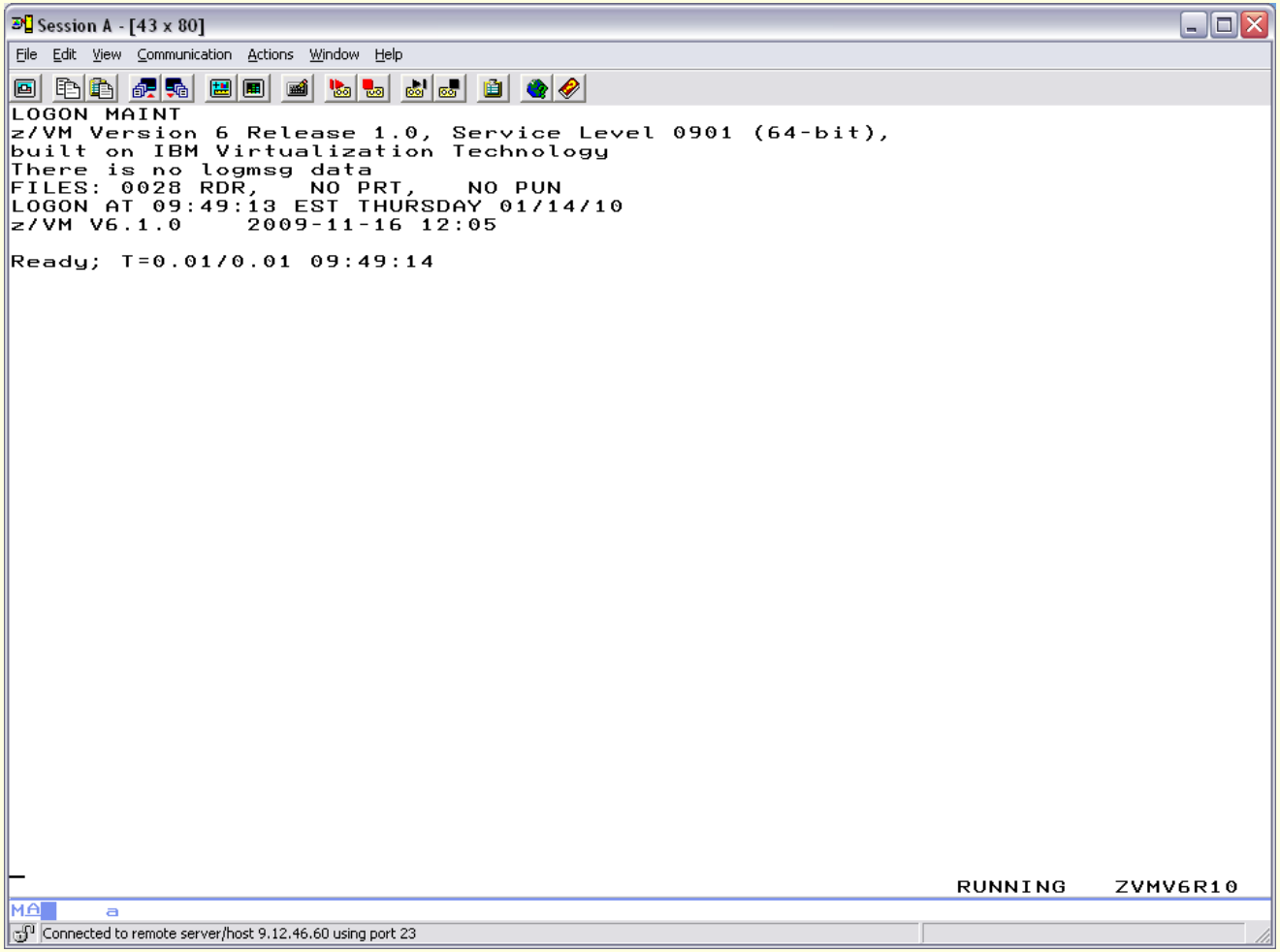

Session A - [24 x 80]
File Edit View Communication Actions Window Help
STAND ALONE PROGRAM LOADER: z/VM VERSION 6 RELEASE 1.0
DEVICE NUMBER: 0200 MINIDISK OFFSET: 00000000 EXTENT: 1
MODULE NAME: CPLOAD LOAD ORIGIN: 2000
----------IPL PARAMETERS----------
----------COMMENTS----------
9= FILELIST 10= LOAD 11= TOGGLE EXTENT/OFFSET
MA a
Connected to remote server/host 9.12.46.60 using port 23


Session A - [43 x 80]
File Edit View Communication Actions Window Help
09:45:43 z/VM V6 R1.0 SERVICE LEVEL 0901 (64-BIT)
09:45:44 SYSTEM NUCLEUS CREATED ON 2009-11-18 AT 15:22:16, LOADED FROM 610LAB
09:45:44
09:45:44 **
09:45:44 * LICENSED MATERIALS - PROPERTY OF IBM* *
09:45:44 * *
09:45:44 * 5741-A07 (C) COPYRIGHT IBM CORP. 1983, 2009. ALL RIGHTS *
09:45:44 * RESERVED. US GOVERNMENT USERS RESTRICTED RIGHTS - USE, *
09:45:44 * DUPLICATION OR DISCLOSURE RESTRICTED BY GSA ADP SCHEDULE *
09:45:44 * CONTRACT WITH IBM CORP. *
09:45:44 * *
09:45:44 * * TRADEMARK OF INTERNATIONAL BUSINESS MACHINES. *
09:45:44 **
09:45:44
09:45:44 HCPZCO6718I Using parm disk 1 on volume 610LAB (device 0200).
09:45:44 HCPZCO6718I Parm disk resides on cylinders 39 through 158.
09:45:44 Start ((Warm|Force|COLD|CLEAN) (DRain) (DIsable) (NODIRect)
09:45:44 (NOAUTOlog)) or (SHUTDOWN)
CP READ ZVMV6R10
MA a
Connected to remote server/host 9.12.46.60 using port 23


Session A - [43 x 80]
File Edit View Communication Actions Window Help
09:46:50 Command accepted
09:46:50 XAUTOLOG AUTOLOG1
09:46:50 Command accepted
09:46:50 XAUTOLOG OPERSYMP
09:46:50 Command accepted
09:46:50 AUTO LOGON *** DISKACNT USERS = 3 BY OPERATOR
09:46:50 HCPCLS6056I XAUTOLOG information for EREP: The IPL command is verified by the IPL command processor.
09:46:50 AUTO LOGON *** AUTOLOG1 USERS = 4 BY OPERATOR
09:46:50 AUTO LOGON *** OPERSYMP USERS = 5 BY OPERATOR
09:46:50 HCPCLS6056I XAUTOLOG information for DISKACNT: The IPL command is verified by the IPL command processor.
09:46:50 HCPCLS6056I XAUTOLOG information for AUTOLOG1: The IPL command is verified by the IPL command processor.
09:46:50 HCPCLS6056I XAUTOLOG information for OPERSYMP: The IPL command is verified by the IPL command processor.
09:46:50 AUTO LOGON *** DTCVSW1 USERS = 6 BY AUTOLOG1
09:46:50 AUTO LOGON *** DTCVSW2 USERS = 7 BY AUTOLOG1
09:46:50 * MSG FROM OPERSYMP: 0 RECORDING FILE(S), 0 RECORDS, A DISK 01 % FULL
09:46:50 HCPCRC8064I Recording data retrieval has been started; recording *SYMPTOM for userid OPERSYMP.
09:46:50 * MSG FROM EREP : 1 RECORDING FILE(S), 3 RECORDS, A DISK 03 % FULL
09:46:50 HCPCRC8064I Recording data retrieval has been started; recording *LOGREC for userid EREP.
09:46:50 * MSG FROM DISKACNT: 5 RECORDING FILE(S), 96 RECORDS, A DISK 08 % FULL
09:46:50 HCPCRC8064I Recording data retrieval has been started; recording *ACCOUNT for userid DISKACNT.
RUNNING ZVMV6R10
MA a
Connected to remote server/host 9.12.46.60 using port 23


Session A - [43 x 80]
File Edit View Communication Actions Window Help
z/VM ONLINE
built on IBM Virtualization Technology
Fill in your USERID and PASSWORD and press ENTER
(Your password will not appear when you type it)
USERID ===>
PASSWORD ===>
COMMAND ===>
RUNNING ZVMV6R10
Connected to remote server/host 9.12.46.60 using port 23


Session A - [43 x 80]
File Edit View Communication Actions Window Help
z/VM ONLINE
built on IBM Virtualization Technology
Fill in your USERID and PASSWORD and press ENTER
(Your password will not appear when you type it)
USERID ===> maint
PASSWORD ===>
COMMAND ===>
RUNNING ZVMV6R10
Connected to remote server/host 9.12.46.60 using port 23


Session A - [43 x 80]
File Edit View Communication Actions Window Help
LOGON MAINT
z/VM Version 6 Release 1.0, Service Level 0901 (64-bit),
built on IBM Virtualization Technology
There is no logmsg data
FILES: 0028 RDR, NO PRT, NO PUN
LOGON AT 09:49:13 EST THURSDAY 01/14/10
z/VM V6.1.0 2009-11-16 12:05
Ready; T=0.01/0.01 09:49:14
RUNNING ZVMV6R10
MA a
Connected to remote server/host 9.12.46.60 using port 23

Introduction to z/VM Hands-on Lab

SHARE Orlando, August 2011

Written and Presented by Martha McConaghy, Marist College

1 Lab Introduction

Welcome to the *Introduction to z/VM Lab* at SHARE. The purpose of this lab is to give attendees a chance to gain "hands on" experience performing some basic z/VM maintenance tasks. The lab is intended for people who have never worked with VM before, or who have been away for a few years and want a "refresher".

The lab exercises are built upon concepts that have been introduced in previous VM project sessions during the week. However, attendance at all of these sessions is not required in order to do the lab. Each exercise contains full explanations of what is being done and why. Some of the steps used in these exercises are based on procedures documented in various IBM z/VM manuals. Others are included based on the author's own experience as a VM system administrator for over 20 years.

All of the exercises are done at your own pace. Lab assistants will be available to help with any problems or questions that arise. The normal SHARE break will occur between sessions. However, attendees are welcome to remain in the lab room and keep working if they wish. **Please note:** all beverages and food should be consumed prior to re-entering the lab room.

1.1 Lab Exercise Skills

Many new z/VM customers are running the system in order to host Linux guests. Others run z/VM for z/OS guests, VSE or to support CMS applications. Regardless of the reasons, there are basic z/VM administration skills that must be mastered in order to maintain a stable and well performing system. The exercises in this lab will cover some of these basic skills. A 2-hour lab cannot cover all of these skills in depth. However, it will help introduce the basic methodology of performing administration tasks on the system.

The lab exercises are divided into 4 main sections:

- CP Administration tasks
- CMS/System Administration tasks
- Directory Management tasks
- Service Machine Automation

All the lab exercises reflect real-life tasks that any z/VM administrator might have to perform. However, some of the details of each exercise will differ from basic z/VM systems. These exercises will be

performed on “second level” systems purposely constructed for use with the labs at SHARE. Some “short cuts” had to be taken to conserve disk space and save time. Wherever necessary, these differences are noted in the text of the exercise.

1.2 Lab Setup

Each workstation in the lab room has been assigned an account on the z114 processor, supplied by IBM to support some of the labs being conducted at SHARE. The z114 is running z/VM 6.1.0. The accounts to be used by each lab attendee are virtual machines on this system, each of which contains a functional z/VM system as well. During the course of the exercises, each attendee will bring up this z/VM system, second level, on the primary z/VM that controls the z114. Each second level z/VM system is self-contained and can be altered without affecting the main, first level system.

The ability to run a copy of the system under itself is due to VM’s virtualization of the hardware environment. VM was the first operating system to do this, and still does it better than anything else. It gives the system administrator a very valuable tool, a fully functional test environment at little or now cost. Among other things, you can use a second level VM system to test upgrades or changes without affecting the main, production system.

