IBM

z/VM on an EFS Base: Getting Started

ITSO/EFS Project

- Practical introduction to simple z/VM usage
- Specific examples using Linux and z/OS virtual machines
- Assumes use of a specific z/VM implementation

Bill Ogden

Redbooks

ibm.com/redbooks

International Technical Support Organization

z/VM on an EFS Base: Getting Started

ITSO/EFS Project

April 2004

SG24-7009-00

Note: Before using this information and the product it supports, read the information in “Notices” on page v.

First Edition (April 2004)

This edition applies to z/VM Version 4 Release 4 and FLEX-*ES* Releases 6.2.15 and 6.2.16.

Contents

Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:
IBM Director of Licensing, IBM Corporation, North Castle Drive Armonk, NY 10504-1785 U.S.A.

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:
This information contains sample application programs in source language, which illustrates programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. You may copy, modify, and distribute these sample programs in any form without payment to IBM for the purposes of developing, using, marketing, or distributing application programs conforming to IBM's application programming interfaces.

Trademarks

The following terms are trademarks of the International Business Machines Corporation in the United States, other countries, or both:

ESCON®
@server®
ibm.com®
IBM®
Lotus®
MVS™
OS/2®
OS/390®
PR/SM™
RACF®
Redbooks (logo) ™
Redbooks™
S/390®
ThinkPad®
VTAM®
WebSphere®
xSeries®
z/OS®
z/VM®
zSeries®

The following terms are trademarks of other companies:

Intel, Intel Inside (logos), MMX, and Pentium are trademarks of Intel Corporation in the United States, other countries, or both.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Other company, product, and service names may be trademarks or service marks of others.

Preface

This IBM® Redbook provides a basic introduction to z/VM® usage in a specific environment and assumes the use of a particular z/VM implementation (which is described in detail). With this starting point, we discuss the initial structure of z/VM and describe a number of common tasks (step by step) along with the rationale for these tasks. The user is assumed to have a common z/OS® implementation or a common Linux for S/390® system, or both, already installed. We assume that the total operation is on an IBM ThinkPad® with the FLEX-ES product installed.

The author

Bill Ogden is a retired IBM Senior Technical Staff Member, still working on projects with the International Technical Support Organization, and is normally located in Poughkeepsie. He specializes in smaller S/390 and z/OS systems and has a strong interest in new users and skills development for these platforms.

This book was written while at St. Michael's Indian School on the Navajo reservation in Arizona.

Many thanks to the following people for their contributions to this project:

Gary Eheman, Fundamental Software, Inc., who provided most of the z/VM skills reflected in this book.

Mike MacIsaac, IBM Poughkeepsie, who helped with z/VM skills and with the Linux usage.

Scott Carter, Fundamental Software, Inc., who helped with FLEX-ES questions and provided his usual eagle-eyed proofreading assistance.

Become a published author

Join us for a two- to six-week residency program! Help write an IBM Redbook dealing with specific products or solutions, while getting hands-on experience with leading-edge technologies. You'll team with IBM technical professionals, Business Partners and/or customers.

Your efforts will help increase product acceptance and customer satisfaction. As a bonus, you'll develop a network of contacts in IBM development labs and increase your productivity and marketability.

Find out more about the residency program, browse the residency index, and apply online at:

ibm.com/redbooks/residencies.html

Comments welcome

Your comments are important to us!

We want our Redbooks™ to be as helpful as possible. Send us your comments about this or other Redbooks in one of the following ways:

- Use the online **Contact us** review redbook form found at:

  ibm.com/redbooks

- Send your comments in an Internet note to:

  redbook@us.ibm.com

- Mail your comments to:

  IBM Corporation, International Technical Support Organization
  Dept. HYJ Mail Station P099
  2455 South Road
  Poughkeepsie, NY 12601-5400

1

Introduction

This redbook is based on a specific implementation of z/VM Version 4.4 running under FLEX-*ES*, typically on an IBM ThinkPad. It is intended to help new z/VM users start using and administering z/VM for simple purposes in a FLEX-*ES* environment. Most virtual machine examples are based on either z/OS (as installed from the current Application Development CDs) or on Linux for S/390 installed as described in Chapter 9 of *EFS Systems on a Linux Base: Additional Topics*, SG24-7008.

The z/OS Application Development CDs (commonly known as the *AD systems*) are available only to members of IBM PartnerWorld for Developers who obtained S/390 systems through this program or to IBM employees participating in the ITSO/EFS project. However, it is a straightforward z/OS implementation and should be similar to other basic z/OS implementations.

The Linux for S/390 that is used for some examples is based on SUSE LINUX Enterprise Server 8.[1]

The FLEX-*ES* system is a product of Fundamental Software, Inc., based in Fremont, California. It is an S/390 emulator and is more fully described in *EFS Systems on a Linux Base: Getting Started*, SG24-7007. FLEX-*ES* can be purchased through many business partners of FSI. The company, commonly known as FSI, can be reached through the following URL:

`http://www.funsoft.com`

Red Hat Linux, used as the base system for FLEX-*ES*, is a widely-known Linux distribution for PCs.

Frequent references are made to two earlier Redbooks:

- *EFS Systems on a Linux base: Getting Started*, SG24-7007
- *EFS Systems on a Linux base: Additional Topics*, SG24-7008

We usually reference these two books as the *Getting Started* book and the *Additional Topics* book. Be certain to use the SG24-7007-01 version of the *Getting Started* book.

The product “stack” discussed in this redbook is illustrated in Figure 1-1 on page 2.

[1] The Linux for S/390 version used was from SuSE before they were acquired by Novell.

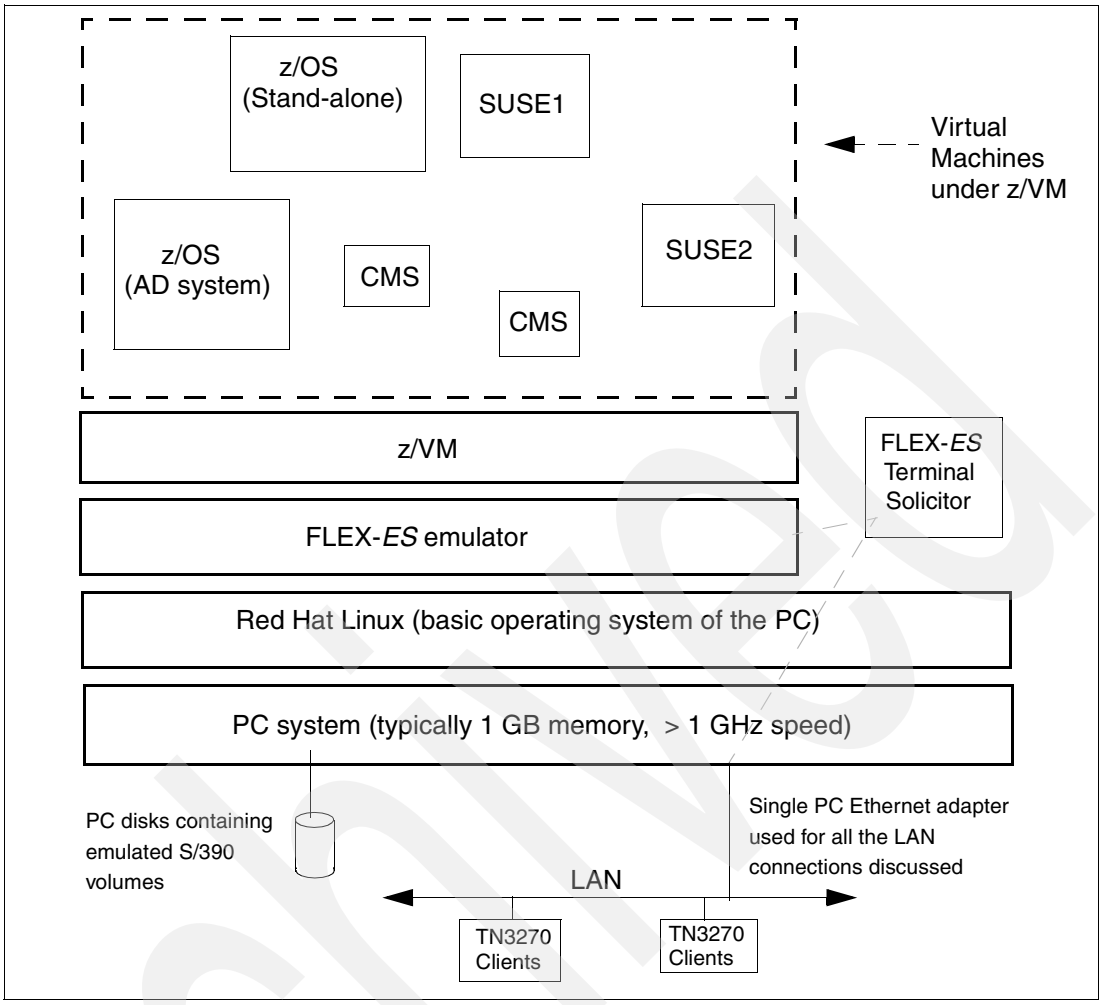

Figure 1-1 Product stack

The two other Redbooks just mentioned describe the installation and use of the base Red Hat Linux, FLEX-*ES*, and some of the networking arrangements that are possible.

Characteristics of the z/VM implementation described in this document include:

- A standard z/VM release is used. There are no changes required to use it with FLEX-*ES*.
- TN3270 clients connected through the FLEX-*ES* Terminal Solicitor appear to z/VM (and to virtual machines running under z/VM) as local, channel-attached, non-SNA 3270 terminals. The TN3270 clients work through TCP/IP in Red Hat Linux using a LAN connection (or a loopback connection for x3270 clients on the Red Hat Linux desktop).
- FLEX-*ES* can use an Ethernet connection[2] to emulate multiple IBM 3172 control units; these are commonly known as LCS interfaces. z/VM (and virtual machines under z/VM) can operate as though multiple 3172 LCS devices are available.
- A single physical LAN interface can provide TCP/IP connectivity to Red Hat Linux, the FLEX-*ES* emulator, z/VM TCP/IP, and to virtual machines under z/VM.

[2] Token-ring interfaces can be used instead of Ethernet. However, all the examples in this redbook use Ethernet interfaces.

1.1 ITSO/EFS z/VM Version 4.4 CD

A CD containing a pre-built z/VM system is available to IBM participants in the ITSO/EFS project. All the discussion and examples in this book are based on this particular z/VM system running under FLEX-*ES*. The general z/VM concepts discussed should apply to any z/VM implementation, but the specific implementation and examples in this document are based on the z/VM system on this CD.

An identical z/VM implementation can be constructed, starting with the standard z/VM product tapes or CDs ordered from IBM. Chapter 7, “z/VM installation” on page 63 describes exactly how the ITSO/EFS z/VM V4.4 CDs were built.

1.2 LAN environment

Our discussion and examples assume that the FLEX-*ES* system is attached to a small router, and no external connections (beyond the router) are discussed. We arbitrarily used the following IP addresses:

192.168.0.1	The router itself
192.168.0.110	Red Hat on the ThinkPad used for this project
192.168.0.111	z/OS (natively under FLEX-*ES* or under z/VM)
192.168.0.112	SUSE LINUX (natively under FLEX-*ES* or under z/VM)
192.168.0.113	z/VM TCP/IP
192.168.0.114	(Second SUSE system, if used, only under z/VM)

2

Installation

The instructions in this chapter apply only if the ITSO/EFS project z/VM CD is available for installation.

For the system and examples discussed in this book, we assume that the following 3390 volumes are installed:

- z/VM system (as provided with the ITSO/EFS z/VM CD):
  - 440RES. This 3390-3 is the IPL volume for z/VM 4.4.
  - 440W01. This 3390-3 is a standard z/VM 4.4 volume that contains additional system minidisks.
  - 440W02. This 3390-3 is a standard z/VM 4.4 volume that contains additional system minidisks.
  - VMPAG1. This 3390-3 is a volume we added to the ITSO/EFS z/VM system. It contains paging space (70%) and space for minidisks (30%).
- z/OS 1.4s (or any recent AD z/OS version):
  - S4RES1. This 3390-3 is the IPL volume for z/OS 1.4s.
  - S4RES2. This 3390-3 contains additional z/OS libraries.
  - OS39M1. This 3390-3 contains parameter libraries, paging space, spooling space, and so forth.
  - S4USS1. This 3390-3 contains the HFS file systems for UNIX® System Services.
  - SARES1. This 3390-3 contains a separate, single-volume z/OS system that is intended as an emergency repair system or as a driver for rearranging the normal system volumes.
  - WORK02. This 3390-1 volume was added for local data sets and is optional.
- SUSE LINUX Enterprise Server 8:
  - LNXA01. This 3390-3 contains a complete SUSE LINUX for S/390 implementation.
  - Three more Linux volumes can be added, based on the FLEX-*ES* definitions provided.

The z/OS systems (the “normal” z/OS 1.4 system and the single-volume version) and the SUSE system are used in z/VM virtual machines. z/VM is not dependent on these particular releases, of course, but our specific examples assume these releases.

The first four z/OS volumes (S4RES1, S4RES2, OS39M1, and S4USS1) are the minimum configuration for IPL for the AD system. The single-volume z/OS system is used merely to

have a second independent z/OS that can be run simultaneously. The sample configuration treats these volumes (and the locally-added WORK02 volume) as shared DASD between the two z/OS systems. Additional volumes of the z/OS AD system could be included with minimal changes to the setup.

In the interests of simplicity, we installed all the emulated 3390 volumes as simple Linux files. (This is in contrast to the use of raw disk devices, which are recommended for production systems and for better performance.) The installation of the z/OS volumes on the FLEX-*ES* system is described in Chapter 5 of the *Getting Started* redbook. The z/OS volumes can be installed before or after the z/VM volumes.

The installation of the single-volume SUSE LINUX system is described in Chapter 9 of the *Additional Topics* redbook. This produces a single copy of the SUSE system. The LINUX for S/390 system is not required for most of the examples in this document. Like the z/OS systems, it simply provides an additional system to run as a virtual machine under z/VM. The SUSE system, if used, can be installed either before or after the z/VM system.

The z/VM implementation we describe keeps the z/OS and SUSE LINUX volumes in their "native" states, and both the z/OS and SUSE LINUX systems can be run without z/VM. This is convenient for beginning z/VM users, who might not want to use z/VM all the time. It is also realistic for z/OS usage. It might not be completely realistic for LINUX for S/390 usage, because this often involves many system "images" (virtual machines) and these might use small minidisks to conserve real disk space. Systems using z/VM minidisks can run only under z/VM. (An exception involves *full-volume* minidisks; this is discussed later.)