1.3 Bringing Up Lab System

On the screen in front of you, there should be a screen displaying the logo for the z114 system being used in the lab.

Action: Place cursor in the Userid field
Type: LINLABxx (where “xx” is your team number)
Action: Move cursor to Password field
Type: LINLABxx
Action: Hit Enter key

The screen should clear and you should see a “VM READ” message in the lower right corner of the screen.

Action: Hit Enter key

You are now logged onto the virtual machine and running CMS. Before continuing with the lab, you will have to bring up the second level VM system to be used.

Type: system clear
Type: term conmode 3270
Type: ipl 200 clear loadparm 009

This series of commands simulates hardware instructions that are normally done on the process. The first clears system memory. That will stop CMS from running and zero out virtual storage. The second makes sure that the console is running in 3270 mode, which is a requirement for z/VM to run. The final

command starts the Initial Program Load of the predefined VM system. The loadparm gives the address of the virtual console.

The IPL will initially bring up the Standalone Loader (SAL). This can be used to change the name of the CP load module, or location of the parm disk. It is used when first installing VM or for emergency recovery if the primary parm disk is lost.

Action: Move cursor to area under the line labeled "IPL Parameters"
Type: cons=009
Action: Hit F10

The second level VM system will now begin to load. You will be prompted for the standard VM questions.

```
Start (Warm|Force|COLD|CLEAN) (Drain) (Disable) (NODIRect) (NOAUTOlog) or (SHUTDOWN)
```

Type: warm

```
Change TOD clock (Yes|No)
```

Type: no

The z/VM system is now completing the IPL process. You will receive a series of messages as various services start initializing. Watch the lower right corner of the screen. When you see "Holding", the screen will stop displaying new messages. To go to the next screen, hit the "Pause/Break" key until "Holding" goes away. When the messages are complete, you will be automatically logged onto the system operator, OPERATOR.

Type: disc

This will disconnect you from OPERATOR. You should receive the standard z/VM logo after hitting the Enter key twice.

Action: Move cursor to Userid field
Type: MAINT
Action: Move cursor to Password field
Type: MAINT (Note, the characters will NOT display)
Action: Hit Enter Key

This will log you onto MAINT, the primary machine for maintaining z/VM and its components. This virtual machine is created as part of the install process and is used to complete it.

2 CP Administration Exercises

2.1 Managing CP System Resources

The primary resources that CP requires in order to run, after hardware, are PAGE, SPOOL and DIRECTORY space. These are special areas of disk space that are allocated and formatted specifically for CP's use. During the installation process, a minimal amount of each is preallocated. This amount, however, is normally not enough to run a production z/VM system. In particular, a z/VM system administrator will have to increase both PAGE and SPOOL within a short amount of time after installing VM. This exercise will demonstrate how to determine how much of each space is allocated, when its time to add more and how to do it.

This exercise must be conducted from the MAINT account on the second level VM system. If you are not logged onto MAINT at this point, please do so before continuing.

2.1.1 Determine current resource allocations

Type: query alloc

This command shows all current resource allocations and where they are located. This includes the PARM disks as well as PAGE, SPOOL, DRCT (directory) and TDSK (temporary disk).

Type: query alloc page

This command shows the same information, but in a summarized format that is easier to read. It includes the percent usage of each allocated area, as well as a total percentage. A good rule of thumb for PAGE space is that it should never be more than 30% totally utilized. Performance starts to degrade after that. Linux guest systems generate a lot of paging for CP. Therefore, this should be watched closely if you are running a number of Linux guests.

```
q alloc page
                EXTENT     EXTENT  TOTAL  PAGES   HIGH    %
VOLID  RDEV      START        END  PAGES IN USE   PAGE USED
------ ---- ---------- ---------- ------ ------ ------ ----
610PAG 0203          1        269  48420      1      3   1%
                                  ------ ------        ----
SUMMARY                           48420      1          1%
USABLE                            48420      1          1%
Ready; T=0.01/0.01 09:50:14
```

Type: query alloc spool

This command shows the amount of SPOOL space in summary. SPOOL is used to store many things including the NSS/DCSS segments, CP abend DUMPs, virtual machine console listings, output to be printed, e-mail files, etc. If CP runs out of PAGE space, it will automatically start using SPOOL as well. Once CP is out of SPOOL space, the system will start to shut itself down. Therefore, SPOOL space is critical to all z/VM systems.

How much SPOOL you need should be determined by watching the usage and projecting future needs. A system with a lot of Linux guests, for example, may need spool to hold large console listings. A system running an IMAP server will need to have sufficient spool to hold incoming e-mail prior to it being processed by IMAP.

2.2 Add more PAGE, dynamically

For the sake of the lab, we will assume that the system needs more PAGE space. The procedure for adding SPOOL is the same.

Where the new space is allocated depends, in part, on the physical characteristics of the disk devices attached to the z/VM system. On older devices, such as real ECKD disks (3380, 3390), it was important to place PAGE or SPOOL near the center of the disk volume in order to get the best performance. This was due to the way the seek arms on the disks functioned. However, with the introduction of emulated disk devices such as the ESS (SHARK) and DS8xxx, this has become less important.

With PAGE space, it is helpful to spread the areas over multiple volumes, on different arrays or LSS (Logical Storage Subsystem on ESS). This helps CP spread the paging load more efficiently. This is less important for SPOOL. Large blocks of space are more useful for SPOOL.

2.2.1 Format and Allocate Disk Space

For the sake of saving time and resources, a disk volume has already been defined for you on the second level system. Its address is **202** and it is an unformatted FBA volume. This is a virtual disk, defined by the first level CP and it is being used to save real disk space. Although it is really virtual, the second level system will treat it as real and it will react as a real disk would react.

Type: query 202

```
query 202
DASD 0202 LINUX
Ready; T=0.01/0.01 12:35:00
```

Type: attach 202 *

This command attaches the disk volume to yourself, in this case, MAINT. A disk volume must not be attached to anything else before issuing this command. Therefore, it's a good idea to query it first to be sure.

Type: cpfmtxa 202

CPFMTXA is a utility provided with z/VM to maintain disk areas required by CP. With this utility, you can format, and allocate CP disk resources. You can also label disk volumes to prepare them for use with CP.

Type: format
Type: 0 end
Type: PAG001
Type: yes

This series answers questions being posed by the utility. There will be a pause while the utility formats the requested area. In this case, that is the entire volume. In real life, you might only want to use a portion of the volume for SPOOL or PAGE. In that case, you would only enter the cylinders or tracks that you wish to use. If the disk volume has not been attached to CP before, then you will also have to format cylinder 0 so that it will be in the correct format for CP to read.

Type: page 0004 end
Type: end
Type: detach 202 (if the format was successful)

Once the format is complete, CPFMTXA will ask for allocation definitions. In this case, we are defining PAGE for CP. **Note:** the allocation is starting at block 4, not 0. The first cylinder (0 for ECKD) or blocks (for FBA) on a disk volume are used by CP for the label and allocation map. If you were to allocate SPOOL or PAGE over this area, it would wipe out this vital information and CP would have problems.

The utility confirms the change after receiving the END command. If you were using an ECKD device such as a 3390-3, cylinder 0 would show up as PERM by default. Any space on a disk volume that is NOT going to be used by CP for its resources is labeled as PERM (short for permanent). CP will not attempt to write to these areas. That is the allocation default for all volumes. Cylinder 0 on an ECKD device should ALWAYS be allocated as PERM.

If you had problems completing this exercise, refer to Appendix A: Sample CPFMTXA Listing, page 35 for a sample listing of the process.

2.2.2 Define CP Owned Volume

All CP disk resources must reside on volumes defined as "CP Owned". You can allocate SPOOL, PAGE, etc. on disk volumes that are not "CP Owned". However, CP will not find and use them.

A volume can be added to the list of "CP Owned" volumes dynamically, without disrupting the running system. However, this change is not permanent and will go away at the next IPL of CP. In this exercise, you will define the volume dynamically and then follow the procedure to make it permanent.

Type: query cpowned

You will see a list of disk volumes, their labels, addresses and what "slot" they are assigned to. The "slot" is very important. This indicates the order in which CP processes these volumes when it IPLs. The integrity of SPOOL depends on this order. Therefore, once you assign a SPOOL volume to a particular slot, you MUST NOT change it. If the slot assignments for SPOOL volumes are changed, you will end up having to do a COLD start of CP. This will wipe out all files in SPOOL, including NSS/DCSS and other vital files. Therefore, deleting or reordering CP Owned volumes should be done with a great deal of caution.

Slots that are labeled "reserved" are being held open for a new volume to be added. You want to be sure to always have a few "reserved" slots defined. Without these slots, you cannot add a volume to the CP Owned list dynamically.

The first "reserved" slot on this system should be slot 7.

```
q cpowned
Slot  Vol-ID  Rdev  Type   Status
   1  610LAB  0200  Own    Online and attached
   2  610SPL  0204  Own    Online and attached
   3  610PAG  0203  Own    Online and attached
   4  610W01  F124  Own    Online and attached
   5  610W02  F125  Own    Online and attached
   6  610RES  F123  Own    Online and attached
   7  ------  ----  -----  Reserved
   8  ------  ----  -----  Reserved
   9  ------  ----  -----  Reserved
  10  ------  ----  -----  Reserved
  11  ------  ----  -----  Reserved
  12  ------  ----  -----  Reserved
  13  ------  ----  -----  Reserved
  14  ------  ----  -----  Reserved
  15  ------  ----  -----  Reserved
  16  ------  ----  -----  Reserved
  17  ------  ----  -----  Reserved
  18  ------  ----  -----  Reserved
  19  ------  ----  -----  Reserved
  20  ------  ----  -----  Reserved
Ready; T=0.01/0.01 10:05:
```

Type: define cpowned slot 7 pag001
Type: attach 202 system pag001
Type: query cpowned
Type: query alloc page

The disk has now been defined as a CP Owned volume, and attached to the system. CP cannot begin to use the disk volume until the attach is done. By querying the PAGE allocation, you can confirm that the space has been seen by CP and is now being used.

As was mentioned earlier, this process is NOT permanent. If CP were to be shut down, crash or lose power, this disk volume would no longer be "CP Owned" and the PAGE space would be unavailable to CP.

2.3 Updating System Configuration

All permanent changes to CP have to be done through the system configuration file, SYSTEM CONFIG. This file is stored on the CP PARM disks and is read when CP IPLs. Since this is an extremely important file for CP, its always a good idea to keep more than one copy. To make it easier to move to a new release of z/VM, I usually keep a copy on the MAINT 2C4 disk (CP Localmod).

2.3.1 Back up current PARM disk

When making changes to the production VM system, it's always a good idea to back up the main PARM disk before making the changes. Then, if you have made a mistake that prevents the system from IPLing, you can switch to the alternate PARM disk to get the system running again.

Type: query cpdisks

```
q cpdisks
Label  Userid   Vdev Mode Stat Vol-ID Rdev Type   StartLoc     EndLoc
MNTCF1 MAINT    0CF1  A   R/O  610LAB 0200 CKD          39        158
MNTCF2 MAINT    0CF2  B   R/O  610LAB 0200 CKD         159        278
MNTCF3 MAINT    0CF3  C   R/O  610LAB 0200 CKD         279        398
Ready; T=0.01/0.01 10:09:09
```

This command lists all the PARM disks defined to the system. CP will come with 3 defined by default. The first one in the list is the primary, and is normally the one used when CP is IPL'ing. The second is an emergency backup disk, which can be used if there is something wrong with the primary. This change is done with the standalone loader when IPLing CP. The third is an extra disk, which can be used as a tertiary backup or for other custom programs. For example, at Marist College, the third disk is used for the text decks of CP exit programs.