2.1 Machine requirements

If all the volumes just described are installed, the PC disk requirements are:

- Ten 3390-3 volumes (five for z/OS, four for z/VM, one for SUSE LINUX). This requires approximately 28 GB disk space on the PC.
- One 3390-1 volume (for the optional 3390-1 WORK02 volume). This requires approximately 950 MB.
- One (or more) 3390-3 volumes if SUSE LINUX cloning will be done. This requires approximately 2.8 GB per clone.
- Approximately 3 to 6 GB for Red Hat Linux, depending on the options selected.

The total is 30 to 40 GB of disk space. With 30 GB, it would be necessary to omit something, such as the z/OS stand-alone system or the SUSE LINUX system. Alternatively, all the z/OS volumes could be omitted and a number of SUSE LINUX clones created. With 40 GB of disk space, all our examples can be implemented.

We suggest a *minimum* of 1 GB for PC memory. All of the examples in this book will run in a 1 GB IBM ThinkPad. However, more memory is better.

Disk performance is always important. z/VM normally pages virtual machine memory, and this is in addition to whatever I/O and paging is performed by the virtual machines. Disk performance can be a bottleneck, especially on a ThinkPad. If using a larger IBM @server® xSeries® server with one or more RAID adapters, disk performance should not be a problem.

Serious use of z/VM and z/OS or multiple LINUX for S/390 virtual machines, or both, on a ThinkPad should involve raw disk devices for FLEX-*ES* emulated volumes. Using raw devices provides more performance and allows disk tuning by volume. For the purposes of this redbook, which concentrates on basic z/VM familiarization and typical administrative tasks, a ThinkPad using simple Red Hat Linux files for emulated volumes is sufficient. The installation and use of raw disk devices is described in the *Additional Topics* redbook.

2.2 Installing the ITSO/EFS z/VM CD

We assume that the z/VM volumes are to be installed as simple Linux files in the /s390 directory. The CD was created using the FLEX-*ES* **ckdbackup** command, and the results were compressed with **gzip**. This is the recommended method of preparing CD distributions for FLEX-*ES* and is described in Chapter 5 of the *Additional Topics* book.

The installation process first creates empty 3390 volumes:

```
$ ckdfmt -r -n /s390/440RES 3390-3
$ ckdfmt -r -n /s390/440W01 3390-3
$ ckdfmt -r -n /s390/440W02 3390-3
$ ckdfmt -r -n /s390/VMPAG1 3390-3
```

The “empty” emulated volumes must be created before the contents can be restored with the **ckdrestore** command, and this is done with **ckdfmt** commands. Next, the CD contents are restored into these volumes:

```
(Load the CD in the CD-ROM drive and let Red Hat automount find it)
$ gunzip -c /mnt/cdrom/440RES.gz | ckdrestore - /s390/440RES
$ cp /mnt/cdrom/Multi  /usr/flexes/rundir/Multi
$ cp /mnt/cdrom/Multi.sh  /usr/flexes/rundir/Multi.sh
$ gunzip -c /mnt/cdrom/440W01.gz | ckdrestore - /s390/440W01
$ gunzip -c /mnt/cdrom/440W02.gz | ckdrestore - /s390/440W02
$ gunzip -c /mnt/cdrom/VMPAG1.gz | ckdrestore - /s390/VMPAG1
(Eject the CD.)
```

The **cp** command steps copy a sample FLEX-*ES* definition and a shell script from the CD to your disk. We assume that your FLEX-*ES* definitions are in /usr/flexes/rundir; if not, change the file name in the **cp** commands.

2.3 FLEX-ES definitions

We included a FLEX-*ES* definition file named Multi. This file makes a number of assumptions, and you will probably need to edit and customize this file. You can use the Linux editor of your choice. We found that **gedit** was easy to use, but you can use **vi** or other common editors. To use **gedit**, open a terminal window on the Red Hat gnome desktop:

```
$ cd /usr/flexes/rundir
$ gedit Multi
```

The **gedit** editor is very PC-like, and most people have no problem using it.

The distributed Multi definition file is approximately similar to this:

```
system Multi:                                   # System file = Multi.syscf
memsize(786432)                                 # S/390 size.
cachesize(4096)
instset(z)                                      # z machine.
cpu(0)
channel(0) local
channel(1) local
channel(2) local
# 3270s used by all
cu devad(0x700,7) path(0) resource(CU3174)      # 3270s at addresses 700, 701, ...
# MVS control units
cu devad(0xA80,16) path(2) resource(MVS3990)    # 16 x 3390s at A80, A81, ...
cu devad(0xE20,2) path(1) resource(MVS3172)     # TCP/IP 3172 at E20/E21
# Linux control units
```

```
cu devad(0x200,4) path(2) resource(LN3990)      # 4 x 3390 at 200, 201, 202, ...
cu devad(0x2E0,2) path(1) resource(LN3172A)     # 3172 for Linux A
cu devad(0x2E2,2) path(1) resource(LN3172B)     # 3172 for Linux B
# VM control units
cu devad(0x400,4) path(2) resource(VM3990)      # 4 x 3390 at 400, 401, 402, ...
cu devad(0x4E0,2) path(1) resource(VM3172)      # 3172 at 4E0/4E1 for VM
cu devad(0x460,1) path(2) resource(VM3480)      # tape drive
end Multi

resources Multi:                                # Resource name = Multi.rescf
CU3174: cu 3174
interface local(1)
device(00) 3278 mstcon                          # The number of 3270s defined here will
device(01) 3278 L701                            # be the total number that can be logged
device(02) 3278 L702                            # onto VM, and dialed to MVS, and
device(03) 3278 L703                            # connected to SUSE (as a 3215).
device(04) 3278 L704                            #
device(05) 3278 L705                            # Add more devices if you need them, and
device(06) 3278 L706                            # update the count in the cu statement
end CU3174

# MVS resources
MVS3990: cu 3990
interface local(1)
device(00) 3390-3 /s390/S4RES1                  # address A80
device(01) 3390-3 /s390/SRES2                   # address A81
device(02) 3390-3 /s390/OS39M1                  # address A82
device(03) 3390-3 OFFLINE                         # use OFFLINE devices to make
device(04) 3390-2 OFFLINE                         # the volume addresses match the
device(05) 3390-3 OFFLINE                         # commonly documented addresses for
device(06) 3390-3 OFFLINE                         # the AD volumes
device(07) 3390-3 /s390/S4USS1                  # address A87
device(08) 3390-3 OFFLINE
device(09) 3390-2 OFFLINE
device(10) 3390-3 OFFLINE
device(11) 3390-3 OFFLINE
device(12) 3390-3 /s390/SARES1                  # address A8C
device(13) 3390-3 OFFLINE
device(14) 3390-3 OFFLINE
device(15) 3390-1 /s390/WORK02                  # address A8F
end MVS3990

MVS3172: cu 3172
interface local(1)
options 'ipaddress=192.168.0.121,adapternumber=0'  #
device(00) 3172 eth0                            # address E20
device(01) 3172 OFFLINE                         # address E21
end MVS3172

# Linux resources
LN3990: cu 3990
interface local(1)
device(00) 3390-3 /s390/LNXA01                  # address 200
device(01) 3390-3 OFFLINE
device(02) 3390-3 OFFLINE
device(03) 3390-3 OFFLINE
end LN3990

LN3172A: cu 3172
interface local(1)
```

```
options 'ipaddress=192.168.0.122,adapternumber=0'
device(00) 3172 eth0                        # address 2E0
device(01) 3172 OFFLINE                     # address 2E1
end LN3172A

LN3172B: cu 3172
interface local(1)
options 'ipaddress=192.168.0.124,adapternumber=0'
device(00) 3172 eth0                        # address 2E2
device(01) 3172 OFFLINE                     # address 2E3
end LN3172B

# VM resources
VM3990: cu 3990
interface local(1)'
device(00) 3390-3 /s390/440RES              # address 400
device(01) 3390-3 /s390/440W01              # address 401
device(02) 3390-3 /s390/440W02              # address 402
device(03) 3390-3 /s390/VMPAG1              # address 403
end VM3990

VM3172: cu 3172
interface local(1)
options 'ipaddress=192.168.0.113,adapternumber=0'
device(00) 3172 eth0                        # address 4E0
device(01) 3172 OFFLINE                     # address 4E1
end VM3172

VM3480: cu 3422
interface local(1)
device(00) 3422 OFFLINE                     # address 460
end VM3480

end Multi
```

You should look at your copy of the FLEX-*ES* definitions. We might have changed minor details since publication of this redbook, and you should adjust the following discussion as needed to match your FLEX-*ES* definitions.[3]

The FLEX-*ES* system definition creates an emulated S/390 with 768 MB memory (and no expanded storage). This is a reasonable size for a base PC with 1 GB memory.[4] The instset(z) statement indicates that zArchitecture (also known as ALS-3 or "64-bit") operations are emulated.

The combination of z/VM and FLEX-*ES* provides a particular advantage in that we can assign whatever addresses (device numbers) we want to the various emulated S/390 devices. z/VM will adapt to any addresses, and there are no IOCDS or "real" hardware addresses to worry about.[5]

[3] You might notice that stanza VM3480 (in the resource definitions) defines a 3422 tape drive. The "VM3480" is just an arbitrary name, although we probably should have used a name such as VM3422 to be slightly less confusing. Device type 3422 is required for z/VM to use an OMA CD.

[4] There is an extended discussion about memory size (PC, virtual, and emulated) in Chapter 2 of the *Additional Topics* redbook.

[5] This is not quite true if channel-attached devices (through FSI channel adapters) are involved. However, none are involved in these examples.

We decided on the following strategy:

- Create addresses that match the commonly-used addresses for the AD z/OS system. That is, use addresses in the A8x range for 3390 volumes, addresses E20/E21 for a 3172 LAN adapter, and 3270 addresses in the 700 range.[6]
- Place all the SUSE devices in the 2xx address range.
- Place all the z/VM devices in the 4xx address range.
- Create multiple emulated 3172 devices and assign IP addresses as follows:
  - z/OS (device address E20/E21) IP address: 192.168.0.111
  - z/VM (device address 4E0/4E1) IP address: 192.168.0.113
  - SUSE A (device address 2E0/2E1) IP address: 192.168.0.112
  - SUSE B (device address 2E2/2E3) IP address: 192.168.0.114
  - (Red Hat Linux IP address: 192.168.0.110)

The addresses for S/390 devices are assigned in the CU statements in the system section of the FLEX-*ES* definitions. For example, the following statement defines seven devices with addresses 700, 701, 702, and so forth:

```
cu devad(0x700,7) path(0) resource(CU3174)
```

The nature of the devices are defined in a stanza named CU3174 in the resources section of the FLEX-*ES* definitions. There *must* be seven devices defined in this stanza, although any or all can be defined as OFFLINE. The path number relates to a defined channel. In the simple example here, the channel definitions are arbitrary and have little meaning. (We named the device at address 700 *mstcon* to be compatible with examples in previous redbooks; these names are arbitrary.)

We defined four 3390-3 devices for SUSE LINUX and two 3172 LAN units. The initial SUSE system that we are installing (as described in Chapter 9 of the *Additional Topics* redbook) uses one 3390-3 and one 3172 and the additional 3390 units are OFFLINE. The additional units are for cloning, expansion in the future, or both. If needed, the additional 3390 volumes can be created using **ckdfmt** commands:

```
$ ckdfmt -r -n /s390/LNXA02 3390-3
$ ckdfmt -r -n /s390/LNXB01 3390-3
$ ckdfmt -r -n /s390/LNXB02 3390-3
```

The volsers are arbitrary. After creating the volumes (with the **ckdfmt** commands), the new volumes could be added to the FLEX-*ES* resource definitions, replacing the OFFLINE definitions. Remember that the **ckdfmt** command creates a basic volume (with tracks and cylinders defined) but does not initialize the volume with a volser or VTOC or Linux partitions.

2.3.1 Dual-use FLEX-ES definitions

The FLEX-*ES* definitions previously described have a particular advantage. Using the systems we discussed (the AD CD-ROM system and the SUSE LINUX implementation described in the *Additional Topics* book), we can run any of our systems directly (without z/VM) or under z/VM without changing the FLEX-*ES* definitions. This is due to the following:

- The FLEX-*ES* definitions include the device addresses used by the AD z/OS system. In particular, these are addresses A8x (3390s), 70x (3270s), and E2x (3172s).
- The FLEX-*ES* definitions include the device addresses we specified when we built the SUSE LINUX system. These are 200 (3390) and 2Ex (3172).

[6] While working under z/VM, all 3270 addresses are arbitrary, and there was no reason to match the 70x addresses normally used by z/OS. However, we needed to select addresses for local 3270 terminals and the 70x range was as appropriate as any other address we might have selected.

- z/VM adapts to available device addresses when it is IPLed.
- There are no conflicts in the addresses. In particular, z/OS (through the active IODF) does not use 2xx and 4xx addresses.

This addressing convenience is not required. For example, if we ran only under z/VM, we could assign any arbitrary addresses (through the FLEX-*ES* definitions) to our devices. We could then have z/VM transform these addresses into whatever addresses are required for each operating system run under z/VM. If we did this, we would need different FLEX-*ES* definitions if we wanted to run one of our operating systems in a stand-alone mode (without z/VM).

2.4 FLEX-*ES* shell script

As in all the redbook examples, we use a shell script to start FLEX-*ES* operation. We created the following script in the /usr/flexes/rundir/Multi.sh file:

```
flexes Multi.syscf
x3270 -model 3 -keymap pc -port tn3270 localhost:mstcon &
x3270 -model 3 -keymap pc -port tn3270 localhost:L701 &
flexescli localhost Multi
```

This simply starts FLEX-*ES* emulation and opens two 3270 sessions (addresses 700 and 701) on the Red Hat Linux desktop.

3

Overview of our VM system

In terms of volumes, our installed z/VM system appears as shown in Figure 3-1. Our other disk volumes, for z/OS and Linux, can be used with the z/VM system, but are not actually part of z/VM. We added the VMPAG1 volume after installing the z/VM 4.4 system on the first three volumes.

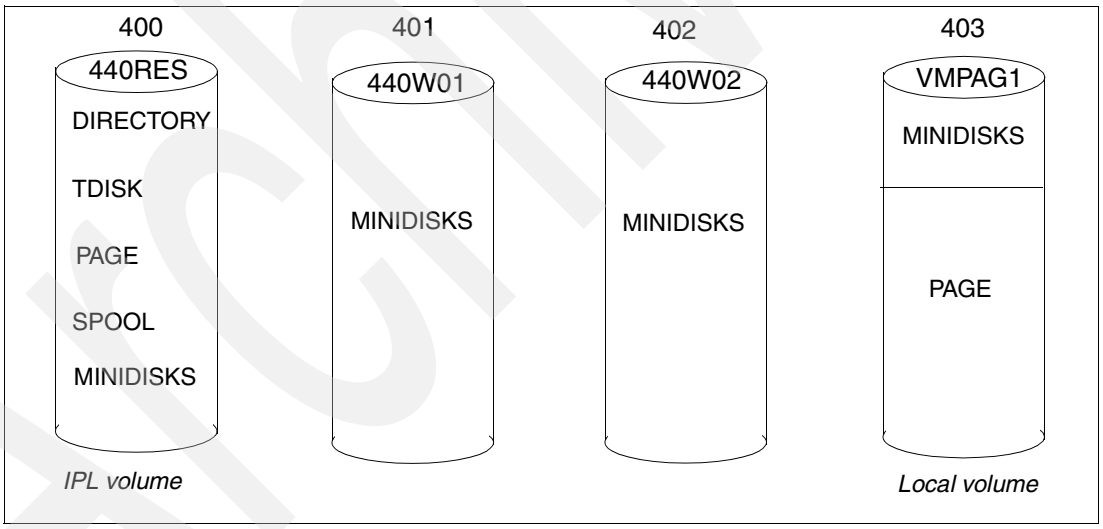

Figure 3-1 Our basic z/VM system

z/VM works with three types of disk volumes:

- CP-owned volumes. All four of our volumes are in this category. Volumes that contain the minidisks needed to run z/VM or that contain paging space or temporary disk space are CP-owned. There is a list (that we can modify) on a minidisk on the 440RES volume that specifies which volsers are CP-owned. CP-owned volumes can also contain user minidisks. The volume we added (VMPAG1) contains both paging space and user minidisks.
- User-owned volumes. These typically contain minidisks used by many users. Again, there is a list of the volsers that are user-owned. In our simple z/VM implementation, we have no user-owned volumes.

- Other disks. These can be used as standard DASD volumes or as whole-pack minidisks. All our z/OS and Linux volumes are used as whole-pack minidisks. This means that they are standard non-VM volumes, but are defined in the z/VM directory as whole-pack minidisks.

When we **ipl** z/VM, we are starting the Control Program (CP) portion of z/VM; this is sometimes known as the hypervisor. Every z/VM user, whether a simple CMS (Conversational Monitor System) user or a complete z/OS system, runs in a virtual machine. Every logon to z/VM creates a new virtual machine within z/VM. A virtual machine is typically in one of three environments:

- It has IPLed CMS, and a user can access both CMS and CP functions.
- It has IPLed an operating system, such as z/OS, and is under the control of that operating system.
- It has not IPLed anything. A user can enter only basic CP commands.

When using CMS (or interacting directly with CP, if no operating system has been started), the 3270 session is in a pseudo-3215 mode. The IBM 3215 console provides a typewriter-like interface. There is a single command line at the bottom of the screen, and output is *pushed up* off the top of the screen.

The remainder of this chapter describes basic attributes of our small z/VM system. If possible, we suggest you execute the commands we describe as you work through this material.

3.1 IPL z/VM

Before IPLing z/VM, you need to complete the necessary FLEX-*ES* definitions. A sample definition file is described in 2.3, “FLEX-ES definitions” on page 7. After making any adjustments, the definitions must be compiled, for example:

```
$ cd /usr/flexes/rundir                 (Assume file Multi is here)
$ gedit Multi                           (Make changes and save it)
$ cfcomp Multi                          (Compile it)
```

The compilation will produce two files: Multi.rescf and Multi.syscf. To use these files, we need to start the FLEX-*ES* resource manager and then use a shell script (an example is described in 2.4, “FLEX-ES shell script” on page 11) to start S/390 emulation. For example:

```
$ cd /usr/flexes/rundir                 (Assume FLEX-ES working files are here)
$ su                                    (Must be root to start resource manager)
# resadm -s Multi.rescf                 (Use resadm to start resource manager)
# exit                                  (Leave root)
$ . Multi.sh                            (Start S/390 emulation)
flexes>                                 (Prompt for flexescli commands)
```

The process is described in detail in the *Getting Started* redbook, and this is a very brief summary. The shell script (Multi.sh) should open two x3270 sessions on the Red Hat desktop and leave a `flexes>` prompt in the terminal window.

We can IPL z/VM by issuing an appropriate flexescli **ipl** command. Using the systems and definitions described in Chapter 2, “Installation” on page 5, the command would be:

```
flexes> ipl 400
```

Within a few seconds, the screen shown in Figure 3-2 on page 15 should appear on the x3270 session with address 700. Enter **warm** in response to the prompt (and press Enter). Reply **no** to a prompt about changing the Time of Day (TOD) value. (Or, if appropriate, enter **yes** and follow the prompts to change the clock.)

Whenever CP READ or VM READ appears in the lower-right portion of the screen, VM is waiting for a command or response. Whenever HOLDING or MORE appears, we must clear the screen.[7] Whenever RUNNING is displayed, CMS is operating and is not waiting for any particular response from us.

```
17:41:29 z/VM  V4 R3.0  SERVICE LEVEL 0201 (64-BIT)
17:41:30 SYSTEM NUCLEUS CREATED ON 2002-05-09 AT 17:30:26, LOADED FROM 430RES
17:41:30
17:41:30 ************************************************************
17:41:30 * LICENSED MATERIALS - PROPERTY OF IBM*                      *
17:41:30 *                                                           *
17:41:30 * 5739-A03 (C) COPYRIGHT IBM CORP. 1983, 2002. ALL RIGHTS    *
17:41:30 * RESERVED. US GOVERNMENT USERS RESTRICTED RIGHTS - USE,     *
17:41:30 * DUPLICATION OR DISCLOSURE RESTRICTED BY GSA ADP SCHEDULE   *
17:41:30 * CONTRACT WITH IBM CORP.                                    *
17:41:30 *                                                           *
17:41:31 * * TRADEMARK OF INTERNATIONAL BUSINESS MACHINES.           *
17:41:31 ************************************************************
17:41:31
 17:41:31 HCPZCO6718I Using parm disk 1 on volume 430RES (device 0400).
 17:41:31 HCPZCO6718I Parm disk resides on cylinders 391 through 435.
17:41:31 Start ((Warm|Force|COLD|CLEAN) (DRain) (DIsable)  (NODIRect)
17:41:31       (NOAUTOlog)) or (SHUTDOWN)


                                                        CP READ   FLEXVM44
================================================================================
```

Figure 3-2 First OPERATOR screen after IPL

After clearing the screen several times, lines similar to those shown in Figure 3-3 on page 16 should appear. At this point, the OPERATOR user ID, is active on this screen. For the small, single-user VM we describe here, the OPERATOR user ID is seldom needed. The typical VM operation is to use the **disc** (disconnect) function for this session. This leaves OPERATOR running as a VM virtual machine, but with no console connected to it. We can log on to the OPERATOR user ID again later and resume its activities.

[7] If you use the extended set of x3270 keyboard modifications described in the *Additional Topics* redbook, the Pause key will clear the screen. With unmodified x3270 keys, the combination Alt-c will clear the screen.

```
18:31:16 AUTO LOGON  ***       VMSERVS  USERS = 6     BY AUTOLOG1
 18:31:16  * MSG FROM OPERSYMP: 0 RECORDING FILE(S), 0 RECORDS, A DISK 01 % FUL
 L
 18:31:16  * MSG FROM DISKACNT: 6 RECORDING FILE(S), 91 RECORDS, A DISK 08 % FU
 LL
18:31:16 HCPCRC8064I Recording data retrieval has been started; recording *ACCOU
NT for userid DISKACNT.
18:31:16 HCPCRC8064I Recording data retrieval has been started; recording *SYMPT
OM for userid OPERSYMP.
18:31:16 AUTO LOGON  ***       VMSERVU  USERS = 7     BY AUTOLOG1
 18:31:16  * MSG FROM EREP    : 1 RECORDING FILE(S), 75 RECORDS, A DISK 09 % FU
 LL
18:31:16 HCPCRC8064I Recording data retrieval has been started; recording *LOGRE
C for userid EREP.
18:31:16 AUTO LOGON  ***       VMSERVR  USERS = 8     BY AUTOLOG1
 18:31:16  * MSG FROM AUTOLOG1: RECORDING ALL OFF WAS EXECUTED
18:31:31


                                                                RUNNING   FLEXVM44
================================================================================
```

Figure 3-3 Last OPERATOR screen after IPL

The **disc** command should produce a logo/logon screen, similar to that shown in Figure 3-4.

```
z/VM ONLINE



ITSO/EFS PROJECT          ITSO/EFS/PROJECT          ITSO/EFS PROJECT


                    /  V      V  MM     MM      IBM Internal Use
           zzzzz   /    V    V   M M   M M
              z   /      V  V    M  M M  M          <-- + -->
             z   /        V V     M   M   M
            zzzzz /         V      M       M      IBM Internal Use


  A basic z/VM 4.4.0 system for new VM owners running under FLEX-ES

ITSO/EFS PROJECT          ITSO/EFS/PROJECT          ITSO/EFS PROJECT



Fill in your USERID and PASSWORD and press ENTER
(Your password will not appear when you type it)
USERID   ===>
PASSWORD ===>
COMMAND  ===>
                                                        RUNNING   FLEXVM44
```

Figure 3-4 Logon screen

3.2 Initial use

The VM user MAINT is the primary administration ID and has all initial authority in the system. In a large production operation, MAINT would be used sparingly.[8] In our small sandbox VM system, we use MAINT frequently. Enter MAINT as the user ID and MAINT as the password in the logon screen. (All the VM user IDs[9] provided with a VM system have the passwords the same as the user names. There are a few exceptions where it is not possible to log on to certain user names.)

This should produce the results shown in Figure 3-5. The system will stop after the line containing `z/VM V4.3.0 2003-12-06 23:13`[10] and will display VM READ on the bottom right of the screen. Press Enter and the last line shown in the figure will be displayed. At this point, MAINT is logged on to the system and is running CMS.

```
LOGON MAINT
z/VM Version 4 Release 3.0, Service Level 0201 (64-bit),
built on IBM Virtualization Technology
There is no logmsg data
FILES: 0004 RDR,   NO PRT,   NO PUN
LOGON AT 17:35:51 EST FRIDAY 10/31/03
z/VM V4.3.0    2003-12-06 23:13

Ready; T=0.12/0.21 17:35:56

                                                        RUNNING   FLEXVM44
================================================================================
```

Figure 3-5 After MAINT's logon

3.2.1 CMS

CMS is the Conversational Monitor System. It is a small, single-user operating system. Every VM user running CMS is running a separate CMS in a separate virtual machine.[11] CMS is a special case operating system that runs only under VM. VM administration is done through CMS. The `Ready` prompt and the `RUNNING` indicator on the screen are indicators that CMS is running in a virtual machine

It is possible to use CMS for application development and as a base for many user applications. This development and application usage is not described in this redbook; instead, our interest is in using VM to host z/OS and Linux for S/390 operating systems. Nevertheless, basic CMS usage is required to set up and administer a VM system.

CMS has a large set of functions and commands. The basic z/VM Control Program (CP) also has a large set of commands. In general, both sets of commands can be entered on the CMS command line. These commands can be entered in either lowercase or uppercase. In some cases, CP commands must be prefixed by the letters CP (and in rarer cases, by the

[8] MAINT is quite similar to IBMUSER in an initial z/OS installation and is somewhat like root in a Linux system. These user IDs have all the necessary authority to further customize and manage the systems. In production operations, various authorities are delegated to other user IDs, and the MAINT/IBMUSER IDs are seldom used. In small sandbox systems, MAINT/IBMUSER/root are often directly used, even when not strictly necessary.

[9] The terms "user" and "user ID" and "virtual machine name" are used interchangeably, although sometimes the use of "virtual machine name" implies that the corresponding user/user ID has logged on to the system to create the virtual machine.

[10] This line contains the date and time, so your display will have different numbers.

[11] Actually, only one copy of CMS exists. It uses shared virtual memory so that it appears to exist in the virtual memory of every user that IPLs CMS.

characters #CP). If a CP command is not recognized, you can try prefixing it with CP or #CP (and a space).

CMS (and basic CP) use the 3215 typewriter-like screen format previously mentioned. There are exceptions to this, with **XEDIT** (the primary CMS editor program) being the most common exception.

We discuss further aspects of CMS usage throughout this document.

3.2.2 Minidisks

We already used the term “minidisk.” Most S/390 users are somewhat familiar with the term. A complete disk volume can be treated as a single minidisk. This is necessary when sharing whole volumes among multiple virtual machines. These are “whole-pack” or “whole-volume” minidisks and can be considered a special case. If we do not use one of these terms, we mean a “normal” minidisk.

A normal minidisk is simply a set of cylinders on a disk volume. It can be regarded as a virtual disk drive; formatting and usage appears to be independent of any other minidisks on the same physical drive. It is created by using an MDISK statement in the VM directory. For example, assume the following statement appears in the z/VM directory for user JOE:

```
MDISK 222 3390 512 10 USER01 WR
```

This tells z/VM that a minidisk starts at cylinder 512 on volser USER01 (on a 3390 drive) and extends for 10 cylinders. JOE addresses this minidisk as address 222 and JOE has read/write access to the minidisk. JOE must format the minidisk in whatever format needed. There are three common formats:

- z/OS with a label, a VTOC, and the usual MVS™ data sets. z/OS (running under z/VM) sees the minidisk as an otherwise normal 3390 that has only 10 cylinders.
- CMS, which includes all the z/VM control files and structures as well as user CMS disks.
- Linux for S/390 (or Linux for zSeries).

The user of the minidisk is responsible for formatting the volume. In CMS, this might be done with a **format 222 j** command, for example, which would allow the owner to access the volume as CMS unit j. For z/OS use, an ICKDFS job would probably be used.

In principle, user JOHN in the z/VM directory might have the statement:

```
MDISK 111 3390 505 10 USER01 WR
```

Careful examination shows that JOE’s 222 minidisk and JOHN’s 111 minidisk have overlapping extents. This is normally an error and would corrupt whatever data is on these cylinders. However, this illustrates the principle that a minidisk is simply an extent of cylinders that is specified in an MDISK statement somewhere in the z/VM directory.[12]

Other virtual machine definitions might contain LINK statements pointing to JOE’s 222 disk, thus allowing other users to share it.[13]

Some minidisk addresses have predefined or conventional meanings. For example, address 190 contains many CMS modules (and is read-only for most users). Most CMS users have a 191 minidisk, which is their default CMS work disk. There are a few other well-known minidisk addresses used by multiple users. Otherwise, the minidisk address is simply a hexadecimal

[12] The z/VM volumes intentionally overlap full-disk definitions with many minidisks within the same disk. This is done for z/VM management purposes and is not normally done for user minidisks.

[13] There are a variety of security and integrity controls to prevent unwanted sharing and to control the levels of sharing. We are ignoring these in this discussion.

address that you select. A minidisk is identified by the owner's name (that is, the user ID in the VM directory) and the address. Except for the well-known addresses, a minidisk address is simply an arbitrary identifier. However, once established, the minidisk addresses used by applications are usually not changed. For example, the z/VM TCP/IP PROFILE file is on TCPMAINT's 198 minidisk.[14] Documentation and operational instructions refer to this minidisk address and changing the address would require changes to much of the TCP/IP documentation.