Type: cprelease b
Type: link * cf2 cf2 w
Type: access cf2 z
Type: access cf1 x

For the first step, we are going to make a back up copy of the primary PARM disk, in case the changes we make to the CONFIG file are bad. This back up does not have to be done every time a change is made to CP. However, it's a good idea to do it periodically to ensure that you have one functional PARM disk.

On a basic installed z/VM system, MAINT has read/only links to all three PARM disks. These commands tell CP to release the second PARM disk so that it can be linked in read/write mode by MAINT. If you do not have CP release the disk first, you will receive an error when you try to link it r/w.

Type: copy * * x = = z (olddate replace
Type: release z (detach
Type: cpaccess maint cf2 b
Type: query cpdisks

This process uses the CMS COPYFILE command to copy all the files from the primary disk accessed as X to the back up disk, accessed as Z. It then removes the back up disk from MAINT and reaccesses it for CP.

If you had difficulty completing this exercise, please see Appendix B: Sample Process for Backing Up PARM Disk on page 37 for a sample listing of the process.

2.3.2 Updating SYSTEM CONFIG File

Type: xedit system config x
Type: set fm a *Sets filemode to A so you can save it.*

Xedit is the full screen editor for z/VM. It allows you to modify most CMS based files. There is a special option that also allows you to edit files in the Byte File System. Xedit is an extremely powerful which can be exploited to do many things. In this lab, however, we are going to use only its basic functions. People who are familiar with the ISPF editor on z/OS will find some things familiar with Xedit. People who are only familiar with Unix or Linux based editors may find it a bit strange at first.

There are 3 basic areas of the screen, while in Xedit, which you need to understand. First is the data area, which takes up most of the screen. The lines of the file you are editing appear in that area. You can, at any time, move the cursor to the data area and make changes to a line. You move the cursor using the arrow keys on most standard keyboards. You can also move the cursor by pointing the mouse where you want it to go and clicking.

The second area is the "prefix" area that normally appears to the left of the data area. This can show up as a series of equal signs "======" or as line numbers. The prefix area is used to enter commands that will take affect against the line next to it. (The prefix area in the ISPF editor works the same way, though the commands are slightly different.)

The third area is the "Command Line", which can be at either the top or bottom of the screen depending on how you have customized your Xedit session. The Command Line is used to enter Xedit, CMS or CP commands that are not included in the prefix area commands.

The other important thing to be aware of are the "program function keys" (PF keys). These keys are normally defined with commands that help you manipulate the data area with ease. The PF key functions are usually mapped to the F keys on standard PC keyboards by the emulator. PF 7 and PF 8 are the most commonly used. PF 7 pages you up in the file, while PF 8 pages down.

Action: Use the PF 8 key to page down in the file until you see the *"CP_Owned Volume Statements"*. This should appear around line 66 in the file.

The list of CP Owned volumes should be familiar from the previous exercises. In "Slot 7" should be the word "RESERVED".

Action: Using the arrow keys, move the cursor to the line containing "Slot 7" and over to where it reads "reserved". Type the label of the new volume over "RESERVED".

The line should now appear as follows:

```
CP_Owned   Slot   7  PAG001
```

Action: Hit the Enter key. The cursor should automatically go back to the Command Line.

Type: file

The FILE command will save the file to disk and then exit Xedit. When you first entered Xedit, the exercise had you change the file mode to "A". That allows you to save a copy of the file on the A mode disk. If you have not done that, you will receive an error when you try to write the file to disk. That is because the CF1 disk in file mode X is still in read/only access. If you get the error, simply issue the command "fm a" again and then issue "file".

A copy of the system configuration is now saved on the A disk of MAINT. It is not yet available to CP, however. That requires it being moved to the primary PARM disk.

2.3.3 Updating Primary PARM Disk

The primary PARM disk must now be accessed in read/write mode so that the config can be moved to it. While doing this, it's a good idea to do it as quickly as possible. In order to get the primary disk in read/write mode, you have to remove it from CP. While it is gone, CP will look to the secondary disk for files such as the logo definitions. If you have not customized the logo, this is not a problem. However, if you are running with a customized logo, you should make sure that a copy of it is on the secondary PARM disk. Otherwise, new sessions may start to show up with the default CP logo. Other customizations such as CP exit programs could be affected as

well. Therefore, it is always a good idea not to start this process and then not finish it.....take that phone call or coffee break when you are done.

Type: cprelease a
Type: link * cf1 cf1 w
Type: access cf1 x

These commands release the primary PARM disk from CP and link it in read/write mode. It's then reaccessed at mode X.

Type: erase system confold x

The first time you do this command, you will get a "file not found" message. That is OK.

Type: rename system config x = confold x
Type: copy system config a = = x
Type: release x (detach
Type: cpaccess maint cf1 a sr

This series of commands creates a back up copy of the current system configuration and then copies the new one to the PARM disk. It's a good idea to keep a copy of a recent, good system configuration on the PARM disk. Then, if there is any problem with the new one, you can point the standalone loader to the old one and still get CP to IPL. (Once you have been burned by a bad system config file, you'll never go without a back up again....trust me on this.)

Once the PARM disk is reaccessed to CP, the change is now permanent. The next time CP is IPL'd, the new volume will be CP Owned.

If you had difficulty completing this exercise, please see Appendix C: Sample Procedure for Updating Primary PARM Disk, page 39 for a sample listing of the process.

2.4 Protecting the New PAGE Space

One step that is often forgotten is protecting the new CP PAGE space from accidentally being "stepped on" when allocating a minidisk or other data area. This is an easy thing to do since the CP Directory has no way of knowing the new PAGE area. Therefore it is ALWAYS a good idea to protect the new volume with "dummy" minidisks in the Directory. This ensures that no one will accidentally allocate over the area.

As installed, z/VM comes with a basic Directory already set up. In it are defined accounts such as MAINT. There are also some "dummy" accounts that are used to simply hold minidisks. These accounts have a password of NOLOG. This is a special word to CP that means the account can never be logged onto the system. In fact, if you try to query the account, CP will report that it does not exist.

Two of the dummy accounts pre-defined in the Directory are $SPOOL$ and $PAGE$. These are used to hold minidisks for the areas where SPOOL and PAGE are allocated. Since the accounts can never be logged on, these minidisks act as holders for the space, preventing DIRMAINT, VM:Secure or other Directory product cannot allocated over it.

On the lab system, there is no Directory product configured as of yet. Therefore, the exercise will have you make a manual change to the Directory and put it online.

Type: xedit user direct
Type: /user $page$

The basic source Directory supplied by IBM is in a file named USER DIRECT on MAINT. The commands put you into an Xedit session of the file. The second command is the Xedit Locate command. It will locate the record in the source Directory file where the account, $PAGE$ starts.

Under the "USER $PAGE$" line should be at least one line starting with the word "MDISK".

```
00100 USER $PAGE$   NOLOG
00101  MDISK A03 3390 000 END 610PAG R
00102 *
```

Action: Move the cursor to the prefix area next to the "MDISK" line.
Type: i [Hit Enter]

After you type "i" and hit the Enter Key, you should see a new, empty line appear just below the "MDISK" line.

Action: Move cursor to the first character of the new line, if it is not already there.
Type: MDISK 202 9336 000 END PAG001 RR
Action: Hit Enter so cursor moves to Command Line.
Type: file

The line that is being added to file defines a minidisk over the area of the new volume. Note, it includes the PAGE area as well as the allocation map at the beginning of the volume. This area must be protected as well as the PAGE area. You can include it in this minidisk as above. Or you can add a separate minidisk to the $ALLOC$ account using the same process.

Type: diskmap user direct a (doends
Type: xedit user diskmap

DISKMAP is a utility provided to help you allocate minidisks before you are able to activate a Directory manager. It takes all the minidisks defined in the source Directory file and maps them out. This mapping will warn you if you have allocated more than one minidisk in a specific spot.

It will also tell you where there are gaps in allocations, which are candidates for a new minidisk. It's very important that you verify any changes in minidisks before putting the source Directory online. It's very hard to fix a problem once that has been done.

The entry for volume PAG001 is located at the top of the file. It should show the minidisk you just created for $PAGE$. If you cannot find it, then there is a problem with the minidisk definition. You should return to the USER DIRECT file and check the minidisk record for problems.

```
00015 ---------------------------------------------------------------------
00016
00017 VOLUME   USERID      CUU   DEVTYPE      START         END        SIZE
00018 PAG001   $PAGE$      202     9336  0000000000         END          ?
00019
00020 ---------------------------------------------------------------------
```

Once you have verified that the change to the source Directory is correct, do the following command:

Type: directxa user direct

You should receive a message that the directory has been updated and is online. When you do, the procedure has been completed.

Note: if you receive an error when putting the directory online, do NOT log off MAINT. Go back to the source directory, correct the problem and reissue the DIRECTXA command until the errors disappear. If you do not do this, you run the risk of disabling the system and being unable to log back on to fix it.

2.5 ReIPLing System and Verifying Changes

The moment of truth is when CP is recycled and you verify that the changes you have made did not disappear.

On a true, production z/VM system, a process should be set up to allow an operator or system admin to shutdown any guest systems prior to recycling CP. In the case of the lab systems, this is not necessary.

Type: shutdown reipl

The REIPL parameter of the SHUTDOWN command will cause CP to shut down, then automatically start to reIPL. It will bypass the normal questions that are asked when CP is initializing (i.e. "Warm start" and "Change TOD clock"). The SHUTDOWN command is a

privilege class A command. Therefore, either the VM operator machine must issue it; MAINT or another class A machine.

Since the operator machine was running in disconnect mode when the SHUTDOWN command was issued, CP will automatically disconnect it after it finishes initializing. If the IPL is successful, you should now be at the z/VM logo.

Action: Log back onto MAINT, password MAINT
Type: query alloc page

The results of the command should verify that the new page area you added in this exercise is still there and in use. If it isn't, go back over the exercise and see if you can identify what went wrong.

2.6 Customizing CP

There are numerous tasks that a VM administrator can do to customize the system to better meet the expectations of their users. Some are fairly simple, such as changing the identifier (ID) of the system to match local standards, or setting the time zone to match local time. Others can be more complex, such as customizing the logo that appears on terminals or TN3270 sessions. The level of customization is up to the administrator and their management. The CP portion of z/VM will run fine without any customization at all. However, it may be less convenient to use.

In this exercise, we will perform the following customization tasks:

- Update the system identifier
- Change the privilege class of a CP command
- Change the time zone, temporarily and permanently
- Enable the use of a product

All, but the change in the system identifier, can be done temporarily through the use of CP commands. These commands will be listed in the exercise. However, the main purpose of the exercise will be making these changes permanently.

Action: Log back onto MAINT, if you are not there already

2.6.1 Dynamic Customization

2.6.1.1 Setting the Timezone

Many z/VM systems are configured for local time zones. CP is not capable of getting its time from a central server like Unix and Linux systems can. Therefore, it has to be pre-configured for the correct zones. At installation, z/VM is set for the Eastern Time zone of the US. If your local time zone does not match EST or EDT, then you will want to change the configuration to reflect the proper offsets from UTC.

In addition, z/VM is pre-configured to automatically change time zones on the date set by the calendar. You may wish to change these definitions if you are also changing the base time zone definition.

Although z/VM is pre-configured to change zones at the appropriate time, you may need to do it earlier, as a test, for example. This can be done dynamically, while CP is running. However, it must be done with care as not all applications can handle the change gracefully.

This change is only temporary until the official time and date have past.

Type: query timezone

Note the active time zone (EDT for the current lab). Pretend that the system is being run in California, on Pacific Daylight Time. We, then need to change the timezone to PDT.

Type: set timezone pdt
Type: query timezone

Enter the time zone opposite of the current setting. The VM system time will now move either 1-hour earlier or later, depending on the time zone specified.

2.6.1.2 Updating Command Privilege Class

As with the previous exercises, this will temporarily change the privilege class of a CP command or diagnose. This change will be lost when CP is recycled.