3.2.3 CMS minidisks and files

A permanent CMS minidisk is normally created by first defining the minidisk (with an MDISK statement in the VM directory) and then formatting it for use by CMS (with a CMS **format** command). The result is a unique format that is not compatible with z/OS or other operating systems.

CMS cannot access a CMS-formatted minidisk (or a full-pack minidisk, for that matter) until an **access** command is used. (Several minidisks at well-known addresses, such as 190 and 191, are automatically accessed when CMS is IPLed.) The **access** command can look like this:

```
acc 222 e
```

This command, issued by user JOE, says to access JOE's 222 disk as CMS disk *e*. A CMS user can have an *a* disk (normally the 191 drive), a *b* disk, a *c* disk, and so forth.[15] In some cases, a user's CMS disks are searched in a, b, c order when searching for a file.

A CMS user identifies a file with three qualifiers: a file name, a file type, and the drive letter for the file. The file name and file type are each up to eight characters. The drive letter is a single letter (as just described), optionally followed a single number. An example of a CMS file name is:

```
MYDATA TEXT A1
------ ---- --
   |     |   |
   |     |   +-------- File mode (fm)    The notation fn, ft, and fm is common.
   |     +------------ File type (ft)
   +-------------------File name (fn)
```

The first part of the name (MYDATA) is completely arbitrary and should have some meaning to the user. The second part of the name (TEXT) *can* be arbitrary, but *some* applications attach meanings to this part of a file name.[16] For example, if the second part of a file name is EXEC, it is assumed to contain commands for a command interpreter such as REXX. The file mode letter indicates which minidisk is being accessed, as established by an **access** command.

Some file mode letters have standard meanings. For example:

- A - This is the CMS user's default work volume. All typical CMS users have an A disk, and it is created as address 191.

[14] We do not know why this minidisk address was initially chosen.

[15] In a very rough sense, this can be compared to PC DOS file letters: The A drive is a diskette, the C drive is the boot partition, the D drive might be another partition, and so forth. With a PC, the drive letters tend to be fixed. With CMS, the drive letters can be changed with the **access** command.

[16] This is very similar to simple PC DOS name suffixes. For example, the Lotus® 1-2-3 product assumes that any file named xxxx.123 (that is, with the suffix 123) is to be used by that product. Microsoft® Windows® provides a long list of suffixes with presumed meanings. You can use any name you like for a file type, but this might cause a problem if an application (perhaps an application called indirectly) associates a specific purpose with that file type. There is a long list of reserved file types. They are not really *reserved* (blocked from other use), but you should be aware that the names might be assumed to have specific purposes.

- S - This is normally the CMS residence volume and is usually at address 190.
- Y/S - The Y drive (normally address 19E) contains more CMS libraries and an extension. The notation Y/S means that Y is an extension of the S drive.

The file mode numbers for minidisks are:

- 0 - This makes the file private. Other users with read access to your minidisk will not see this file. Users with read/write access will see it.
- 1 - This is the default file mode number for normal read/write files.
- 2 - This functions like number 1 and can be used to subset files.
- 3 - Files with this number are automatically erased when they are read.
- 4 - These files are in a simulated MVS format.
- 5 - This functions like number 1 and can be used to subset files.
- 6 - This indicates that the file should be updated in place, with appropriate programming.

If you ignore file mode numbers, CMS will use the default value of one. This is suitable for most uses.

3.2.4 Inspecting your disks

Assuming you are logged on to z/VM as MAINT and CMS is running, try the commands **`q da all`** and **`q disk`**. These are the common abbreviated forms of the commands **`query dasd all`** and **`query disk`**. User MAINT has the necessary authority to see all the hardware in your system; the same commands from a user with less authority would have different results.

Figure 3-6 on page 21 shows typical results. The **`q da all`** command shows all the defined disks in our FLEX-*ES* definitions. We can see the z/OS volumes and a volume owned by Linux (with volser 0X0200). This command displays information about *real* disk volumes.

```
Ready; T=0.49/0.91 17:41:58
q da all
DASD 0400 CP OWNED  440RES   116
DASD 0401 CP OWNED  440W01   34
DASD 0402 CP OWNED  440W02   1
DASD 0403 CP OWNED  VMPAG1   0
DASD 0200 0X0200  , DASD 0A80 S4RES1  , DASD 0A81 S4RES2  , DASD 0A82 0S39M1
DASD 0A87 S4USS1  , DASD 0A8C SARES1  , DASD 0A8F WORK02
DASD 0201 OFFLINE , DASD 0202 OFFLINE , DASD 0203 OFFLINE , DASD 0A83 OFFLINE
DASD 0A84 OFFLINE , DASD 0A85 OFFLINE , DASD 0A86 OFFLINE , DASD 0A88 OFFLINE
DASD 0A89 OFFLINE , DASD 0A8A OFFLINE , DASD 0A8B OFFLINE , DASD 0A8D OFFLINE
DASD 0A8E OFFLINE
Ready; T=0.01/0.01 20:09:01
q disk
LABEL  VDEV M  STAT   CYL TYPE BLKSZ   FILES  BLKS USED-(%) BLKS LEFT  BLK TOTAL
MNT191 191  A   R/W   175 3390 4096       33        203-01      31297       31500
MNT5E5 5E5  B   R/W     9 3390 4096      127       1180-73        440        1620
MNT2CC 2CC  C   R/W     5 3390 4096       46        452-50        448         900
MNT51D 51D  D   R/W    13 3390 4096      224       1168-50       1172        2340
MNT190 190  S   R/O   100 3390 4096      692      14303-79       3697       18000
MNT19E 19E  Y/S R/O   250 3390 4096     1009      26522-59      18478       45000
Ready; T=0.01/0.01 20:09:07



                                                        RUNNING   FLEXVM44
================================================================================
```

Figure 3-6 Query dasd and query disk commands

The **q disk** command displays information about *minidisk* volumes that are currently accessed by the current (MAINT) user. For example, MAINT is currently accessing its 191 drive as file mode A. If we examine the statements for user MAINT in the VM directory, we will find an MDISK statement defining a minidisk at address 191. The label for a minidisk is assigned by the user and is normally not significant (except, perhaps, to that user).

Many of the fields in the **q disk** command output have obvious meanings. File mode Y/S means that this Y minidisk should be, in effect, concatenated to the S minidisk when the S minidisk is searched. CMS disks are formatted in 4 KB blocks, and the utilization of the blocks is shown in the listing.

We can list the files on a minidisk with a command such as **filelist * * c**. This example says to list all file names (first asterisk) and all file types (second asterisk) on the minidisk currently accessed as drive c. The result is shown in Figure 3-7 on page 22. (We return to this specific listing later because it contains the source file for the VM directory.) We can use the **filelist** command to subset the files names retrieved. For example, **filelist * LOG c** would list only the files on disk c with file type LOG.[17]

[17] We could enter the operands in either uppercase or lowercase. Case is not significant with normal CMS file names.

```
 MAINT    FILELIST A0  V 169  Trunc=169 Size=66 Line=1 Col=1 Alt=9
Cmd   Filename Filetype Fm Format Lrecl    Records     Blocks   Date     Time
x     USER     DIRECT   C1 F         80       1838         36 10/30/03 16:18:02
      USEROLD  DIRECT   C1 F         80       1707         34 10/08/03 12:46:28
      IPWIZARD $FILE$   C1 V         80        129          2 10/07/03  0:14:48
      $INSTD$  $FILE$   C1 F         80         24          1 10/06/03 22:28:41
      INSTDEF  $MSGLOG  C1 V         88         16          1 10/06/03 22:28:41
      $INSTDEF $FILE$   C1 V         80         36          1 10/06/03 22:28:30
      $IVM$    $FILE$   C1 F         80          9          1 10/06/03 22:27:15
      POSTLOAD $MSGLOG  C1 V         98         22          1 10/06/03 22:22:38
      POSTDDR  $MSGLOG  C1 V         57          2          1 10/06/03 22:22:33
      POSTDDR  PRODLIST C1 F         80         18          1 10/06/03 22:22:31
      DDR      $MSGLOG  C1 V        132       4412        145 10/06/03 22:22:01
      $LOAD$   LOG      C1 F         80         62          2 10/06/03 22:21:57
      COMP     $MSGLOG  C1 V         61          7          1 10/06/03 22:21:57
      VMFRMT   EXTENTS  C1 V         80         34          1 10/06/03 21:43:56
      $INST$   $FILE$   C1 V         80         55          2 10/06/03 21:42:08
      $USERDIR TEMPLATE C1 F         80       1707         34  4/04/02 16:54:29
      $ITEMMD$ $TABLE$  C1 F         80        765         15  4/04/02 16:54:06
      SYSTEMCK CONFIG   C1 F         80        281          6  4/04/02 16:47:33
      PRODUCT  LAYOUT   C1 F         80        477         10  4/04/02 16:39:20
      INSTUSER DIRECT   C1 F         80       1707         34  4/04/02 16:38:08
      $IDISK$  $TABLE$  C1 F         80         31          1  4/01/02  7:06:28
      $VMFRMT  TEMPLATE C1 F         80         34          1  3/22/02 14:45:40
      $LOADLOG TEMPLATE C1 F         80         62          2  3/22/02 14:45:20
      MOVE2SFS $TABLE$  C1 F         80        721         15  3/22/02 14:45:12
      UIDMDSKS $FILE$   C1 F         80        116          3  3/22/02 14:44:40
1= Help       2= Refresh  3= Quit   4= Sort(type)  5= Sort(date)  6= Sort(size)
7= Backward   8= Forward  9= FL /n 10=           11= XEDIT/LIST 12= Cursor

====>
                                                          X E D I T  1 File
================================================================================
```

Figure 3-7 Filelist command results

The output from the **filelist** command is in a full-screen format that can be paged with PF8 and PF7. PF3 is used to exit from the filelist screen. This filelist is a very useful format, because CMS commands can be entered in the first column of a line and the file named in that line becomes the operand for the CMS command. Perhaps the most common use is with the **xedit** command (which can be abbreviated **x**).

For example, if we enter **x** in the first column of the first file named (USER DIRECT C1), we should see the screen shown in Figure 3-8 on page 23.

```
 USER     DIRECT   C1  F 80  Trunc=72 Size=1838 Line=0 Col=1 Alt=0

00000 * * * Top of File * * *
00001 ****************************************************************
00002 *  z/VM 4.3.0  SYSTEM DIRECTORY                                *
00003 ****************************************************************
00004 *                                                              *
00005 *  THE ADDRESSES 123 THROUGH 133 ARE VIRTUAL ADDRESSES.        *
00006 *                                                              *
00007 *  NOTES:                                                      *
00008 *  REMEMBER THESE ARE ONLY VIRTUAL ADDRESSES NOT REAL          *
00009 *  ADDRESSES, SO THERE IS NO NEED TO CHANGE THEM TO MATCH      *
00010 *  YOUR HARDWARE ADDRESSES.                                    *
00011 *                                                              *
00012 *                                                              *
00013 ****************************************************************
00014 *                                                              *
00015 *  FOR A DESCRIPTION OF DIRECTORY STATEMENTS SEE:              *
00016 *     VM ENTERPRISE SYSTEM ARCHITECTURE                        *
00017 *     PLANNING AND ADMINISTRATION MANUAL.                      *
00018 *                                                              *
00019 ****************************************************************
00020 *
00021 *
00022 *
00023 DIRECTORY 123 3390 430RES
00024 *
00025 ****************************************************************
00026 *
00027 GLOBALDEFS
00028       POSIXGROUP system 0
====>
================================================================================
```

Figure 3-8 XEDIT screen

Do not change this file until you are fairly certain what you are doing! If you are in XEDIT, use the command **qquit** (on the command line at the bottom of the XEDIT screen) to exit without saving any changes.

3.2.5 XEDIT

XEDIT is the normal CMS editor. Most z/VM administration and customization is done by editing CMS files, and XEDIT is the tool for doing this. Many new z/VM users are already familiar with the ISPF editor under z/OS and will find XEDIT easy to use. It is a full-screen line editor with many features and functions. There is an extensive HELP function that can be accessed by either PF1 or a HELP command.

You can create new CMS files with XEDIT, and we suggest that you practice using XEDIT on a test file before you begin any serious z/VM administration. Assuming you are logged on as MAINT, enter the following command on the CMS command line:

```
x myfile text a
```

This will create a new file on your A disk (assuming you save the file after entering lines). The file name is *myfile* and the file type is *text*; both these names are arbitrary. The command should produce the display shown in Figure 3-9 on page 24. Notice that the user entered the

line command **i20** in the first line. When you press Enter, this inserts 20 blank lines after this point.

```
MYFILE   TEXT     A1  F 80  Trunc=80 Size=0 Line=0 Col=1 Alt=3

i20 0 * * * Top of File * * *
00001 * * * End of File * * *









====>
================================================================================
```

Figure 3-9 Create a new file with XEDIT

A few key points for users familiar with the z/OS ISPF editor include:

- The command line is at the bottom of the screen in our examples. (It can be positioned to the top of the screen, if desired.)
- Important commands (on the command line) include:
  - **file** - Save the current file and exit.
  - **qquit** - Exit without saving the current file.
  - **/xxx/** - Locate the next line containing characters xxx.
  - **+nn** - Scroll forward nn lines.
  - **-nn** - Scroll backward nn lines.
  - **top** - Scroll to the beginning of the file.
  - **bottom** - Scroll to the bottom of the file.
  - **nulls on** or **nulls off** - Use the 3270 null character function.
- Important default PF key definitions include:
  - PF2 - Insert a blank line after the line with the cursor.
  - PF3 - Quit (if no changes were made to the file); exit from the Help function.
  - PF4 - Tab to columns 5, 10, 15, and so forth.
  - PF7 - Scroll backward.
  - PF8 - Scroll forward.
  - PF9 - Split or join, depending on the cursor location.
  - PF10 - Scroll right 10 columns.
  - PF11 - Scroll left 10 columns.
  - PF12 - File.
- *Line commands* (as opposed to commands in the command line) are entered by typing over the numeric line number field at the left of each text line. (These line numbers are not part of the data record). Common line commands include:
  - **d** - Delete this line.
  - **i** - Insert a blank line after this line.
  - **"** - Duplicate this line (using the “double quotation marks” character).
  - **dnn** - Delete the indicated number of lines. This form can also be used with the **i** and **"** commands.
  - **dd** followed by **dd** in a later line - Delete a block of lines. This form can also be used with **" "** block commands to duplicate a block.

– **cc** followed by **cc** in a later line - Copy a block of lines. The target line is noted by **p** (copy prior to this line) or **f** (copy following this line). An **mm** block can be used to move lines in the same way.

3.3 System disks overview

A CP-owned disk (as are all the z/VM disks in our implementation) contains a mixture of the following:

- Paging space
- Temporary disk space (noted as TDISK in allocation displays)
- Spool space
- Directory space (noted as DRCT in allocation displays)
- Minidisks
- Unused space

You can display some of this information with the **q alloc all** and **q alloc map** commands. Typical results are shown in Figure 3-10 on page 26. Our z/VM IPL volume (address 400, volser 440RES) contains 50 cylinders for temporary disks, 134 cylinders for z/VM paging, 178 cylinders for spooled data, and 20 cylinders for active directory use. (The remainder of the volume is used mostly for z/VM system minidisks, but minidisks are not shown by the **q alloc** commands.)

```
q alloc all
DASD 0400 440RES 3390 CKD-ECKD (UNITS IN CYLINDERS)
     TDISK TOTAL=000050 INUSE=000000 AVAIL=000050
     PAGE  TOTAL=000134 INUSE=000004 AVAIL=000130
     SPOOL TOTAL=000178 INUSE=000076 AVAIL=000102
     DRCT  TOTAL=000020 INUSE=000001 AVAIL=000019, ACTIVE
DASD 0401 440W01 3390 CKD-ECKD (UNITS IN CYLINDERS)
     TDISK TOTAL=000000 INUSE=000000 AVAIL=000000
     PAGE  TOTAL=000000 INUSE=000000 AVAIL=000000
     SPOOL TOTAL=000000 INUSE=000000 AVAIL=000000
     DRCT  TOTAL=000000 INUSE=000000 AVAIL=000000
DASD 0402 440W02 3390 CKD-ECKD (UNITS IN CYLINDERS)
     TDISK TOTAL=000000 INUSE=000000 AVAIL=000000
     PAGE  TOTAL=000000 INUSE=000000 AVAIL=000000
     SPOOL TOTAL=000000 INUSE=000000 AVAIL=000000
     DRCT  TOTAL=000000 INUSE=000000 AVAIL=000000
DASD 0403 VMPAG1 3390 CKD-ECKD (UNITS IN CYLINDERS)
     TDISK TOTAL=000000 INUSE=000000 AVAIL=000000
     PAGE  TOTAL=002339 INUSE=000000 AVAIL=002339
     SPOOL TOTAL=000000 INUSE=000000 AVAIL=000000
     DRCT  TOTAL=000000 INUSE=000000 AVAIL=000000
IPL NUCLEUS ACTIVE ON VOLUME 440RES
Ready; T=0.01/0.01 20:10:19
q alloc map
            EXTENT EXTENT                        % ALLOCATION
VOLID  RDEV  START    END  TOTAL IN USE   HIGH USED TYPE
------ ---- ------ ------ ------ ------ ------ ---- -------------
440RES 0400      1     20     20      1      2   5% DRCT ACTIVE
                29     78     50      0      0   0% TDISK
                79    256  32040  12123  19626  37% SPOOL
               257    390  24120    541    571   2% PAGE
VMPAG1 0403   1000   3338 421020      0      0   0% PAGE
Ready; T=0.01/0.01 20:10:53
                                                      RUNNING   FLEXVM44
================================================================================
```

Figure 3-10 CP disk allocation

The **q alloc map** command shows where these system areas are allocated on the disk. For example, the temporary disk space on volser 440RES starts at cylinder 29 and ends in cylinder 78, for a total of 50 cylinders. In this display, the TOTAL, IN USE, and HIGH values for SPOOL and PAGE are displayed in terms of 4096 byte blocks. (There are 12 blocks per track and 15 tracks per cylinder on 3390 volumes.)

Minidisk spaces can be displayed with the **diskmap** command. This command produces an output file containing its results; a minidisk map is usually much too large to display directly on the screen. We used the command **diskmap user direct c**.[18] This produced file USER DISKMAP A (on our A disk). We then used the command **browse user diskmap a** to view the results. (The operation of **browse** is very similar to the operation of XEDIT.) The first part of the results is shown in Figure 3-11 on page 27.

[18] As discussed later, the source file for our z/VM directory is USER DIRECT C, and this file name is the operand for the **diskmap** command.

```
 USER     DISKMAP  A1 F 80            7 BLKS 03/12/08 LINE      21 OF      348
 ===>                                                                   BROWSE

VOLUME   USERID      CUU   DEVTYPE   START      END      SIZE
440RES   $ALLOC$     A01     3390    00000    00000    00001
         $DIRECT$    A04     3390    00001    00020    00020
         $SYSCKP$    A06     3390    00021    00024    00004
         $SYSWRM$    A07     3390    00025    00028    00004
         $TDISK$     B01     3390    00029    00078    00050
         $SPOOL$     B01     3390    00079    00256    00178
         $PAGE$      A03     3390    00257    00390    00134
         MAINT       CF1     3390    00391    00435    00045
         MAINT       CF2     3390    00436    00480    00045
         MAINT       CF3     3390    00481    00540    00060
         MAINT       190     3390    00541    00647    00107
         MAINT       2CC     3390    00648    00652    00005
         MAINT       191     3390    00653    00827    00175
         MAINT       193     3390    00828    00994    00167
         MAINT       19D     3390    00995    01140    00146
         MAINT       19E     3390    01141    01390    00250
         MAINT       201     3390    01391    01412    00022
         MAINT       319     3390    01413    01421    00009
         MAINT       400     3390    01422    01496    00075
         MAINT       401     3390    01497    01642    00146
         MAINT       402     3390    01643    01788    00146
         MAINT       405     3390    01789    01944    00156
         MAINT       490     3390    01945    02051    00107
         MAINT       493     3390    02052    02218    00167
         MAINT       49B     3390    02219    02418    00200
         MAINT       51D     3390    02419    02431    00013
         AUDITOR     191     3390    02432    02436    00005
         AUTOLOG1    191     3390    02437    02437    00001
================================================================================
```

Figure 3-11 Portion of DISKMAP results

User IDs such as $ALLOC$, $DIRECT$, and $PAGE$ are, in effect, dummy user IDs that are used primarily to make disk maps more readable. For example, in Figure 3-11, user ID $TDISK$ has a minidisk defined starting at cylinder 29 and extending for 50 cylinders. This matches what was shown by the **q alloc map** command. The address assigned, B01, has no meaning. A minidisk must have an address, and B01 was selected arbitrarily. This minidisk is not actually used; the allocation and control of temporary disk space is through another mechanism. However, the allocation of minidisks to match disk areas used for paging, spooling, temporary disks, and directories is helpful because it lets the **diskmap** command present a more complete picture of all the used space on a disk.

The example shown in Figure 3-11 lists some the minidisks owned by user ID MAINT. (MAINT owns many minidisks, more than shown on this screen.) These minidisks are defined by MDISK statements in the z/VM directory entry for MAINT. A portion of MAINT's directory entry in shown in Figure 3-12 on page 28.

```
 USER     DIRECT   C1  F 80  Trunc=72 Size=2004 Line=504 Col=1 Alt=0

00520  MDISK CF1 3390 391 045 440RES RR READ     WRITE    MULTIPLE
00521  MDISK CF2 3390 436 045 440RES RR READ     WRITE    MULTIPLE
00522  MDISK CF3 3390 481 060 440RES RR READ     WRITE    MULTIPLE
00523  MDISK 190 3390 541 107 440RES MR ALL      WRITE    MULTIPLE
00524  MDISK 2CC 3390 648 005 440RES MR READ     WRITE    MULTIPLE
00525  MDISK 191 3390 653 175 440RES MR READ     WRITE    MULTIPLE
00526  MDISK 123 3390 000 END 440RES MR
00527  MDISK 124 3390 000 END 440W01 MR
00528  MDISK 125 3390 000 END 440W02 MR
====>
================================================================================
```

Figure 3-12 Part of MAINT's directory

As an example, look at the MDISK line for address 2CC. This defines a minidisk area on volser 440RES, starting with cylinder 648 and extending for five cylinders. Looking at Figure 3-11 on page 27, you will see this minidisk at the appropriate position in the disk. We examine more MDISK operands in the next chapter.

3.3.1 Additional system space

As can be seen, the CP-owned z/VM disks contain paging, spooling, temporary disk, and directory space in addition to many minidisks. Are these spaces sufficient for our small z/VM system? They are sufficient for IPLing z/VM and doing minor CMS tasks. For our purposes (mostly for running z/OS and SUSE virtual machines), the directory space, the temporary disk space, and the spooling space are adequate. The paging space on the z/VM system volumes is not adequate.

The usage we describe requires little or no temporary disk space and very little spooling space. (Some CMS tools might internally use temporary disk space; we do not explicitly use any. Normal system startup uses some spooling space. A large portion of the existing spool space is used for *named systems*, such as CMS, that are also kept in the spool space.)

We created volume VMPAG1 primarily as an additional paging volume. Following common advice to keep local minidisks off the system volumes, we reserved part of the space on VMPAG1 for our local minidisks.

Efficient paging is critical to good operation of a larger z/VM system, and a larger system will have multiple paging volumes spread over multiple channels and control units. z/VM running under FLEX-*ES* on an xServer with fast RAID disks would benefit from more paging volumes. Our z/VM system is very small and can be run on a ThinkPad. No matter how many z/VM paging volumes we define (when running on a ThinkPad), the physical disk I/O to the single (relatively slow) ThinkPad disk is a bottleneck.

3.3.2 Managing minidisks and disk cylinders

Attempting to manage absolute cylinder addresses and ranges when defining minidisks can appear messy and crude. It is. Nevertheless, this is the way z/VM works. There are higher-level tools to make the task easier. DIRMAINT is the best known of these tools. However, a small z/VM system, such as our system, can be readily managed by directly editing and controlling the MDISK statements in the z/VM directory, and that is how we proceed throughout this redbook.

There are a few guidelines to make the task easier:

- Do not change the minidisk definitions (or paging, spooling, temporary disk space, or directory space) on the z/VM system volumes. These are somewhat complex, and there is no real need to change them. You do need to edit some of the z/VM control files that are contained in various system minidisks, but this does not alter the minidisk definitions.
- Create your own disk or disks for your own minidisks. Define minidisks using sequential space on these disks. We created volume VMPAG1 partly for this purpose. As previously described, we used about 70% of this disk as paging space and left 30% for our minidisks.
- Use the **diskmap** command frequently to check for *overlaps* and *gaps*.
- Remember to *never* allocate a minidisk (or anything else) on cylinder 0. The volume label (volser) is on the first track of the first cylinder and the rest of this cylinder is lost space. (This restriction does not apply to a full-pack minidisk.)

If we scroll through the **diskmap** output created earlier, we see the data in Figure 3-13.

```
 USER     DISKMAP  A1 F 80            7 BLKS 03/12/08 LINE      1 OF      348
 ===>                                                                 BROWSE

--------------------------------------------------------------------------------

VOLUME   USERID      CUU   DEVTYPE   START      END     SIZE
                                         0      500      501    GAP
$$$$$$   DATAMOVE    5F0     3380    00501    00501    00001
         DATAMOVE    5FF     3380    00502    00502    00001

--------------------------------------------------------------------------------

VOLUME   USERID      CUU   DEVTYPE   START      END     SIZE
                                         0        0        1    GAP
VMPAG1   MVSAD       191     3390    00001    00002    00002
         MVSSA       191     3390    00003    00004    00002
         SUSE1       191     3390    00005    00006    00002
         BILL        191     3390    00007    00008    00002
                                         9      999      991    GAP
         $PAGE$      A04     3390    01000    03339    02039

--------------------------------------------------------------------------------

VOLUME   USERID      CUU   DEVTYPE   START      END     SIZE
440RES   $ALLOC$     A01     3390    00000    00000    00001
         $DIRECT$    A04     3390    00001    00020    00020
         $SYSCKP$    A06     3390    00021    00024    00004
         $SYSWRM$    A07     3390    00025    00028    00004
         $TDISK$     B01     3390    00029    00078    00050
         $SPOOL$     B01     3390    00079    00256    00178
         $PAGE$      A03     3390    00257    00390    00134
         MAINT       CF1     3390    00391    00435    00045
================================================================================
```

Figure 3-13 More DISKMAP results

Consider the data for volume VMPAG1 in Figure 3-13. Notice there are two GAPs; no minidisks are allocated in cylinder 0 or in the range 9-999. If we wanted to allocate a new minidisk, we would probably start it at cylinder 9 on this disk. The $PAGE$ minidisk (cylinders 1000-3339) was defined under user ID $PAGE$. This is a dummy user, and this area is not really used as a minidisk. We defined it as z/VM paging space and made the dummy minidisk solely to make the **diskmap** command output more complete.

In this same output, you might notice that the 191 minidisks for MVSAD, MVSSA, SUSE1, and BILL are each two cylinders long. This was quite arbitrary. We use BILL for various small CMS tasks. The others are used for our z/OS and SUSE virtual machines, and there is normally very little on the CMS 191 disks for these machines. (Remember that the 191 drive is a user's CMS A disk, which is the default CMS work and storage minidisk for that user. Each user normally has their own 191 minidisk.)

This **diskmap** output does not have any overlap examples. If they existed, the word OVERLAP would appear on the right side of the listing. Overlapping minidisk allocations are normally an error.

Again, we repeat that a minidisk is created simply by adding an MDISK statement to the z/VM directory (and activating the changed directory). Multiple users can access a minidisk by using LINK statements (in their directory entries or by interactive CMS commands) pointing to the user having the MDISK statement and by specifying the address assigned in the MDISK statement.

3.4 The z/VM directory

The directory is the heart of a z/VM system. All virtual machines (that is, all users or user IDs) are defined here, as well as all minidisks.[19]

```
USER     DIRECT   C1  F 80  Trunc=72 Size=1838 Line=117 Col=1 Alt=0

00131 *
00132 USER BILL BILL 256M 256M G
00133 mach esa
00134 cpu 00 base
00135 SPOOL 000C 2540 READER *
00136 SPOOL 000D 2540 PUNCH A
00137 SPOOL 000E 1403 A
00138 CONSOLE 009 3215 T
00139 LINK MAINT 0190 0190 RR
00140 LINK MAINT 019D 019D RR
00141 LINK MAINT 019E 019E RR
00142 LINK TCPMAINT 592 592 RR
00143 MDISK 191 3390 0007 002 VMPAG1 MR
00144 *
====>
================================================================================
```

Figure 3-14 Simple user in directory

In a sense, there are two versions of the directory in z/VM: the source directory (a portion of which is shown in Figure 3-14) and the working directory. In principle, the source directory could be any file. In practice, it is file USER DIRECT on the minidisk defined for MAINT at address 2CC. When logged on to MAINT, this disk is automatically accessed as file mode C. To edit the z/VM directory source (from user ID MAINT), we can use the command **x user direct c**.