Type: cp modify command msgnoh ibmclass b priv bm

2.6.2 Permanent Customization

2.6.2.1 Permanent Time Change

In this exercise, we will change the system to Pacific Daylight Time (PDT). If you completed Updating System Configuration , page 8 successfully, there should be a copy of SYSTEM CONFIG still on your A disk.

Type: xedit system config a
Action: Use PF8 to page down in file until you see the “Timezone_Definition” records, approx. line 41.

Action: Notice that the standard North American timezones are already predefined. If you run in a timezone different than the ones in this list, you can add it by duplicating the entry and adding your own offset and timezone name.

Action: Use F8 to page down one page. Move cursor down to "Timezone_boundary" statements applicable to 2009 (or 2010 if the date of this workshop is later than March 2010) and replace "EDT" and "EST" with "PDT" and "PST".

Action: Hit Enter key to go to command line.

Type: save

The SAVE command will preserve a copy of the file on disk, but keep you in Xedit. Once this copy of the SYSTEM CONFIG is put on the PARM disk, the time zone change will be permanent. However, we are going to make a few more changes first.

2.6.2.2 Customizing the System Identifier

The system identifier is 8 character name that appears at the bottom right side of the screen on all 3270 sessions, real or emulated. On a basic installed z/VM system, this name normally refers to the release of z/VM, such as ZVMV6R10. This can be confusing to users who see it. Also, if you have multiple z/VM systems running (first or second level), it can be difficult to tell which you are connected to at a given time. Updating the system identifier can make that easier. You should still be in an Xedit session of the SYSTEM CONFIG file, the cursor on the Command Line.

Type: /System_Identifier

You should see a line that looks like the following:

System_Identifier_Default ZVMV6R10

Action: Move the cursor over "ZVMV6R10" and type "VMLAB ". Hit Enter to go to the Command Line.

Type: save

As with the timezone change, the system identifier change will become permanent when we place the new config file on the PARM disk and reIPL the system.

2.6.2.3 Changing Command Privileges

The basic privilege classes for CP commands can be changed or extended to meet the needs of the local system or security. IBM supplies basic command classes A through G. Other classes can be defined both dynamically and permanently. For example, many products require class B to be assigned to a service machine so that it can use the MSGNOH (message no header) command. Class B is a very strong privilege, which includes other important commands that could be a security problem. Therefore, it might be desirable to assign a different class to MSGNOH. That

class can be then assigned to the service machine so that it can use MSGNOH, but not other class B commands.

Action: Use PF8 to page down to the bottom of the SYSTEM CONFIG file.
Action: Move cursor to prefix area to the left of last line in file.

Type: i [hit Enter]

Type: modify command msgnoh ibmclass b priv bm

Action: Hit Enter key to go to Command Line.

Type: file

With these commands, you are inserting a new line into the SYSTEM CONFIG file that will add the privilege class "m" to the MSGNOH command. It will take effect whenever CP is IPL'd. In addition, you can use the MODIFY COMMAND command to dynamically change the class. Class M can now be assigned to any VM account so that it can use the MSGNOH command.

2.6.3 Put Changes Online

Action: Use process in exercise **Updating System Configuration, 2.3.3** to put new SYSTEM CONFIG file on primary PARM disk.

Type: shutdown reipl

After system has reIPL'd, log back onto MAINT. Verify that the changes have taken effect.

Type: query time

Type: query cpcmd msgnoh

The "PrivClasses" field should include "M".

Action: Look at the lower right side of your screen. The new system identifier should appear there.

If any or all of these changes did not take effect, repeat this exercise and try to identify what went wrong.

3 CMS Administration Exercises

The CMS environment is vital to the maintenance of z/VM as well as many of the products that run on it. There are a variety of steps that can be taken to customize this environment depending on the requirements of the local site. One very useful step is setting up the SYSTEM NETID file.

3.1 Customizing the SYSTEM NETID file

Many products that run on z/VM such as RSCS and TCPIP use the CMS command, ID, to determine the name of the system it is running on. These products can make decisions based on this information. For example, most TCPIP services can choose a configuration file based on the system ID. This allows multiple configuration files to reside on the same minidisk, reducing the complexity of maintaining multiple systems.

The CMS command, ID, gets its information from the SYSTEM NETID file, which resides on the CMS system disk, 190. At the minimum, the SYSTEM NETID file should be set up to match your main production z/VM system. More sophisticated additions can be made to this file later.

As supplied by IBM, the SYSTEM NETID file looks like the following:

```
*CPUID NODEID NETID
```

The intention is that the VM system administrator will add relevant records to this file following the header.

- **CPUID** – refers to the serial number of the processor (real or virtual)
- **NODEID** – refers to the system identifier for z/VM system on that processor
- **NETID** – refers to the name of the RSCS machine servicing that system

Action: If you are not still logged onto MAINT, log onto it now.

Type: query cpuid

This command will return the value of the virtual CPU serial number assigned to this virtual machine. Unless it is set by command or in the CP directory, the CPU ID will default to the value of the real serial number of the processor running on z/VM. In the case of the lab, however, the systems are running as guests on the main z/VM system. Therefore, their "real" CPU ID is also virtual. For the sake of the lab, it makes no difference.

The serial number is digits 3-8 of the number returned by the command. Following the serial number is the model number of the processor. This portion of the CPUID cannot be changed. The serial number, however, can be changed using the SET CPUID command or using the CPUID option in the Directory.

In this lab, the virtual CPUID should look like the following:

```
q cpuid
CPUID = FF11110720988000
Ready; T=0.01/0.01 08:13:24
```

Make note of the CPUID.

Type: vmfsetup zvm cms
Type: xedit system netid s
Type: fm a

The first command accesses the VMSES disks for CMS. The last command changes the file mode of the SYSTEM NETID file in memory to A, so that you can save it after making the necessary changes.

Action: Move cursor to the prefix area on the left side of the screen, next to the 1 line in the file.

Type: i [Hit Enter]

Action: In the new, blank line, type the serial number displayed by the QUERY CPUID command, *space*, the new system identifier, *space*, *RSCS*. The new line should look something like the following:

111130 VMLAB RSCS

Note: the text MUST be in UPPERCASE.

Action: Hit the Enter key to go to the Command Line. File it and exit Xedit.

Type: access 190 x
Type: rename system netid x = netid@ x0
Type: copy system netid a = = x (olddate replace
Type: release x

The first command accesses the CMS system disk (190) in file mode X. Normally, it is accessed in file mode S, where it cannot be changed. The following two commands preserve a copy of the original file and then moves the new one onto the system disk.

3.2 Resaving CMS

Whenever the CMS System disk (190) or System Extension disk (19E) are changed, the CMS NSS segment must be resaved. The CMS segment contains pointers to all the files on both disks.

This helps CMS to run more efficiently. However, whenever a file on one or both of the disks is changed, it causes the pointers to be out of sync with the files on the actual disks. This will result in a message like one of following occurring when logging on or IPLing CMS:

Shared S-Stat not available
Shared Y-Stat not available

CMS will continue functioning, but it won't be as efficient as it could be. If there are many CMS users, this can adversely affect performance.

Resaving the CMS segment is not difficult. It's a two step process consisting of creating the new segment and issuing a command to resave CMS into it.

Type: sampnss cms

The SAMPNSS exec is provided by IBM to aid in the creation of the CMS NSS. It issues a DEFSYS command to actually create it. This command creates the NSS shell.

Type: query nss name cms

```
query nss name cms
OWNERID  FILE TYPE CL RECS DATE  TIME     FILENAME FILETYPE ORIGINID
*NSS     0029 NSS  A  1302 11/18 12:24:26 CMS      NSS      BLDCMS
*NSS     0030 NSS  S  0001 01/14 08:16:01 CMS      NSS      MAINT
Ready; T=0.01/0.01 08:16:14
```

The first segment is in class A status. That means that it is available for use. This is the segment in use by the other virtual machines. The second segment is in class S status. This means that it is a shell that is ready and waiting for something to be stored in it.

Type: ipl 190 clear parm savesys cms

This command will reIPL the CMS system and causes it to be saved into the segment named CMS. Once this is complete, you will receive a message like the following:

Named Save System (NSS) CMS was successfully saved in fileid 0030.

Type: query nss name cms

Now, you will notice, that the original class A segment is class P (pending purge), and the new class S segment is class A (active)..

4 Directory Management Exercises

Managing the CP directory is one of the most important tasks for a VM system administrator. The contents of the directory is the key for everything that happens on an individual VM system. Making a mistake with the directory can destroy information, or at worst, damage VM itself. At its most basic, the CP directory is a text file, which is compiled into an object directory using the CP command, DIRECTXA. When VM is first installed, this is how the directory is supplied. On a simple system, such as the basic install system, this is not hard to manage. However, as virtual machines are defined and more disk space is added to the system, this gets more and more difficult to handle. It's easy to make a mistake and cause a serious problem.

Two major software packages are available to assist in the management of the CP directory, DIRMAINT from IBM and VM:Secure from Computer Associates. DIRMAINT is available as a priced feature of z/VM and is included in the basic z/VM system installation. In order to be used, DIRMAINT must be "enabled". This can be done temporarily using the SET PRODUCT command. However, that goes away when CP is recycled. To permanently enable a product, it must be set in the SYSTEM CONFIG file, and VMSES control files must be updated.

The exercises in this section will demonstrate how to configure DIRMAINT. You will also work with the Programmable Operator (PROP), which is an excellent tool for managing virtual machines. There will also be an exercise using TRACK, a tool for doing problem determination on a virtual machine.

Note: If you are not currently logged onto MAINT, please do so before continuing with the exercise.

4.1 Enable DIRMAINT Feature

Type: service dirm enable

The SERVICE EXEC is part of the VMSES component of z/VM. It may be used to install corrective service from tape or electronic envelope, as well as Recommended Service Upgrade (RSU) tapes from tape, or electronic envelope. In this case we are using SERVICE to enable a product. SERVICE will update the product record in the SYSTEM CONFIG file for DIRMAINT, as well as update key VMSES inventory files.

The system will respond to this command by producing many messages. Keep clearing your console until you receive the CMS Ready prompt.

Type: query product

Use this command to verify that DIRMAINT is, in fact, enabled on your system. If "6VMDIR10" shows up as "disabled", then determine what went wrong in the step above.

4.2 Activating DIRMAINT Virtual Machine

Type: xedit user direct a
Type: /USER DIRMAINT

This command will move you to the place in the file where the DIRMAINT virtual machine is defined. The definition begins with the "USER" record and ends with the next "USER" record. Everything in between defines the environment in which the virtual machine named DIRMAINT will be created when it's logged onto CP. This is true of all virtual machines contained in the directory.

The third word on the "USER" record is the password for the virtual machine. Currently, it should say **NOLOG**. This is a special password for CP. It instructs the system to create the virtual machine, but treat it as if it does not exist. So, while the machine exists and can hold minidisks, it cannot be logged onto CP. This is a useful feature for creating "holding" machines such as $PAGE$. It also allows you to create a virtual machine in advance, but not activate it until you are ready to use it. There is a CP command, LOCK, which allows you to keep a virtual machine from logging onto CP, temporarily. However, this lock does not carry over a recycle of CP. Also, another user with the appropriate privilege can override it. The NOLOG password can only be overridden by someone with access to the directory.

Action: Move cursor to the "N" in "NOLOG".
Type: VMLAB (over top of "NOLOG")
Action: Hit Enter to move cursor to command line.
Type: file
Type: directxa user direct

You have now changed the password of the DIRMAINT virtual machine to VMLAB. In so doing, you have also given it the ability to be logged onto the system.

4.3 Updating Config Files

Type: disc
Action: Hit Enter twice, you should see the z/VM logo.

Each IBM product that runs on z/VM is installed on an account separate from MAINT. The name of each account usually contains the product number. In the case of products that come with z/VM as a feature, the name of the account matches VMSES name. For DIRMAINT, that is 6VMDIR10.