Changes to the z/VM directory are not effective until the directory source is processed into the active directory. This is done with the **directxa** command, and an example is shown in 4.5,

[19] This is not completely true because some minidisks can be defined dynamically.

"Adding a basic user" on page 37. All the directory statements are fully explained in z*/VM: CP Planning and Administration*, SC24-6043.

The z/VM 4.4.0 system discussed here has almost 100 system-supplied virtual machines (user IDs) in the directory. It is best to leave these directory definitions as they are distributed unless you have specific instructions for altering them.

3.5 TCP/IP control

We used the **ipwizard** EXEC to provide our initial TCP/IP configuration for z/VM, as described in 7.3.1, "TCP/IP customization" on page 69. We also added TCPIP to the PROFILE EXEC for the AUTOLOG1 user; this causes the user ID TCPIP to be logged on automatically when z/VM starts, and this user ID brings up TCP/IP.

To stop and restart TCP/IP while z/VM is running, we can issue the following commands from an appropriately authorized user (such as MAINT):

```
cp force tcpip                (To stop TCP/IP)
cp xautolog tcpip             (To restart TCP/IP)
```

The important customization files for TCP/IP are TCPIP DATA (on TCPMAINT 592) and PROFILE TCPIP (on TCPMAINT 198). The TCPIP DATA file contains domain and DNS information, and PROFILE TCPIP contains almost everything else. If we log on to user ID TCPMAINT (password TCPMAINT) and issue a **q disk** command, we find that 592 is already accessed as file mode F and 198 as file mode D. We can use the following commands to edit the files:

```
x profile tcpip d

x tcpip data f
```

If changes are made, it is necessary to stop and restart TCP/IP. (An alternative is to use OBEY commands, but this is more complex to set up.)

4

Common tasks

This chapter describes common administrative tasks and queries, in no particular order. As with this entire redbook, only very basic approaches and techniques are presented. A skilled z/VM user could add many additional details.

4.1 Operator session

When z/VM is IPLed, the OPERATOR user ID is automatically logged on to the primary console. The normal procedure, especially on a smaller system, is to disconnect the operator session with a **disc** command. This leaves the operator session running internally, but without a console.

Most FLEX-*ES* users have a Red Hat terminal session active (and at least partly visible on their desktop) with a `flexes>` prompt. This is the FLEX-*ES* command line interpreter (CLI) prompt. This session can also serve as a *hardware system console*, similar to the system console sessions (one per LPAR) that are available on a IBM @server zSeries® HMC. z/VM supports the hardware system console as a limited-function terminal. After disconnecting the operator session from the x3270 terminal, we can log on to it from the `flexes>` prompt. This is primarily so that we can see urgent system messages without requiring a complete x3270 terminal session. (The Red Hat desktop is probably very busy with x3270 sessions while z/VM is active, and anything to relieve this congestion is helpful.)

We can proceed as follows:

```
flexes> hwc logon operator
ENTER PASSWORD  (IT WILL NOT APPEAR WHEN TYPED)
HCPSED6013A A CP read is pending.
flexes> hwc operator                                (Or whatever your password is)
```

We will then see z/VM logon and logoff messages and similar messages. Press Enter (in the flexescli window) to recover the `flexes>` prompt. We can enter z/VM commands by using the CLI **hwc** command:

```
flexes> hwc q n
```

(The names of existing z/VM virtual machines are listed.)

We can use the **hwc disc** command to disconnect the operator virtual machine from the flexescli screen. Most operator commands can be entered from this interface. For example, the **hwc shutdown** command will terminate z/VM. Be careful with this particular example! If we enter **shutdown** instead of **hwc shutdown**, we will instantly terminate the FLEX-*ES* S/390 emulation.

4.2 Obtaining more 3270 sessions

A z/VM with several virtual machines running can require a number of 3270 sessions. These can be TN3270 clients with LAN connections to a FLEX-*ES* system. We can also open more x3270 sessions on the Red Hat desktop. (It can be convenient to use multiple desktop panels, to avoid too much crowding on a single screen.) The normal FLEX-*ES* startup that has been documented in this series of Redbooks starts two x3270 sessions. If more are needed, we do the following:

1. Move to a different Red Hat desktop panel, if appropriate.
2. Open a new terminal, using the right mouse button.
3. `$ x3270 -model 3 -keymap pc -port tn3270 localhost &`
   This command can be repeated to open additional x3270 sessions.

This results in a Terminal Solicitor connection, as shown in Figure 4-1 on page 35.

```
Welcome to the FLEX-ES Terminal Solicitor (node: t23)
 Please select (X) the desired service and press enter
 (PA1 to exit; CLEAR to refresh)

 _ L703              _ L704              _ L702



================================================================================
```

Figure 4-1 Terminal Solicitor screen

The session names in this screen, such as L703, are taken from the FLEX-*ES* definitions. The names shown are currently unused sessions. If we select one of these with an X and press Enter, we should receive a z/VM logo screen, as shown in Figure 3-4 on page 16. We can then log on to z/VM from it, or we can **dial** to a virtual machine (such as z/OS) from it.

4.3 Simple system queries

These commands are best issued from MAINT (or any other user ID with equivalent authority):

- What are the current virtual machines (users)?

  q n

  Current virtual machines (users) are listed with an indicator of the terminal associated with that user. These are commonly in four forms: DSC (for disconnected virtual machines, where no terminal is associated with it), a four-digit hexadecimal address of a local 3270, an SNA LU name, or SYSC (for the system hardware console).

- How busy is the system?

  ind

  This provides a global indication of CPU busy (percentage) and paging rates.

- What real disks are present?

  q dasd all

  This lists all the disks defined in FLEX-*ES*, including offline drives. These are “real” disks, not minidisks.

- What is the S/390 memory size?

  q stor

  This reflects the memsize parameter in the FLEX-*ES* definitions when used by an authorized user such as MAINT. For a normal user, it shows the size of the user’s virtual machine.

- What LCS (3172) LAN interfaces are available?

  q ctca all

  Remember that 3172 TCP/IP LAN devices are seen by the hardware as CTC (channel-to-channel) adapters.

- Are any tape drives available?

  q tape all

  This displays any tapes defined through FLEX-*ES*. Remember that FLEX-*ES* needs to have a tape device of some type mounted on the defined tape unit. This could be a SCSI tape drive, a FakeTape file, an AWSTAPE file, or an OMA file (on a CD). A “real” channel-attached tape drive could also be used.[20]

- What CP-owned volumes are online?

  ```
  q cpowned
  ```

- What minidisks are being used?

  ```
  q system 400                                (Query a specific real address.)
  q system                                    (Ask about all real disks.)
  ```

  This command displays the minidisk name (owner ID and address) and the current access level of every minidisk being defined or linked by any logged on user ID. For example, the output might be:

  ```
  q system 400
  DASD 0400 ATTACHED CPVOL 0122 440RES
  MAINT 05A4 R/W     , MAINT 05A2 R/W       , MAINT 04C4 R/W
  ...
  ```

  This indicates that the disk at real address 400 (volser 440RES) has 122 minidisks currently being used. The first one is MAINT's 05A4 minidisk, which is being used in read/write mode, and so forth.

4.4 Simple user queries and commands

Simple user queries and commands include:

- What CMS minidisks are currently accessible?

  ```
  q accessed
  q disk
  ```

  These two commands provide slightly different information.

- Display a snapshot of current allocations:

  ```
  q all
  ```

  This command displays more information for normal users than for MAINT. It includes the memory size of the virtual machine and all the devices (including consoles, spooled devices, and temporarily linked minidisks) currently associated with the user.

- What users are linked to a particular minidisk?

  ```
  q links 120
  ```

  This command displays the names of all the current virtual machines with links to the minidisk that is the 120 minidisk for the current user.

- What PF keys are defined?

  ```
  q pf
  set pf12 retrieve
  ```

  CMS users usually set their PF key preferences in their PROFILE EXEC A file, which is executed automatically when CMS is started.

- Release access and the link to a minidisk:

  ```
  rel z                                       (Release access to file mode z.)
  det 120                                     (Detach link to address 120.)
  ```

[20] This option was not available for Linux-based FLEX-*ES* at the time of writing.

4.5 Adding a basic user

The general process for adding a basic user is this:

1. Determine the volume and cylinder numbers to assign for any minidisks to be created for this user. Even the most basic user will probably have a 191 minidisk and two cylinders might be considered a minimum size. We normally use the **mapdisk** command output to find a gap on the appropriate volume.
2. Log on to MAINT and edit the z/VM directory source:

   ```
   x user direct c
   ```

   The PROFILE EXEC that is executed automatically when logging on to MAINT causes MAINT's 2CC minidisk to be accessed as file mode c. This minidisk contains file *user direct*, and this is the z/VM directory source file.
3. Using XEDIT, insert the definitions for a new user in the file.[21] Use existing user definitions as prototypes, but be certain that minidisk definitions are not copied blindly. An example is:

   ```
   USER BILL BILL 64M 128M G
   MACH ESA
   CPU 00
   IPL 190
   SPOOL 00C 2540 READER *
   SPOOL 00D 2540 PUNCH A
   SPOOL 00E 1403 A
   CONSOLE 009 3215 T
   LINK MAINT 0190 0190 RR
   LINK MAINT 019D 019D RR
   LINK MAINT 019E 019E RR
   LINK TCPMAINT 592 592 RR
   MDISK 191 3390 007 002 VMPAG1 MR
   ```

   This creates a new user named BILL with password BILL. The virtual machine size is initially 64 MB, which is ample for CMS. The machine mode is ESA, which can be considered normal. One virtual CPU is used by BILL. (If multiple CPUs are available to FLEX-*ES* and are used by z/VM, it might be appropriate to add a line with "CPU 01" to allow the use of two CPUs.) The SPOOL functions cannot be used by BILL, but these definitions are considered normal and should be included unless there is a specific reason not to do so.

   The IPL statement is optional. If present, it starts CMS automatically when the user logs on to the system. If not present, the user needs to enter **IPL CMS** when the system is waiting for CP input.

   The CONSOLE address (009) is arbitrary unless a specific console address is needed by something that will run in BILL's virtual machine. For example, for z/OS we often need a console at address 700, so we would make the virtual machine's console at 700 also.

   The first three LINK statements access standard CMS minidisks that are owned by MAINT. This is a normal configuration for basic CMS usage. The first LINK statement says that MAINT's 190 minidisk will be used at address 190 (in read-only) mode by this virtual machine.

   The last LINK statement is not standard, but is useful. The TCPMAINT 592 minidisk contains TCP/IP commands such as **ping**. In this example, we assume that BILL will want to use TCP/IP commands.

[21] The USER DIRECT file is simply a sequential file. You can insert the new user definitions anywhere between existing user definitions. Be careful not to insert a new user definition in the middle of an existing user definition. We usually placed our new user definitions near the beginning of the USER DIRECT file, after the dummy definitions (for $PAGE$ and similar user IDs) and before all the "built-in" z/VM virtual machine definitions.

The MDISK statement creates a minidisk (address 191) for BILL on volser VMPAG1 using two cylinders starting with cylinder 7.

4. Save the changes to the directory source (with the **file** command).
5. Run the command **directxa user direct c**. This causes the active z/VM directory to be rewritten from the source file.
6. Log on to the user ID to verify that it works. If a 191 minidisk was created, then:

```
ipl 190                              (Start CMS, if not done automatically)
format 191 a                         (Format the 191 disk)
Format will erase all files on disk A(191).  Do you wish to continue?
Enter 1 (YES) or 0 (NO).
1
Enter disk label:
Bil191                               (The disk label is arbitrary, 6 characters)
Formatting disk A
2 cylinders formatted on A(191)
x profile exec a                     (Use XEDIT to create PROFILE EXEC A)
   SET PF12 RETRIEVE                 (Enter these lines in the file)
   ACCESS 592 Z                      (Enter these lines in the file)
file                                 (Save the file and leave XEDIT)
logoff
```

When CMS is IPLed (in the user's virtual machine), it automatically accesses the 191 disk as file mode A. It will also automatically execute file PROFILE EXEC if this file exists. It will not automatically access the TCPMAINT 592 disk we included for this user. We added an access command in PROFILE EXEC. (We access the TCPMAINT 592 disk as file mode z; this is arbitrary. We also included a command to make PF12 a retrieval key; this is a common convenience.)

4.5.1 Managing users (virtual machines)

A user with appropriate authority (such as MAINT) can force another virtual machine (user) to log off or disconnect. The syntax is:

```
force <userid>                       (Forces a logoff)
force <userid> disc                  (Forces a disc command)
```

4.6 Managing spool contents

z/VM can emulate card reader, line printer, and card punch devices for users. The data involved is placed in the system spool space. Figure 3-10 on page 26 shows that 178 cylinders of spool space are available on one of the CP volumes. This figure also indicates that 76 cylinders are in use, leaving 102 cylinders of spool space free. More spool space can be added for the system. For example, we could have added additional spool space on our VMPAG1 volume.

In general, for the simple z/VM uses described in this document, additional spool space is not necessary. However, some understanding of spool space management is needed. A key point is that z/VM keeps other objects in the spool space in addition to the reader, printer, and punch files. Of the 76 cylinders used in the example, 75 are for these additional objects. This amount of space is fixed when z/VM is installed and does not grow.[22] The additional objects include named systems (such as the basic CMS code), and various other executable

[22] This is not true for more sophisticated z/VM administrators, who might add or delete objects in the spool space.

modules and tables that are normally shared by multiple virtual machines. A general idea of these objects can be obtained with the commands:

```
q sdf                                (Query system data files)
q nss                                (Query named saved systems/segments)
```

As a new z/VM administrator, we accept and do not try to change these objects.

The usage examples we discuss in this document do not create spooled reader, printer, or punch files. However, the system automatically creates a few reader and printer files. We must manage the space used by these files, and we should occasionally look at the files (especially in a new z/VM system) to see if anything important is involved. These files tend to be console logs from important user IDs (such as OPERATOR and MAINT) and startup logs from TCPIP.

One way to manage spool files is to *cold start* the system. This erases all spooled files (but does not alter the other objects stored in the spool space[23]). It is done by entering **cold** instead of **warm** in the IPL screen displayed in Figure 3-2 on page 15. This is a rather drastic way to manage spool space, although it might be practical if you are certain that you do not need to inspect the spooled files.

Spool management is easier if we work with a user ID that has operator and spool management authority. User MAINT has this authority, and the following discussion assumes we are working as MAINT. The **q rdr** command displays all the spooled reader files in the system; an example is shown in Figure 4-2 on page 40. Likewise, the **q prt** command displays all the spooled printer files in the system; an example is shown in Figure 4-3 on page 40. (The **q pun** command shows all the spooled punch files in the system, but there are normally none of these in the basic system we are discussing.)