Type: 6VMDIR10 (in Userid field of logo)
6VMDIR10 (in Password field of logo)
Action: Hit Enter
Action: Hit Enter (again, to execute PROFILE EXEC)

Note: A base install system comes with the product accounts defined so that the password matches the name. This is for convenience when first setting up the system. **Never run in production like this.** It can be a serious security problem. Most product accounts do not have special privileges. However, they do have access to minidisks being used by production virtual machines, such as DIRMAINT. A hacker could do a lot of damage to them. **Always change the default passwords before making the system available to the world.**

Type: access 492 u
Type: dir2prod access_new 6vmdir10 dirm

The first step in configuring DIRMAINT is to update the main file, CONFIG DATADVH. Instead of directly modifying the file that comes from IBM, we will create an override file with just the statements we want to change.

Type: xedit config99 datadvh L
Type: input
Type: RUNMODE=OPERATIONAL
Type: ONLINE=IMMED

Action: Hit Enter to move cursor to the Command Line
Type: file

These two changes to the CONFIG DATADVH file will tell DIRMAINT that it is running in "operational", rather than "testing" mode, and to put any changes in the directory online as soon as they are submitted.

Another file that must be updated is the AUTHFOR CONTROL file. This identifies the accounts that are allowed to issue privileged DIRMAINT commands. This list should be limited only to those accounts controlled by system administrators, such as MAINT.

Type: x authfor control j
Type: input
Type: ALL MAINT * 140A ADGHOPS
Action: move cursor to the beginning of the next empty line
Type: ALL MAINT * 150A ADGHOPS
Action: Hit Enter twice, to get out of input mode and to go to the Command Line
Type: file

The final file that must be updated is the EXTENT CONTROL file. There are other, less important, files, but they are not critical to bringing up DIRMAINT for the first time. The EXTENT CONTROL file defines several important pieces of information for DIRMAINT. First, it defines what disk volumes are to be used for allocating minidisks. If the VM system is sharing volumes with other systems, then you should only include in EXTENT CONTROL those volumes that you want VM to use. Trying to allocate on any volumes not in EXTENT CONTROL will cause an error.

The file also includes a section that defines “groups”. A “group” is a collection of VM volumes that are to be used for a specific purpose, such as minidisks for Linux machines, for example. Once you create a “group”, you can allocate minidisks to that group name. DIRMAINT will look through the volumes on the list and allocate the minidisk in the first available space. Groups are a good way of collecting categories of minidisks on particular volumes.

The other important section in the EXENT CONTROL file is the “exclude” section. By default, DIRMAINT will not allow you to allocate a minidisk over an area of a volume that already contains a minidisk. This is a VERY good feature, which will protect you from mistakes that can ruin your system. However, there are situations where you may want to overlay one minidisk over another. For example, the MAINT 123 disk is normally a full pack volume that is defined over your SYSRES volume (as installed with the base system). If you follow DIRMAINT’s default, this full pack volume will prevent you from allocate any minidisks on the SYSRES volume. In fact, the 123 disk exists only to give you the ability to manipulate the volume when necessary. You do not want it to block other minidisks from being created. Therefore, you want this disk to be in the exclude list. DIRMAINT will then ignore its existence, except for including it on reports.

We’ll start by editing the sample file supplied by DIRMAINT:

Type: x extent control j
Action: Use PF8 to page down to line that starts “*RegionId”
Action: Move cursor to prefix area next to line
Type: i2 (Hit Enter)

Two blank lines should appear just below the “*RegionId” line.

Action: Move cursor to first blank line, just created.
Type: LINUX01 610LAB 967 1499 3390-3
Action: Move to next blank line
Type: LINUX02 610LIN 001 END 3390-3

Your screen should look like the following:

```
:REGIONS.
  *RegionId  VolSer    RegStart      RegEnd  Dev-Type  Comments
  LINUX01    610LAB    967           1499    3390-3
  LINUX02    610LIN    001           END     3390-3
:END.
```

Action: Move cursor prefix area next to line that beings with “*GroupName”, approximately 2 lines below
Type: i (Hit Enter)
Action: Move cursor to the blank line just created
Type: GRPLNX LINUX01

Action: Move cursor to prefix area next to line beginning with "* UserId"
Type: i2 (Hit Enter)
Action: Move cursor to first blank line created
Type: MAINT 012*
Action: Move cursor to second blank line
Type: SYSDUMP1 012*

These two sections of the file should now look like the following:

```
:GROUPS.
  *GroupName RegionList
  GRPLNX LINUX01
:END.
:EXCLUDE.
  *UserId Address
  MAINT 012*
  SYSDUMP1 012*
:END.
```

Action: Move to the bottom of the file and locate the line that starts with "*DASDType".
Action: Insert a new blank line following this one. Move cursor to beginning of new line.
Type: 3390-3 3339

The section should now look like the following:

```
00057 :DEFAULTS.
00058   * IBM supplied defaults are contained in the DEFAULTS DATADVH file.
00059   * The following are customer overrides and supplements.
00060   *
00061   *DASDType Max-Size
00062   3390-3 3339
00063 :END.
```

Action: Hit Enter key to move to Command Line
Type: file

4.4 Bringing up DIRMAINT machine

Updating the necessary config files is now complete. There is one last step to take before DIRMAINT can be started. It needs a copy of the current directory file. This will serve as the basis for DIRMAINT to start maintaining the directory itself. Once DIRMAINT is activated, the USER DIRECT file will no longer be valid.

Type: link maint 191 1191 rr
Action: Since this system has no security manager, such as RACF or VM:SECURE, you will receive a prompt for the READ password for the minidisk.
Type: READ (Hit Enter)
Type: access 1191 x

Type: copy user direct x user input j
Type: release x (detach

This sequence of commands linked MAINT's 191 disk in read/only mode and, then, copied the current USER DIRECT file to one of DIRMAINT's disk. The file type was changed to INPUT because DIRMAINT will specifically look for a file named USER INPUT when building its directory for the first time.

Until this point, all the file changes have taken place on DIRMAINT's "test level" disks. These are disks that can be used to test out changes before they are made permanent. Before activating DIRMAINT, these new files must be moved to the "production level" disks. We will perform this task back on the MAINT userid.

Type: logoff
Action: Hit Enter twice to get back to the z/VM logo.

Action: Log back onto MAINT, password MAINT
Type: b
Action: Hit Enter.

Type: put2prod dirm
Action: Hit Enter.

The VMSES tool put2prod will move the DIRMAINT components from their test locations to the production runtime locations. Many messages will scroll by as the process completes.

It's now time to start up DIRMAINT and let it take control of the directory.

Type: disc
Action: Hit Enter twice to get back to the z/VM logo.

Type: DIRMAINT (in the Userid field)
VMLAB (in the password field)
Action: Hit Enter (to logon, and to execute the PROFILE EXEC)

You are now logged onto the DIRMAINT virtual machine. Normally, you should run this machine in "disconnected" mode rather than logged on. However, bringing it up for the first time is a special case that requires direct monitoring.

You will see a number of messages on the screen indicating that DIRMAINT has started. However, because you are logged on, rather than disconnected, DIRMAINT will not start its main program. You will have to do that manually.

Type: dvhbegin (Hit Enter)

At this point, you will start to see a very large number of messages appearing on the screen. DIRMAINT is known for being somewhat verbose. Use the Clear key to keep clearing the screen. Look for any messages that indicate a problem, or that DIRMAINT could not initialize the directory. The last messages you receive should look like the following:

```
DVHREQ2289I Your BACKUP request for DIRMAINT at * has completed; with RC
DVHREQ2289I = 0.
DIRMAINT VMLAB... - 2010/01/14; T=0.12/0.12 08:55:19
DVHREQ2290I Request is: ELINK CLEAN ALL
DVHREQ2288I Your ELINK request for DIRMAINT at * has been accepted.
DVHREQ2289I Your ELINK request for DIRMAINT at * has completed; with
DVHREQ2289I RC = 0.
DIRMAINT VMLAB... - 2010/01/14; T=0.02/0.02 08:55:20
DVHREQ2290I Request is: CMS EXEC DVHOURLY
DVHREQ2288I Your CMS request for DIRMAINT at * has been accepted.
DVHRLY3886I Hourly processing started; with 0 log
DVHRLY3886I files.
DVHREQ2289I Your CMS request for DIRMAINT at * has completed; with RC
DVHREQ2289I = 0.
DIRMAINT VMLAB... - 2010/01/14; T=0.02/0.03 08:55:21
DVHWAI2140I Waiting for work on 10/01/14 at 08:55:21.
```

Type: #cp set run on
Type: #cp disc
Action: Hit Enter twice to get back to VM logo

These two commands enable you to disconnect from the DIRMAINT machine and leave it running in "disconnected" mode.

4.5 Testing DIRMAINT

Action: Logon to MAINT using procedure in previous exercises
Type: b
Action: Hit Enter.

When you log onto a virtual machine that is in disconnect mode, check the lower right hand corner. If it says "CP READ", then you must use the BEGIN command to restart CMS where it left off.

Type: dirm needpass no
Type: maint (in response to prompt for password)

You should receive a series of messages similar to the following:

```
DVHREQ2288I Your USEROPTN request for MAINT at * has been accepted.
DVHBIU3450I The source for directory entry MAINT has been updated.
DVHBIU3456I Object directory update is not required for this source
DVHBIU3456I update.
DVHREQ2289I Your USEROPTN request for MAINT at * has completed; with
```

```
DVHREQ2289I RC = 0.
```

Issuing the DIRM NEEDPASS NO command verifies that DIRMAINT is up and running. It also sets MAINT so that you do not have to enter the password every time you issue a DIRMAINT command. If you receive error messages instead, then you need to back and check to see if DIRMAINT actually came up successfully.

4.5.1 Using TRACK

The VM community has a long history of developing tools to help with system administration and problem resolution tasks. One of the best tools is named TRACK. Originally developed by Serge Goldstein of Princeton University, it has been maintained by several people over the years. The current version of TRACK is maintained by Jim Vincent of Nationwide and is available at ***http://www2.marist.edu/track***.

TRACK has many useful features. It allows you, from a privileged virtual machine, to peer into the operation of any other virtual machine on the system. You can see the open console, virtual devices, PSW, registers, running status and many other pieces of information useful in doing problem determination. TRACK is able to do this without any interruption to the virtual machine being viewed.

Type: attach 6202 *
Type: access 6202 x/x
Type: vmarc unpk trackv52 vmarc x
Type: rename trackv52 module a track = =

The VMARC program is another example of a tool written by a VM system programmer that has become widely used throughout the VM community. It allows you to pack groups of files for transport over a network. It is available off of IBM's VM download web page, ***http://www.vm.ibm.com/download***.

Both the VMARC program and the TRACK package are preinstalled for you for the lab. By unpacking the TRACK package, you are installing it on your A-disk. It can also be moved to a public tool disk.

Type: track dirmaint cons

This command starts TRACK and puts you in a display of the open console for the DIRMAINT machine. You can page through the current spool buffer for the console. It will not show you the complete console. However, it will show you the last 20-40 messages, which is usually enough to determine if a problem is occurring.

Action: Hit PF9 key.

This screen shows you details about the DIRMAINT machine, including its privilege classes, amount of virtual storage and when it was logged onto the system. Each time you hit the Enter Key, the screen will refresh with new information.

Action: Hit PF2 key.

This puts TRACK into AUTO mode. It will refresh the screen every 2 seconds. This is useful if you want to monitor the machine for a period of time.

Action: Hit PF3 key.

This turns off AUTO mode.

Action: Hit PF11 key.

This displays the list of virtual devices attached to the virtual machine. The list also includes the types of devices, as well as their addresses. If any have outstanding I/O queued, the address will be highlighted. This is useful if you are trying to determine why a machine may be "hung".

Action: Hit PF3 key to exit TRACK.

5 Service Machine Automation Exercises

All VM systems, regardless of their purpose, will have some service virtual machines (SVM) that run in disconnected mode. Some machines, such as DIRMAINT, may provide services directly for VM. Others, however, may provide network services, such as TCPIP, or FTPSERVE. The third type of SVM is guest systems, such as Linux, which are running in disconnected mode. A small VM system could have 10 or more SVMs, a large system can have hundreds or thousands. Managing all of these machines is more work than a system administrator can do alone. The machines have to be constantly monitored for problems. Logs of activity have to be kept. The machines have to be recovered if something goes wrong, operators have to be notified. It can be a lot of work.

Several commercial products are available to assist with this work. The most common is VM:Operator from Computer Associates. It is extremely useful, especially for large VM systems. Various Linux vendors are also developing new products, specifically, to assist in managing a large number of Linux images on VM.

It's not necessary, however, to purchase a commercial product to build automatic SVM monitoring tools. A tool is provided by VM, which can be used to start building this automation. The tool is called the Programmable Operator (PROP). It is similar, in concept, to a SYSLOG daemon on a Unix/Linux system. The main part of PROP is provided as part of CMS. This portion allows for the creation of a routing table. The PROP function takes incoming messages and searches the routing table for matches. When a match occurs, then an action can be taken based on the match.

The other part of the PROP function is part of CP, the Secondary Console Facility (SCIF). This facility allows you to define a "secondary" virtual machine for a SVM. This is defined by putting the name of the "secondary" machine on the CONSOLE record of the CP Directory entry for the SVM. When the SVM is running connected, on a terminal or logical session, the SCIF definition has no effect. However, when the SVM is running in disconnected mode, all messages that would normally be displayed on the console, will be directed to the "secondary" machine instead. If the "secondary" machine is running the PROP function, then it can receive the incoming messages and take action depending on the contents of the routing table (RTABLE). Normally, at least one "secondary" machine is defined as the CP system operator. Details on how PROP and SCIF work together can be found in the *CMS Planning and Administration Guide (SC24-6042)*.

In this exercise, you will create a "secondary" machine to monitor SVMs, and give it a simple PROP routing table that will handle DIRMAINT. A sample PROP RTABLE is provided on the MAINT 490 disk. For the purposes of the lab, sample action routines have also been created.

5.1 Create "Secondary" Operator Machine

The first step is to create the "secondary" machine that will handle the PROP functions for the SVMs, such as DIRMAINT. If you are not already logged onto MAINT, do so now.

Type: access 6202 x/x
Type: dirm add secoper

In order to save time, a "secondary" machine has been defined for you. Issuing the DIRMAINT command has now created it in the CP directory. If you would like to see what the definition of the machine looks like, use XEDIT to edit SECOPER DIRECT. After issuing this command, you should see messages similar to the following:

```
DVHDRC3428I Changes made to directory entry SECOPER have just been
DVHDRC3428I placed online.
DVHREQ2289I Your ADD request for SECOPER at * has completed; with RC =
DVHREQ2289I 0.
```

Type: link secoper 191 9191 mr
Type: format 9191 z
Type: yes (in response to continue question)
Type: sec191 (in response to label question)
Type: copy prop rtable x = = z
Type: copy profile execprop x = exec z
Type: copy archlog exec x = = z
Type: copy stopem exec x = = z
Type: copy sendem exec x = = z
Type: copy logemoff exec x = = z
Type: release x (det

This series of commands copied the predefined sample RTABLE file to the new "secondary" machine's 191 disk. It also copied a PROFILE EXEC and some sample action routines to the new disk. These routines are written in standard REXX and can be inspected using XEDIT.

Type: xedit profile exec z

You are now editing the PROFILE EXEC that will be executed whenever SECOPER is autologged.

Action: Move cursor to beginning of line 3.
Type: 'CP SPOOL CONS * START'
Action: Hit Enter Key
Type: file
Type: xedit prop rtable z
Action: Move cursor to 5th line in file, which should start with "LGLOPR"

This record defines the "logical operator" for the PROP function. The "logical operator" should be a virtual machine that is normally running in connected mode, such as a machine monitored by a human operator. The machine does not have to have any special privileges to act as the "logical operator". This machine receives any messages that pass through this routing table, but do not match any of the filters. The "logical operator" is essentially the machine of last resort for any message passing through the "secondary" operator.

The two parameters on the “LGLOPR” record are the name of the “logical operator” machine and the Identifier of the system it resides upon. The later parameter is necessary in case you have multiple VM systems networked together.

Action: Move cursor over to the word “LINLABnn”.
Type: VMLAB (over the full word)

Action: Move cursor down to line 22 and put it in the prefix area
Type: i3 (hit Enter Key)

The position of each character in an RTABLE entry is important. The dashed lines are provided to give you a guide as to where each field begins and ends. The fields are as follows:

TEXT - Text filter for message. Wildcard characters are normally used.
SCOL - Position in incoming message where match should start
ECOL - Position in incoming message where to stop match
TYPE - Type of incoming message. Messages are broken up into several types. If you are targeting a specific kind of message, TYPE can help.
USER - Name of virtual machine which generated the message
NODE - Identifier of System which virtual machine is running on
ACTN - Name of action routine
PARM - Optional parameter to action routine

Most of the fields are optional. If nothing is specified in the field, than all messages are passed through this filter. Using very specific filters can be both a positive and a negative. On the positive side, it can make the RTABLE searches more efficient. For a “secondary” operator that controls hundreds of SVMs, this can be very important as thousands of messages will pass through the search. However, very specific filters are more difficult to define correctly and can break easily if a message number or text changes. Finding a balance between the two is one of the important skills to learn when creating PROP RTABLEs.

Action: Move cursor to first blank line inserted in file
Action: Enter the following 3 lines into the blanks lines. Use dashed line in the file as guide for placement of fields. The first column is the TEXT field, second column is the SCOL value. The third is the USER field, and the fourth is the NODE field. The fifth is the ACTN field, and the last is the PARM field.
Note: The columns MUST line up properly or you will receive an error when activating the RTABLE.

```
*------------------------------------ --- --- -- ---------------- --------------- ---------------- -------------
$DVHSHU2198A$                        1       DIRMAINT  VMLAB      DMSPOR          LGLOPR
$DVH$2008$                           1       DIRMAINT  VMLAB      DMSPOR          LGLOPR
                                             DIRMAINT  VMLAB
```

The first two records will capture the logoff message and the start up message and route them to the logical operator. The third record will capture all the other messages from DIRMAINT and log them. This will prevent all the DIRMAINT messages from flooding the logical operator.

The section should look like the following:

```
00017 *----------------------- --- --- -- -------- -------- -------- --------
00018 *T                      S   E   T  U        N        A        P
00019 *E                      C   C   Y  S        O        C        A
00020 *X                      O   O   P  E        D        T        R
00021 *T                      L   L   E  R        E        N        M
00022 *----------------------- --- --- -- -------- -------- -------- --------
00023 $DVHSHU2198A$           1          DIRMAINT VMLAB    DMSPOR   LGLOPR
00024 $DVH$2008$              1          DIRMAINT VMLAB    DMSPOR   LGLOPR
00025                                    DIRMAINT VMLAB
```

Action: Hit Enter Key to go to command line
Type: file
Type: release z (det
Type: xautolog secoper
Type: track secoper cons

You should be able to see, from the SECOPER console, if it is running or has gotten an error. If there is a problem, then it is probably with the RTABLE. Do the following ONLY if there is a problem:

Type: force secoper
Type: link secoper 191 9191 mr
Type: access 9191 z
Type: xedit prop rtable z

Review the changes to the RTABLE and see if you can identify the problem.

5.1.1 Setting “Secondary” Machine Name

Now that the “secondary” operator has been created and is running, its time to define a SVM so that it will make use of it. The “secondary” operator for a SVM is defined on the CONSOLE record of the directory.

Type: dirm for secoper authscif dirmaint
Type: dirm for dirmaint secuser secoper

The first DIRMAINT command authorizes SECOPER to act as a secondary user for DIRMAINT. The second command actually defines it. The messages you receive should look like the following:

```
dirm for secoper authscif dirmaint
DVHXMT1191I Your AUTHSCIF request has been sent for processing.
Ready; T=0.08/0.09 17:00:35
DVHREQ2288I Your AUTHSCIF request for SECOPER at * has been accepted.
DVHREQ2289I Your AUTHSCIF request for SECOPER at * has completed; with
DVHREQ2289I RC = 0.
dirm for dirmaint secuser secoper
DVHXMT1191I Your SECUSER request has been sent for processing.
Ready; T=0.08/0.09 17:01:03
DVHREQ2288I Your SECUSER request for DIRMAINT at * has been accepted.
DVHBIU3424I The source for directory entry DIRMAINT 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 DIRMAINT have been placed
DVHBIU3428I online.
DVHREQ2289I Your SECUSER request for DIRMAINT at * has completed; with
DVHREQ2289I RC = 0.
```

The change will not take effect until the DIRMAINT virtual machine is logged off. You can also use the CP command, SET SECUSER to define the "secondary" operator dynamically. This, however, is only in effect until the virtual machine logs off.

Type: dirm shutdown
Type: q dirmaint (verify that machine has logged off)
Type: xautolog dirmaint
Type: query secuser dirmaint

When DIRMAINT has finished initializing, you should receive the following message on MAINT:

DVHPRO2008I ROLE = DIRMAINT

You now have a functioning PROP SVM that can be used to monitor and automate DIRMAINT. It can also be extended to monitor and control all the SVMs and guest systems on your system.

Appendix A: Sample CPFMTXA Listing

The following is a sample of how to use CPFMTXA to format and allocate a new spool area.

```
cpfmtxa 202
ENTER FORMAT, ALLOCATE, LABEL, OR QUIT:
format
ENTER THE PAGE RANGE TO BE FORMATTED ON DISK 0202 OR QUIT:
0 end
ENTER THE VOLUME LABEL FOR DISK 0202:
pag001
CPFMTXA:
FORMAT WILL ERASE PAGES 000000000-000002499 ON DISK 0202
DO YOU WANT TO CONTINUE? (YES | NO)
yes
HCPCCF6209I INVOKING ICKDSF.
ICK030E DEFINE INPUT  DEVICE: FN FT FM, "CONSOLE", OR "READER"
CONSOLE
ICK031E DEFINE OUTPUT DEVICE: FN FT FM, "CONSOLE", OR "PRINTER"
CONSOLE
ICKDSF - CMS/XA/ESA DEVICE SUPPORT FACILITIES 17.0                TIME: 14:06:53
01/20/09     PAGE   1

ENTER INPUT COMMAND:
  CPVOL FMT MODE(ESA) UNIT(0202) VOLID(PAG001) NOVFY -
ENTER INPUT COMMAND:
 RANGE(0,2499)
ICK00700I DEVICE INFORMATION FOR 0202 IS CURRENTLY AS FOLLOWS:
          PHYSICAL DEVICE = 9336-20
          STORAGE CONTROLLER = 6310
          STORAGE CONTROL DESCRIPTOR = 80
          DEVICE DESCRIPTOR = 10
ICK03020I CPVOL WILL PROCESS 0202 FOR VM/ESA MODE
ICK03090I VOLUME SERIAL = ......
ICK03011I PAGE RANGE TO BE FORMATTED IS 0 - 2499
ICK003D REPLY U TO ALTER VOLUME 0202 CONTENTS, ELSE T
U
ICK03000I CPVOL REPORT FOR 0202 FOLLOWS:

          FORMATTING OF PAGE 0 STARTED AT: 14:06:53
          FORMATTING OF PAGE 2499 ENDED AT: 14:06:53

          VOLUME SERIAL NUMBER IS NOW = PAG001

          PAGE ALLOCATION CURRENTLY IS AS FOLLOWS:
          TYPE     START      END        TOTAL
          ----     -----      ---        -----
          PERM     4          2499       2496

ICK00001I FUNCTION COMPLETED, HIGHEST CONDITION CODE WAS 0
          14:06:53    01/20/09

ENTER INPUT COMMAND:
 END
```

```
ICK00002I ICKDSF PROCESSING COMPLETE. MAXIMUM CONDITION CODE WAS 0
ENTER ALLOCATION DATA
TYPE PAGES
.................
page 0004 end
end
HCPCCF6209I INVOKING ICKDSF.
ICK030E DEFINE INPUT  DEVICE: FN FT FM, "CONSOLE", OR "READER"
CONSOLE
ICK031E DEFINE OUTPUT DEVICE: FN FT FM, "CONSOLE", OR "PRINTER"
CONSOLE
ICKDSF - CMS/XA/ESA DEVICE SUPPORT FACILITIES 17.0               TIME: 14:07:34
01/20/09     PAGE   1

ENTER INPUT COMMAND:
  CPVOL ALLOC MODE(ESA) UNIT(0202) VFY(PAG001) -
ENTER INPUT COMMAND:
        TYPE((PAGE,0004,2499))
ICK00700I DEVICE INFORMATION FOR 0202 IS CURRENTLY AS FOLLOWS:
          PHYSICAL DEVICE = 9336-20
          STORAGE CONTROLLER = 6310
          STORAGE CONTROL DESCRIPTOR = 80
          DEVICE DESCRIPTOR = 10
ICK03020I CPVOL WILL PROCESS 0202 FOR VM/ESA MODE
ICK03090I VOLUME SERIAL = PAG001
ICK003D REPLY U TO ALTER VOLUME 0202 CONTENTS, ELSE T
U
ICK03000I CPVOL REPORT FOR 0202 FOLLOWS:

          PAGE ALLOCATION CURRENTLY IS AS FOLLOWS:
          TYPE     START      END        TOTAL
          ----     -----      ---        -----
          PAGE     4          2499       2496

ICK00001I FUNCTION COMPLETED, HIGHEST CONDITION CODE WAS 0
          14:07:34    01/20/09

ENTER INPUT COMMAND:
 END

ICK00002I ICKDSF PROCESSING COMPLETE. MAXIMUM CONDITION CODE WAS 0
Ready; T=0.01/0.01 14:07:36
```