[23] A **clean** start will erase all the objects in the spool space. Never do this unless you are certain you know how to manage the resulting system.

```
q rdr
OWNERID  FILE CLASS RECORDS  CPY HOLD USERFORM OPERFORM DEST     KEEP MSG
MAINT    0008 T CON 00000050 001 NONE STANDARD STANDARD OFF      OFF  OFF
MAINT    0007 T CON 00000095 001 NONE STANDARD STANDARD OFF      OFF  OFF
TCPMAINT 0008 T CON 00000081 001 NONE STANDARD STANDARD OFF      OFF  OFF
TCPMAINT 0007 T CON 00000029 001 NONE STANDARD STANDARD OFF      OFF  OFF
MAINT    0005 T CON 00000142 001 NONE STANDARD STANDARD OFF      OFF  OFF
TCPMAINT 0004 A PRT 00000003 001 NONE STANDARD STANDARD OFF      OFF  OFF
OPERATOR 0004 A PRT 00000003 001 NONE STANDARD STANDARD OFF      OFF  OFF
TCPMAINT 0005 T CON 00000125 001 NONE STANDARD STANDARD OFF      OFF  OFF
MAINT    0004 T CON 00000010 001 NONE STANDARD STANDARD OFF      OFF  OFF
MAINT    0006 T CON 00000030 001 NONE STANDARD STANDARD OFF      OFF  OFF
TCPMAINT 0009 T CON 00000081 001 NONE STANDARD STANDARD OFF      OFF  OFF
TCPMAINT 0001 A PRT 00000003 001 NONE STANDARD STANDARD OFF      OFF  OFF
OPERATOR 0003 A PRT 00000003 001 NONE STANDARD STANDARD OFF      OFF  OFF
TCPMAINT 0002 T CON 00000121 001 NONE STANDARD STANDARD OFF      OFF  OFF
MAINT    0003 T CON 00000014 001 NONE STANDARD STANDARD OFF      OFF  OFF
MAINT    0002 T CON 00000093 001 NONE STANDARD STANDARD OFF      OFF  OFF
MAINT    0001 T CON 00000140 001 NONE STANDARD STANDARD OFF      OFF  OFF
TCPMAINT 0010 T CON 00000078 001 NONE STANDARD STANDARD OFF      OFF  OFF
OPERATNS 0001 D SYS 00000000 001 NONE                            OFF  OFF
OPERATNS 0002 D SYS 00000000 001 NONE                            OFF  OFF
Ready; T=0.01/0.01 11:47:58

                                                        RUNNING   FLEXVM44
================================================================================
```

Figure 4-2 Results of the q rdr command

```
q prt
OWNERID  FILE CLASS RECORDS  CPY HOLD USERFORM OPERFORM DEST     KEEP MSG
OPERATOR 0005 T CON 00000104 001 NONE STANDARD STANDARD OFF      OFF  OFF
OPERATOR 0006 T CON 00000072 001 NONE STANDARD STANDARD OFF      OFF  OFF
OPERATOR 0002 T CON 00000101 001 NONE STANDARD STANDARD OFF      OFF  OFF
OPERATOR 0001 T CON 00000083 001 NONE STANDARD STANDARD OFF      OFF  OFF
OPERATOR 0007 T CON 00000080 001 NONE STANDARD STANDARD OFF      OFF  OFF
TCPIP    0013 T CON 00000083 001 NONE STANDARD STANDARD OFF      OFF  OFF
MAINT    0009 T CON 00000195 001 NONE STANDARD STANDARD OFF      OFF  OFF
Ready; T=0.01/0.01 11:48:48


                                                        RUNNING   FLEXVM44
================================================================================
```

Figure 4-3 Results of the q prt command

Key information in these listings is the owner ID (the first field) and the file number (the second field). We need this information to manage these spooled files.

4.6.1 The rdrlist command

Any user (including MAINT) can easily inspect his or her own spooled rdr files. In z/VM terminology, such files are in the user's *virtual reader*. The **rdrlist** command provides a list of all the rdr files owned by the current user. It does not include files owned by other users. The particular example shown in Figure 4-4 on page 41 does not have file names or file types for any of the reader files, but is otherwise representative. We can display the contents of any

of these reader files with the **peek** command, which is available as PF11. This is used by placing the cursor at the beginning of the line containing the file we want to inspect and pressing PF11. The resulting display can be scrolled (PF8/PF7); the **peek** command is terminated with the PF3 key.

```
MAINT    RDRLIST  A0  V 164  Trunc=164 Size=8 Line=1 Col=1 Alt=0
Cmd   Filename Filetype Class User  at Node     Hold  Records  Date      Time
_     (none)   (none)   CON T MAINT    FLEXVM44 NONE      140 12/03    13:58:46
      (none)   (none)   CON T MAINT    FLEXVM44 NONE       93 12/03    14:33:19
      (none)   (none)   CON T MAINT    FLEXVM44 NONE       14 12/03    14:49:10
      (none)   (none)   CON T MAINT    FLEXVM44 NONE       10 12/03    15:46:22
      (none)   (none)   CON T MAINT    FLEXVM44 NONE      142 12/03    16:01:44
      (none)   (none)   CON T MAINT    FLEXVM44 NONE       30 12/04    15:48:56
      (none)   (none)   CON T MAINT    FLEXVM44 NONE       95 12/04    18:00:16
      (none)   (none)   CON T MAINT    FLEXVM44 NONE       50 12/04    18:08:11

1= Help      2= Refresh  3= Quit      4= Sort(type) 5= Sort(date) 6= Sort(user)
7= Backward  8= Forward  9= Receive  10=          11= Peek      12= Cursor

====>
                                                              X E D I T  1 File
================================================================================
```

Figure 4-4 Results of the rdrlist command

```
 MAINT    RDRLIST  A0  V 164  Trunc=164 Size=8 Line=1 Col=1 Alt=8
Cmd   Filename Filetype Class User  at Node     Hold  Records  Date      Time
      (none)   (none)   has been discarded
discardnone)   (none)   CON T MAINT    FLEXVM44 NONE       93 12/03    14:33:19
=     (none)   (none)   CON T MAINT    FLEXVM44 NONE       14 12/03    14:49:10
=     (none)   (none)   CON T MAINT    FLEXVM44 NONE       10 12/03    15:46:22
      (none)   (none)   CON T MAINT    FLEXVM44 NONE      142 12/03    16:01:44
      (none)   (none)   CON T MAINT    FLEXVM44 NONE       30 12/04    15:48:56
      (none)   (none)   CON T MAINT    FLEXVM44 NONE       95 12/04    18:00:16
      (none)   (none)   CON T MAINT    FLEXVM44 NONE       50 12/04    18:08:11


1= Help      2= Refresh  3= Quit      4= Sort(type) 5= Sort(date) 6= Sort(user)
7= Backward  8= Forward  9= Receive  10=          11= Peek      12= Cursor

====>
                                                              X E D I T  1 File
================================================================================
```

Figure 4-5 Discarding rdr files

If we do not want to keep a reader file, we can use the **discard** command, as shown in Figure 4-5. This frees the spool space used by the file. We can enter multiple **discard** commands at the same time by entering an equals sign before additional file names, as shown in the figure.

A limitation of the **rdrlist** command is that it works only with rdr spool files belonging to the current user. There is no **prtlist** command, unfortunately. We can bypass this restriction by using the **transfer** command. (This is a CP command, not a CMS command.) Assuming the current user has the necessary authority, this command can transfer spool files from another user to the current user. It can also transfer prt files to rdr files. An example is shown in Figure 4-6 on page 42. In this example, we want to inspect rdr file 8 belonging to TCPMAINT.

```
Ready; T=1.32/2.74 11:53:48
q rdr
OWNERID  FILE CLASS RECORDS  CPY HOLD USERFORM OPERFORM DEST     KEEP MSG
TCPMAINT 0008 T CON 00000081 001 NONE STANDARD STANDARD OFF      OFF  OFF
TCPMAINT 0007 T CON 00000029 001 NONE STANDARD STANDARD OFF      OFF  OFF
TCPMAINT 0004 A PRT 00000003 001 NONE STANDARD STANDARD OFF      OFF  OFF
OPERATOR 0004 A PRT 00000003 001 NONE STANDARD STANDARD OFF      OFF  OFF
TCPMAINT 0005 T CON 00000125 001 NONE STANDARD STANDARD OFF      OFF  OFF
Ready; T=0.01/0.01 11:53:56



cp transfer tcpmaint rdr 8 to * rdr
                                                                RUNNING   FLEXVM44
================================================================================
```

Figure 4-6 Transferring a spooled file

The syntax we need is:

```
cp transfer tcpmaint rdr 8 to * rdr
 |    |        |      | |    |  +----> The result is a rdr file.
 |    |        |      | |    +-------> Current user (which is MAINT in the examples).
 |    |        |      |  +------------> File number from q rdr or q prt output.
 |    |        |      +---------------> Either PRT or RDR.
 |    |        +----------------------> Owner ID from q rdr or q prt.
 |    +-----------------------------> Transfer (the command).
 +----------------------------------> Might not be needed.
```

By using the **transfer** command and PF12 (the retrieve function in MAINT's default setup), a number of spooled files can be easily transferred to MAINT's reader. After doing this, we could use the **rdrlist** command, the **peek** command to inspect the files, and the **discard** command to delete unwanted files.

A transfer might be rejected for some spooled files. This can happen if the file is currently open by the owner.

4.6.2 Purge commands

After some experience, a z/VM administrator might recognize the general nature of the rdr and prt files created automatically by the system. In cases where the administrator recognizes the general nature of the files and knows that they are not needed, **purge** commands may be used. These are in various formats:

```
purge <userid> rdr <file number>              (Purge a specific rdr spool file.)
purge <userid> rdr ALL                        (Purge all user's rdr files.)
purge <userid> prt <file number>              (Purge a specific prt file.)
purge <userid> prt ALL                        (Purge all user's prt files.)
purge system rdr ALL                          (Purge all rdr files in spool.)
purge system prt ALL                          (Purge all prt files in spool.)
```

Purge is also a CP command and it may be necessary to use **cp purge...** in some cases. The <userid> operand can be an asterisk, indicating the current user. The <file number> operand is usually taken from **q rdr** or **q prt** output. Both the **transfer** and **purge** commands can be used with pun (punch) files; we did not illustrate this because punch files are seldom used.

4.7 Using the VMLINK command

A CMS user can access a minidisk owned by another user (that is, defined in the z/VM directory entry of another user). This might require knowing the passwords for the minidisk, if any are specified. Suppose we know that user JOE has minidisk 456 defined (without passwords) and we want to access it for some reason. We can do the following:

```
link joe 456 456                    (Link Joe's 456 to my address 456.)
acc 456 j                           (Access my 456 as file mode j.)
filelist * * j                      (See what files are on the minidisk.)
```

There are two assumptions involved here. What if I already have a minidisk defined at address 456?[24] What if I am already using file mode j? A typical CMS user might not have these questions, but a more intense user, after many LINKs and ACCESSes, might lose track of which disk addresses are being used and what file mode letters are being used at that moment.[25]

An easy way to avoid these questions is to use the **vmlink** command. This command selects an unused address for you and then selects an unused file mode letter. For example:

```
vmlink joe 456
DMSVML2060I JOE 456 linked as 0120 file mode z
Ready
```

With this information, we could examine the files on JOE's disk by using the **filelist * * z** command. When we are finished using JOE's minidisk, we can:

```
rel z                               (Using the file mode letter assigned by VMLINK)
det 120                             (Using the address assigned by VMLINK)
```

The **release** and **detach** commands undo the effect of the **vmlink** command. Logging off from z/VM also releases and detaches all minidisks.

4.8 Adding a paging volume

Our small z/VM system includes an additional volume 3390-3 that is 70% for paging. This provides about 1.7 GB paging space. This paging space is sufficient for most of our small uses, especially on a ThinkPad. When a larger xServer is used as the FLEX-*ES* base, more z/VM paging volumes might be appropriate both for paging space and paging performance.

Adding a paging volume involves the following steps:

1. Make a disk volume available. With FLEX-*ES*, this involves the following:
   a. Create a volume with the FLEX-*ES* **ckdfmt** command.
   b. Add the volume to the FLEX-*ES* definitions (remembering to adjust the number of units in the cu statement, if necessary).
   c. Recompile the definitions with the **cfcomp** command.
   d. Restart the FLEX-*ES* resources manager.
2. Use the z/VM **cpfmtxa** function to initialize the volume and assign a range of cylinders as PAGE.
3. Add the volume to the list of CP-owned volumes.
4. ReIPL z/VM.

[24] You could define a different address for your link, such as LINK JOE 456 789, but you then need to know that address 789 is free.

[25] Even typical CMS users might have more minidisks defined in their z/VM directory entry than they uses, with each using a virtual device address. The system administrator might do this for various reasons.

FLEX-ES steps

The FLEX-*ES* steps are:

1. Assuming that simple Linux files are used for emulated volumes and that sufficient space is available in /s390, the following command can be used:

   ```
   $ ckdfmt -n -r /s390/VMPAG2 3390-3
   ```

   The file name (VMPAG2) is arbitrary; we intend to use this name as the volser and using the same name for the Linux file name is common.

2. We can add this new volume to our FLEX-*ES* definitions. Using the Multi file (as listed in 2.3, “FLEX-ES definitions” on page 7) as a base, we modified the following sections:

   ```
   ...
   cu devad(0x400,5) path(2) resource(VM3990)      #(Increased unit count to 5)
   ...
   VM3990: cu 3990
   interface local(1)
   device(00) 3390-3 /s390/440RES           # address 400
   device(01) 3390-3 /s390/440W01           # address 401
   device(02) 3390-3 /s390/440W02           # address 402
   device(03) 3390-3 /s390/VMPAG1           # address 403
   device(04) 3390-3 /s390/VMPAG2           # address 404   <== new line
   ```

3. We then compiled the file with the **cfcomp Multi** command. This produced new Multi.rescf and Multi.syscf files. After stopping any running S/390 instance, we then restarted the resource manager using the following commands:

   ```
   # resadm -T                              (Release all running resources.)
   # resadm -x Multi.rescf                  (Start our new resource definitions.)
   ```

   If the resource manager was not already running, the command **resadm -s Multi.rescf** would be used.