Appendix B: Sample Process for Backing Up PARM Disk

```
q cpdisks
Label  Userid   Vdev Mode Stat Vol-ID Rdev Type   StartLoc     EndLoc
MNTCF1 MAINT    0CF1  A   R/O  610LAB 0200 CKD          39        158
MNTCF2 MAINT    0CF2  B   R/O  610LAB 0200 CKD         159        278
MNTCF3 MAINT    0CF3  C   R/O  610LAB 0200 CKD         279        398
Ready; T=0.01/0.01 10:14:15
cprelease b
CPRELEASE request for disk B scheduled.
HCPZAC6730I CPRELEASE request for disk B completed.
Ready; T=0.01/0.01 10:14:19
acc cf1 x
DMSACP723I X (CF1) R/O
Ready; T=0.01/0.01 10:14:26
link * cf2 cf2 w
Ready; T=0.01/0.01 10:14:31
access cf2 z
Ready; T=0.01/0.01 10:14:43
copy * * x = = z (olddate replace
Ready; T=0.01/0.06 10:14:57
listfile * * z (d
FILENAME FILETYPE FM FORMAT LRECL       RECS     BLOCKS   DATE     TIME
CPLOAD   MODULE   Z1 V      65535        149       2356 11/18/09 15:22:16
CPLOLD   MODULE   Z1 V      65535        149       2355  9/11/09 16:51:48
DDR      MODULE   Z2 V      65535          7         70  5/08/09  9:43:43
DEFAULT  LOGO     Z1 F         78         15          1  8/17/01 15:24:42
ICKSADSF MODULE   Z2 V      65535         17        256  2/04/09 11:31:53
INPTAREA SAMPLE   Z1 F         78          6          1  4/27/93 18:27:26
IPL      DDRXA    Z2 F         80       4238         83  5/08/09  9:51:04
IPL      ICKDSF   Z2 F         80      14546        285  2/04/09 11:21:35
IPL      SALIPL   Z2 F         80        973         20  5/08/09  9:51:04
LDEV     LOGO     Z1 F         78         15          1  8/17/01 15:24:06
LOCAL    LOGO     Z1 F         78         15          1  8/17/01 15:23:44
LOGO     CONFIG   Z1 V         69         63          1  4/27/93 15:41:58
MINIMUM  LOGO     Z1 F         78         16          1  8/17/01 15:26:04
ONMESS   SAMPLE   Z1 F         78          1          1 10/18/00 21:47:32
PRINTSEP LOGO     Z1 F         49         16          1  8/17/01 15:25:49
SALIPL   MODULE   Z2 V      59560          3         15  5/08/09  9:43:49
SNA      LOGO     Z1 F         78         15          1  8/17/01 15:24:18
SYSTEM   CONFIG   Z1 F         80        248          5  1/13/10 15:14:36
Ready; T=0.01/0.01 10:15:10
rel z (det
DASD 0CF2 DETACHED
Ready; T=0.01/0.01 10:16:42
cpaccess maint cf2 b
CPACCESS request for mode B scheduled.
Ready; T=0.01/0.01 10:16:50
HCPZAC6732I CPACCESS request for MAINT's 0CF2 in mode B completed.
q cpdisks
Label  Userid   Vdev Mode Stat Vol-ID Rdev Type   StartLoc     EndLoc
MNTCF1 MAINT    0CF1  A   R/O  610LAB 0200 CKD          39        158
MNTCF2 MAINT    0CF2  B   R/O  610LAB 0200 CKD         159        278
MNTCF3 MAINT    0CF3  C   R/O  610LAB 0200 CKD         279        398
Ready; T=0.01/0.01 10:16:54
```

Appendix C: Sample Procedure for Updating Primary PARM Disk

```
q cpdisks
Label  Userid   Vdev Mode Stat Vol-ID Rdev Type   StartLoc     EndLoc
MNTCF1 MAINT    0CF1  A   R/O  610LAB 0200 CKD          39        158
MNTCF2 MAINT    0CF2  B   R/O  610LAB 0200 CKD         159        278
MNTCF3 MAINT    0CF3  C   R/O  610LAB 0200 CKD         279        398
Ready; T=0.01/0.01 10:23:17
cprelease a
CPRELEASE request for disk A scheduled.
HCPZAC6730I CPRELEASE request for disk A completed.
Ready; T=0.01/0.01 10:23:21
link * cf1 cf1 w
Ready; T=0.01/0.01 10:23:27
acc cf1 x
DMSACC724I CF1 replaces X (CF1)
Ready; T=0.01/0.01 10:23:31
erase system confold x
DMSERS002E File SYSTEM CONFOLD X not found
Ready(00028); T=0.01/0.01 10:23:38
rename system config x = confold x
Ready; T=0.01/0.01 10:23:48
copy system config a = = x (oldd
Ready; T=0.01/0.01 10:23:56
rel x (det
DASD 0CF1 DETACHED
Ready; T=0.01/0.01 10:23:58
cpaccess maint cf1 a sr
CPACCESS request for mode A scheduled.
Ready; T=0.01/0.01 10:24:06
HCPZAC6732I CPACCESS request for MAINT's 0CF1 in mode A completed.
```

Appendix D: Sample Timezone Change

```
q timezone
Zone  Direction   Offset   Status
UTC     ----     00.00.00  Inactive
GMT     ----     00.00.00  Inactive
EDT     West     04.00.00  Inactive
EST     West     05.00.00  Active
CDT     West     05.00.00  Inactive
CST     West     06.00.00  Inactive
MDT     West     06.00.00  Inactive
MST     West     07.00.00  Inactive
PDT     West     07.00.00  Inactive
PST     West     08.00.00  Inactive
Ready; T=0.01/0.01 10:40:05
set timezone pdt
08:40:31 HCPTZN6759I The time zone has changed to PDT.
Ready; T=0.01/0.01 08:40:31
query timezone
Zone  Direction   Offset   Status
UTC     ----     00.00.00  Inactive
GMT     ----     00.00.00  Inactive
EDT     West     04.00.00  Inactive
EST     West     05.00.00  Inactive
CDT     West     05.00.00  Inactive
CST     West     06.00.00  Inactive
MDT     West     06.00.00  Inactive
MST     West     07.00.00  Inactive
PDT     West     07.00.00  Active
PST     West     08.00.00  Inactive
Ready; T=0.01/0.01 08:40:40
x system config a
00050     Timezone_boundary on 2007-03-11 at 02:00:00 to EDT
00051     Timezone_boundary on 2007-11-04 at 02:00:00 to EST
00052
00053     Timezone_boundary on 2008-03-09 at 02:00:00 to EDT
00054     Timezone_boundary on 2008-11-02 at 02:00:00 to EST
00055
00056     Timezone_boundary on 2009-03-08 at 02:00:00 to EDT
00057     Timezone_boundary on 2009-11-01 at 02:00:00 to EST
00058
00059     Timezone_boundary on 2010-03-14 at 02:00:00 to PDT
00060     Timezone_boundary on 2010-11-07 at 02:00:00 to PST
00061
00062     Timezone_boundary on 2011-03-13 at 02:00:00 to EDT
00063     Timezone_boundary on 2011-11-06 at 02:00:00 to EST

save
```

Appendix E: Sample Product Enabling

```
service dirm enable

VMFSRV2760I SERVICE processing started
VMFINS2767I Reading VMFINS DEFAULTS B for additional options
VMFINS2760I VMFINS processing started
VMFINS2603I Processing product :PPF 6VMDIR10 DIRM :PRODID
            6VMDIR10%DIRM
VMFINS2603I Enabling product 6VMDIR10%DIRM
VMFINS2771I The CP SET PRODUCT command completed successfully for
            product 6VMDIR10
VMFINS2772I File 6VMDIR10 PRODSYS created on your A-disk contains the
            system configuration PRODUCT statement for product
            6VMDIR10
VMFINS2603I Product enabled in VMSES/E, CP processing required
VMFINS2760I VMFINS processing completed successfully
HCPZAC6730I CPRELEASE request for disk A completed.
DASD 0CF1 DETACHED
HCPZAC6732I CPACCESS request for MAINT's 0CF1 in mode A completed.
VMFSRV2774I The system configuration PRODUCT statement for product
            6VMDIR10%DIRM was successfully copied from file 6VMDIR10
            PRODSYS to system configuration file SYSTEM CONFIG on
            MAINT CF1
VMFSUI2760I VMFSUFIN processing started
VMFSUI2760I VMFSUFIN processing started for product 6VMDIR10%DIRM
VMFSET2760I VMFSETUP processing started for SERVP2P DIRM
VMFSET2204I Linking 6VMDIR10 2B2 as 2B2 with the link mode MR
VMFSET2204I Linking 6VMDIR10 2B1 as 2B1 with the link mode MR
VMFSET2204I Linking 6VMDIR10 2C4 as 2C4 with the link mode MR
VMFSET2204I Linking 6VMDIR10 2C2 as 2C2 with the link mode MR
VMFSET2204I Linking 6VMDIR10 2D2 as 2D2 with the link mode MR
VMFSET2204I Linking 6VMDIR10 2A6 as 2A6 with the link mode MR
VMFSET2204I Linking 6VMDIR10 2A2 as 2A2 with the link mode MR
VMFSET2204I Linking 6VMDIR10 29E as 29E with the link mode MR
VMFSET2204I Linking 6VMDIR10 492 as 492 with the link mode MR
VMFSET2204I Linking 6VMDIR10 41F as 41F with the link mode MR
VMFSET2204I Linking 6VMDIR10 29D as 29D with the link mode MR
VMFSET2204I Linking 6VMDIR10 502 as 502 with the link mode MR
VMFUTL2205I Minidisk|Directory Assignments:
            String    Mode  Stat  Vdev  Label/Directory
VMFUTL2205I LOCALMOD  E      R/W  2C4   DRM2C4
VMFUTL2205I LOCALSAM  F      R/W  2C2   DRM2C2
VMFUTL2205I APPLY     G      R/W  2A6   DRM2A6
VMFUTL2205I           H      R/W  2A2   DRM2A2
VMFUTL2205I DELTA     I      R/W  2D2   DRM2D2
VMFUTL2205I BUILD0    J      R/W  29E   DRM29E
VMFUTL2205I BUILD1    K      R/W  492   DRM492
VMFUTL2205I BUILD3    L      R/W  41F   DRM41F
VMFUTL2205I BUILD6    M      R/W  29D   DRM29D
VMFUTL2205I BUILD6U   N      R/W  502   DRM502
VMFUTL2205I BASE      O      R/W  2B2   DRM2B2
VMFUTL2205I BASE1     P      R/W  2B1   DRM2B1
VMFUTL2205I --------  A      R/W  191   MNT191
VMFUTL2205I --------  B      R/W  5E5   MNT5E5
```

```
VMFUTL2205I --------  C      R/W  2CC   MNT2CC
VMFUTL2205I --------  D      R/W  51D   MNT51D
VMFUTL2205I --------  S      R/O  190   MNT190
VMFUTL2205I --------  Y/S    R/O  19E   MNT19E
VMFSET2760I VMFSETUP processing completed successfully
VMFBLD2760I VMFBLD processing started
VMFBLD1851I Reading build lists
VMFBLD2182I Identifying new build requirements
VMFBLD2182I No new build requirements identified
VMFBLD2180I There are 0 build requirements remaining
VMFBLD2760I VMFBLD processing completed successfully
VMFBLD2760I VMFBLD processing started
VMFBLD1851I Reading build lists
VMFBLD2182I Identifying new build requirements
VMFBLD2182I No new build requirements identified
VMFBLD2179I There are no build requirements matching your request at
            this time. No objects will be built
VMFBLD2180I There are 0 build requirements remaining
VMFBLD2760I VMFBLD processing completed successfully
VMFSET2760I VMFSETUP processing started for DETACH DIRM
DMSDIO905S WRITE-INHIBIT switch set on drive; notify operator
DMSDIO905S WRITE-INHIBIT switch set on drive; notify operator
DMSDIO905S WRITE-INHIBIT switch set on drive; notify operator
DMSDIO905S WRITE-INHIBIT switch set on drive; notify operator
DMSDIO905S WRITE-INHIBIT switch set on drive; notify operator
DMSDIO905S WRITE-INHIBIT switch set on drive; notify operator
DMSDIO905S WRITE-INHIBIT switch set on drive; notify operator
DMSDIO905S WRITE-INHIBIT switch set on drive; notify operator
DMSDIO905S WRITE-INHIBIT switch set on drive; notify operator
VMFSET2760I VMFSETUP processing completed successfully
VMFSUI2760I VMFSUFIN processing completed successfully for product
            6VMDIR10%DIRM
VMFSUI2760I VMFSUFIN processing completed successfully
VMFSUT2760I VMFSUFTB processing started
VMFSUT2760I VMFSUFTB processing completed successfully
VMFSRV2760I SERVICE processing completed successfully
```

Introduction to z/VM Hands-on Lab

The following is a list of z/VM manuals that are very useful in learning the basic concepts of the VM system and how to manage it.

- z/VM System Operation (SC24-6000)
- REXX/VM User's Guide (SC24-5962)
- XEDIT User's Guide (SC24-5972)
- CMS User's Guide (SC24-6009)
- CMS Planning and Administration (SC24-6042)
- CP Planning and Administration (SC24-6043)
- Directory Maintenance Facility Tailoring and Administration Guide (SC24-6024)
- z/VM General Information (GC24-5991)
- Running Guest Operating Systems (SC24-5997)
- Saved Segments Planning and Administration (SC24-6056)
- z/VM Service Guide (GC24-5993)
- z/VM TCP/IP Planning and Customization (SC24-6019)

Glossary of Terms

Section 1 Terms:

CMS — "Conversational Monitoring System" – a component of z/VM providing file system services. Originally, CMS was an operating system that ran on mainframes of the 1960's. The CP component of VM does not provide file system services such as an editor. So, CMS was run as a guest system to provide these things. Eventually, CMS became a standard on all, early VM systems and became part of the VM product in the early 1970's.

Console — Refers to a device, real or virtual, that receives messages from an operating system and allows the entering of commands to control the system. On old mainframes, the console was a real terminal. On modern zSeries processors, the console runs on the HMC (Host Management Console). In the z/VM environment, the console for each guest is virtual. It can be either connected to a session on a real device (terminal or emulator session), or it can be disconnected, running in the background. Service machines normally run with their console in disconnected state.

CP — "Control Program" – the component of z/VM that provides the virtual environment for each individual user. CP is often referred to as a "hypervisor".

Disconnected Mode — Runs without a physical connection to a terminal or session.

First Level Refers to the operating system in control of the hardware resources, normally the z/VM running native on the processor or in an LPAR partition.

Guest Also “guest operating system” – refers to any operating system running in a virtual machine under CP. It usually refers to systems that could run on the hardware as well as in virtual machine. Virtual machines running CMS or GCS are normally not considered “guests” as they completely depend on VM for their existence.

Hypervisor An operating system that allows other operating systems to exist within it’s environment. CP does this by simulating the hardware environment for each system running under it.

IPL “Initial Program Load” – is the hardware instruction that loads the operating system into memory and starts it running. z/VM simulates this instruction in order to load a guest operating system into virtual memory and start executing it. The IPL command points to a specific disk volume. This volume must contain a “loader program” at cylinder 0. In the case of a first level system, this is normally a full disk volume. For a second level, guest system, it may be a minidisk.

SAL “Standalone Loader” – a “standalone” program is one that can be loaded into memory on a processor and executed. It is normally intended for a specific function and is not a full operating system. The “Standalone Loader” is a program that is run to create the “loader program”.

Second Level Refers to an operating system running as a “guest” of the first level z/VM system, in a virtual machine.

SVM “service virtual machine” – refers to an individual virtual environment used to run programs that provide a service. A service virtual machine is normally run in “disconnected mode”. Examples of service machines are DIRMAINT, and VMSERVS.

Warm Start Refers to the IPL of z/VM that preserves the spool file pointers and other information from the previous shutdown of the system. This is a normal start up. Other types of starts include:

- “force” – attempts to recreate spool file pointers after an aborted shutdown of the system. Use this only if “warm start” fails. If CP is unable to recreate the pointers for one or more spool files, it gives you the option of purging them, but saving the rest.

- “cold” – brings up the system without spool files because it cannot recreate the pointers. Use this only if both “warm” and “force” start fail or if you wish to wipe out the spool file system. This start should not affect NSS, and NLS spool files.

- "clean" – the mode of "last resort". This start wipes all of spool, including NSS and NLS files. It should be used only if all other types of start modes fail.

VM "virtual machine" – refers to the operating system that simulates the functioning of a real machine. It also refers an individual virtual environment created by the operating system.

Section 2 Terms:

Access/Release Two commands used in the CMS environment to gain access to a file system and then remove it.

Attach/Detach Two commands used to obtain access to a real device or minidisk from a virtual machine and to remove it.

Command Classes All CP commands belong to one or more command classes defined by alphabetic characters. Class A is the most powerful and class G the least. Command classes are assigned to virtual machines via the directory.

CPFMTXA A tool provided to assist with the formatting and allocation of CP areas of disk, like PAGE, SPOOL, DRCT.

DASD Stands for "Direct Access Storage Devices". It is usually used to refer to older style mainframe disk devices, such as 3380's and 3390's.

Directory The CP "directory" defines all virtual machines for the system. All aspects of the virtual machines are included in the directory, including passwords, virtual and dedicated devices, and command privilege classes. The compiled version of the directory is stored in a special of disk, DRCT. CP cannot run without an active directory.

ECKD/FBA These are two types of DASD. ECKD refers to "Extended Count Key Disk" and FBA stands for "Fixed Block Architecture". Most modern disk devices for mainframes emulate one of these types.

Full Volume Refers to a minidisk that spans the entire disk volume. Also referred to as a "full pack volume".

Minidisk Refers to a small portion of disk volume that is allocated to a virtual machine.

PAGE Refers to special areas of space on disk volumes allocated and formatted to be used for CP storage paging. A "page" of storage is 4K. Too little page space can adversely affect the performance of the entire system.

PARM — Refers to the “parameter disk” areas. CP has three PARM disks that are used to hold the CP nucleus, system config file, logo files and CP exit modules. The primary PARM disk resides on the System Residence (SYSRES) volume and is used to IPL CP. The second and third PARM disks are used as back ups in case there is something wrong with the primary disk. The PARM disks are located on in areas labeled PARM.

Query — A command that provides information about the system or virtual machine. Both CP and CMS provide a large number of “query” commands.

SPOOL — Refers to special areas of space on disk volumes that contain the spool files of the z/VM system, including NSS, NLS, reader, print and punch files. Like PAGE space, SPOOL is allocated and formatted especially for use with CP.

T-disk — Stands for “temporary disk”. It refers to a temporary minidisk that can be defined to a virtual machine. It exists until the machine logs off, or the disk is detached. The t-disks are allocated in special areas of disk volumes labeled TDSK. The number and size of the t-disks are limited by the amount of TDSK area allocated to CP.

Xedit — File editor provided by the CMS environment.

Section 3 Terms:

CPUID — Stands for “Central Processing Unit ID”. It consists of the serial number and model of the processor on which the first level CP runs. Each virtual machine also has a CPUID which, by default, matches the first level processor. However it can also be set by an option in the directory or by a CP command.

DCSS — Stands for “Discontiguous Saved Segment”. It consists of one or more saved segments that hold shareable, reentrant programs. These programs are loaded and executed using Diagnose 64 rather than the IPL command. Portions of some program products such as DB2, QMF or compilers are stored in DCSS.

NSS — Stands for “Named Save System”. A NSS is a collection of sharable programs that are written to a saved segment and can be invoked by the IPL command. An example of a NSS is the reentrant portion of CMS which is saved in a segment named CMS during the installation process. When a user issues the IPL CMS command, the segment containing the NSS is linked to the virtual machine and executed, establishing the CMS environment. All virtual machines using CMS will share the same segment, which is a more efficient use of memory.

RSCS — Stands for “Remote Spooling Communication Subsystem”. It is a licensed feature of z/VM that provides printing, job submission and communication services.

Saved Segment — Refers to a range of 1Meg pieces of virtual storage that can be used to store data or reentrant programs. Saved Segments are normally used to share either data or programs between multiple virtual machines.

Track — A “open source” program available to the VM community. Originally written by Serge Goldstein of Princeton University, it is now maintained by Jim Vincent of Nationwide. Track is a very powerful diagnostic tool. It allows you to see various aspects of a running virtual machine without interrupting it. This includes viewing the console, PSW, registers and virtual devices.

Section 5 Terms:

Daemon — Unix and Linux term for a service running on the system. Similar in concept to a service virtual machine in z/VM.

PROP — Stands for “Programmable Operator”. This is a service provided by CP and CMS in conjunction. The PROP program is part of CMS. It allows you to define a routing table that checks an incoming message and takes an action based on it. This can be used in conjunction with the CP secondary console function to allow you to build service machines that monitor the operations of guest systems and take action based on specific messages on the console. You can use this as the basis for building very powerful monitors for your system.

RTABLE — Stands for “Routing Table”. It is the file that PROP uses to select out incoming messages and take specific action.

SCIF — Stands for “Secondary Console Interactive Facility”. It refers to the function in CP that allows you to define a secondary console machine for a disconnected service virtual machine. All console messages that would normally appear on the console are sent to the secondary console machine. If the PROP program is running in the secondary console machine, it can take action based on the incoming messages.

SYSLOG — Refers to a service normally found on Unix and Linux systems. It provides message logging. Actions can be taken based on the text of an incoming message.

SYSRES — Abbreviation for “System Residence Volume”. It refers to the disk volume that contains the primary PARM disk and is IPL’d in order to bring up CP.