4. We then left the *root* window and used our Multi.sh shell script to start a S/390 instance.

z/VM steps

The z/VM steps are:

1. We then IPLed z/VM as usual:

   ```
   flexes> ipl 400
   ```

2. We logged on to the system as MAINT and used a **q dasd all** command to verify that our new volume could be seen at address 404. It has no proper disk label yet. We then proceeded as follows:

   ```
   attach 404 system
   cpfmtxa
   ENTER FORMAT, ALLOCATE, LABEL, or QUIT:
   format
   ENTER THE VDEV TO BE PROCESSED OR QUIT:
   404
   ENTER THE CYLINDER RANGE TO BE FORMATTED ON DISK 0400 OR QUIT:
   0 end
   ENTER THE VOLUME LABEL FOR DISK 0404:
   VMPAG2
   CPFMTXA: FORMAT WILL ERASE CYLINDERS 00000-03337 ON DISK 0404
   DO YOU WANT TO CONTINUE? (YES | NO)
   yes
      (Formatting occurs.  Clear screen as needed.)
   ENTER ALLOCATION DATA
   TYPE CYLINDERS
   page 1 3338                          (There is no prompt after entering this line.)
   end
   ```

If we wanted to use some of this disk for minidisks, we would specify a limited range in the **page** statement. For example, we specified **page 1000 3338** when we created the VMPAG1 volume and later used cylinders 1 through 999 for local minidisks. Notice that cylinder 0 is not used; it contains the volume label, and any other use would overlay this. No special indication is used for minidisk space, although it is known as PERM space. That is, if a volume is formatted for z/VM (and owned by CP, as described next) and a cylinder range is not specified as PAGE, TDISK, or SPOOL, then that cylinder range is PERM and can be used for minidisks.

3. A disk must be CP-owned before it can be used for paging. The process for adding a CP-owned volume is this:

```
cprel a                                 (Release CP minidisk A.)
link * cf1 cf1 mr                       (Link first CP minidisk to MAINT as cf1.)
acc cf1 e                               (Access it as CMS disk e.)
filelist * * e                          (List the files on this disk.)
(Place X before file name SYSTEM CONFIG to edit this file.)
```

4. Scroll to the list of CP-owned volumes, as shown in Figure 4-7. Add the new volser (VMPAG2) in the next slot, overlaying the word RESERVED.

```
 SYSTEM   CONFIG   F1  F 80  Trunc=80 Size=281 Line=65 Col=1 Alt=0
00072
00073 /*****************************************************************************/
00074 /*                     CP_Owned Volume Statements                            */
00075 /*****************************************************************************/
00076
00077     CP_Owned   Slot   1  440RES
00078     CP_Owned   Slot   2  440W01
00079     CP_Owned   Slot   3  440W02
00080     CP_Owned   Slot   4  VMPAG1
00081     CP_Owned   Slot   5  RESERVED     <== change this line
00082     CP_Owned   Slot   6  RESERVED
00083     CP_Owned   Slot   7  RESERVED
 ====>
================================================================================
```

Figure 4-7 CP-owned volumes

5. Use the **file** command and PF3 to save the change and exit from XEDIT. Then continue:

```
acc 193 g                               (Access tools on the MAINT 193 disk.)
cpsyntax system config e                (Check the file we just edited.)
  (Verify no errors are indicated)
rel e                                   (Release our drive e.)
det cf1                                 (Detach address cf1.)
cpaccess maint cf1 a                    (Tell CP that it again has this disk.)
```

6. z/VM then needs to be reIPLed to start using the new paging space.

4.9 Logo and system ID

The *logo* is the content of a 3270 screen before logging on to z/VM. Most installations like to customize this with a semi-graphic design. The *system ID* is the character string displayed on the lower-right corner of many z/VM screens. These are both set on the same minidisk used to add CP-owned volumes to the z/VM system. This system minidisk is altered using a special technique.

To access this disk:

```
cprel a                          (Release CP minidisk A.)
link * cf1 cf1 mr                (Link first CP minidisk to MAINT as cf1.)
acc cf1 e                        (Access it as CMS disk e.)
filelist * * e                   (List the files on this disk.)
```

Logo screen

In the filelist find LOCAL LOGO E and edit it. We suggest using a maximum of 15 lines. Whatever you place in this file will appear on the z/VM logo screen, above the area for the user ID and password.

System ID

In the filelist find SYSTEM CONFIG E and edit it. (This is the same file used to add CP-owned volumes and other key information.) Scroll through this file and find lines such as the following:

```
/*********************************************************************/
/*            System_Identifier Information                          */
/*********************************************************************/
      System_Identifier_Default FLEXVM44
```

This constant, shown here as FLEXVM44, can be up to eight characters.

Saving the changes

To save changes to this disk, do the following. (The first two steps are needed only when SYSTEM CONFIG was changed).

```
acc 193 g                        (Access tools on the MAINT 193 disk.)
cpsyntax system config e         (If we changed SYSTEM CONFIG.)
  (Verify no errors are indicated)
rel e                            (Release our drive e.)
det cf1                          (Detach address cf1.)
cpaccess maint cf1 a             (Tell CP that it again has this disk.)
```

The **cpsyntax** command takes several seconds to run. The file mode letters e and g used here are arbitrary and can be any unused CMS file mode letters.

4.10 Changing the IP address

We set our initial IP address using **ipwizard**. It can be changed by logging on to z/VM as TCPMAINT and issuing the **x profile tcpip d** command. (The file PROFILE TCPIP is on TCPMAINT's 198 minidisk. The PROFILE EXEC automatically accesses this minidisk as file mode d.) This file contains statements such as:

```
HOME
  192.168.9.113 ETH1
```

To change the IP address of z/VM, simply change this statement and save (using a **file** command) the changes. Then, issue the following commands from MAINT:

```
force tcpip
xautolog tcpip
```

Any existing z/VM TCP/IP sessions will be disrupted.

4.11 Guest IP addresses

The FLEX-*ES* and z/VM configurations we use provide a separate (emulated) 3172 LAN interface for each virtual machine. The appropriate 3172 (using two addresses for each 3172) is dedicated to our MVSAD and SUSE1 virtual machines. IP addresses for the virtual machines are set by parameters within each operating system. In z/OS, for example, this is in the TCPIP.PROFILE.TCPIP data set.

In addition, the FLEX-*ES* definitions for the 3172s contain the relevant IP address. This IP address in the 3172 definitions is optional. It acts as a filter so that irrelevant IP traffic is not seen by the emulated 3172.[26] Changing a guest operating system's IP address requires changing the relevant parameters in that operating system and changing the FLEX-*ES* definitions.

A more sophisticated z/VM networking setup, using a *guest LAN*, for example, is quite different and is not described in this document.

[26] This is a performance issue. If an emulated 3172 sees all the IP packets in the network, the associated TCP/IP stack (in z/OS, for example) must inspect each packet and reject those not intended for this TCP/IP stack.

5

Running z/OS

We included two user IDs (virtual machine definitions) for z/OS. One is for the full AD CD-ROM system and the other is for the single-volume z/OS that is shipped with the AD system. Both can be run at the same time.[27] A third user ID is needed to own the z/OS volumes. This is required if the volumes are to be shared (such as when both z/OS systems are running at the same time).

The z/VM directory entry for the dummy user is shown in Figure 5-1.

```
00110 ********************************
00111 *
00112 USER MVSDUMMY NOLOG 256M 512M G
00113 MDISK A80 3390 DEVNO A80 MWV
00114 MDISK A81 3390 DEVNO A81 MWV
00115 MDISK A82 3390 DEVNO A82 MWV           (The numbers on the left side
00116 MDISK A83 3390 DEVNO A83 MWV             of each line are XEDIT line
00117 MDISK A84 3390 DEVNO A84 MWV             numbers and are not part of
00118 MDISK A85 3390 DEVNO A85 MWV             the directory entry.)
00119 MDISK A86 3390 DEVNO A86 MWV
00120 MDISK A87 3390 DEVNO A87 MWV
00121 MDISK A88 3390 DEVNO A88 MWV
00122 MDISK A89 3390 DEVNO A89 MWV
00123 MDISK A8A 3390 DEVNO A8A MWV
00124 MDISK A8B 3390 DEVNO A8B MWV
00125 MDISK A8C 3390 DEVNO A8C MWV
00126 MDISK A8D 3390 DEVNO A8D MWV
00127 MDISK A8E 3390 DEVNO A8E MWV
00128 MDISK A8F 3390 DEVNO A8F MWV
00129 *
```

Figure 5-1 Dummy user to own z/OS volumes

It is not possible to log on to this user ID. (Among other factors, the NOLOG in the user definition prevents this.) The format of the MDISK definitions used here is for full-pack minidisks. These point to volumes that are exactly the same as used for z/OS without z/VM.

[27] This is marginal on a ThinkPad. If you try it, IPL one system and let the I/O activity complete before IPLing the other system. Disk I/O is the primary limitation on a ThinkPad, and there is no point in stressing it beyond reason.

Using the FLEX-*ES* definitions described in 2.3.1, “Dual-use FLEX-ES definitions” on page 10, we can IPL z/OS directly using these same volumes. The following directory statement says that this virtual machine’s A80 device is a real 3390 at real address (as defined by FLEX-*ES*) A80:

```
MDISK A80 3390 DEVNO A80 MWV
```

We could have different virtual and real addresses, but there is no reason to do this here, and it would only confuse the situation. The MWV operand indicates this is a multi-user read/write volume and that channel **reserve** and **release** commands are to be emulated.

The z/VM directory definitions for our full z/OS system virtual machine are shown in Figure 5-2.

```
00146 *
00147 USER MVSAD MVSAD 256M 512M G
00148 mach esa
00149 cpu 00
00151 SPOOL 000C 2540 READER *
00152 SPOOL 000D 2540 PUNCH A
00153 SPOOL 000E 1403 A
00154 CONSOLE 700 3215 T
00155 LINK MAINT 0190 0190 RR
00156 LINK MAINT 019D 019D RR
00157 LINK MAINT 019E 019E RR
00158 LINK TCPMAINT 592 592 RR
00159 DEDICATE E20 E20
00160 DEDICATE E21 E21
00161 SPECIAL 701 3270
00162 SPECIAL 702 3270
00163 SPECIAL 703 3270
00164 SPECIAL 704 3270
00165 SPECIAL 705 3270
00166 MDISK 191 3390 0011 001 430W02 MR
00167 LINK  MVSDUMMY A80 A80 MW
00168 LINK  MVSDUMMY A81 A81 MW
00169 LINK  MVSDUMMY A82 A82 MW
00170 LINK  MVSDUMMY A83 A83 MW
00171 LINK  MVSDUMMY A84 A84 MW
00172 LINK  MVSDUMMY A85 A85 MW
00173 LINK  MVSDUMMY A86 A86 MW
00174 LINK  MVSDUMMY A87 A87 MW
00175 LINK  MVSDUMMY A88 A88 MW
00176 LINK  MVSDUMMY A89 A89 MW
00177 LINK  MVSDUMMY A8A A8A MW
00178 LINK  MVSDUMMY A8B A8B MW
00179 LINK  MVSDUMMY A8C A8C MW
00180 LINK  MVSDUMMY A8D A8D MW
00181 LINK  MVSDUMMY A8E A8E MW
00182 LINK  MVSDUMMY A8F A8F MW
00183 *
```

Figure 5-2 User MVSAD for full z/OS AD CD-ROM system

The purpose of the directory definition statements are as follows:

- The USER statement provides the virtual machine name (MVSAD), indicates it is to have 256 MB main storage (expandable by user command to 512 MB), and is to have only G class z/VM command authority. (This is the authority of a routine z/VM user and is all that is needed.)
- The MACH ESA statement indicates the mode of the emulated machine. The CPU statement is optional when only a single processor is involved.
- The SPOOL, CONSOLE, and first set of LINK statements are normal for a CMS user. (The LINK TCPMAINT 592 statement allows the use of TCP/IP functions while in CMS.) We do not require this user to have CMS capabilities, but it does no harm and might be useful in some circumstances. z/OS running in this virtual machine does not normally use any of these definitions. The CONSOLE address is set to 700, because we will use the same terminal address as the MVS console, and this needs to be address 700 for the z/OS system we used.
- The two DEDICATE statements indicate that real devices E20 and E21 (as defined by FLEX-*ES*) are to be dedicated to this virtual machine (when it is logged on) at virtual addresses E20 and E21. Real device E20/E21 is an IBM 3172 LAN control unit, as defined in the FLEX-*ES* definitions. This provides TCP/IP connectivity for z/OS.
- The SPECIAL statements define virtual local 3270 terminals at the indicated virtual addresses. These appear to z/OS as local, channel-attached 3270s. A user with access to a z/VM 3270 can use a **dial mvsad** command to connect to one of these virtual 3270s. Our z/OS has 3270s at these addresses connected to VTAM®, and a user will receive the VTAM logo screen when the user dials one of these terminals.
- The MDISK statement provides the A minidisk for CMS. Again, we do not require CMS for this user ID, but it might be useful in some circumstances.
- The remaining LINK statements provide access to the disk drives defined by the MVSDUMMY user ID. The drive defined as address A80 in MVSDUMMY is also addressed as A80 by this user, and so forth. The MW operand indicates the drive is used by multiple users in read/write mode.

The directory definition for the single-volume z/OS user ID is shown in Figure 5-3 on page 52. It is almost the same as for the full-system z/OS user ID except that no 3172 device is provided (because the single-volume z/OS provided with the AD system does not support TCP/IP operation) and fewer SPECIAL terminals are provided (because this z/OS is not intended for general usage). Notice that the minidisk at address 191 uses a different cylinder on the VMPAG1 volume.

```
00185 *
00186 USER MVSSA MVSSA 256M 512M G
00187 mach esa
00188 cpu 00
00189 SPOOL 000C 2540 READER *
00190 SPOOL 000D 2540 PUNCH A
00191 SPOOL 000E 1403 A
00192 CONSOLE 700 3215 T
00193 LINK MAINT 0190 0190 RR
00194 LINK MAINT 019D 019D RR
00195 LINK MAINT 019E 019E RR
00196 LINK TCPMAINT 592 592 RR
00197 SPECIAL 701 3270
00198 SPECIAL 702 3270
00199 MDISK 191 3390 0012 001 430W02 MR
00200 LINK  MVSDUMMY A80 A80 MW
00201 LINK  MVSDUMMY A81 A81 MW
00202 LINK  MVSDUMMY A82 A82 MW
00203 LINK  MVSDUMMY A83 A83 MW
00204 LINK  MVSDUMMY A84 A84 MW
00205 LINK  MVSDUMMY A85 A85 MW
00206 LINK  MVSDUMMY A86 A86 MW
00207 LINK  MVSDUMMY A87 A87 MW
00208 LINK  MVSDUMMY A88 A88 MW
00209 LINK  MVSDUMMY A89 A89 MW
00210 LINK  MVSDUMMY A8A A8A MW
00211 LINK  MVSDUMMY A8B A8B MW
00212 LINK  MVSDUMMY A8C A8C MW
00213 LINK  MVSDUMMY A8D A8D MW
00214 LINK  MVSDUMMY A8E A8E MW
00215 LINK  MVSDUMMY A8F A8F MW
00216 *
```

Figure 5-3 User MVSSA for single-pack z/OS system

The process for running the full z/OS system under z/VM is as follows:

1. Log on to z/VM with user ID MVSAD.

   (If you have any need to use CMS, enter an **i cms** command and perform whatever CMS functions you need.)

2. Enter the commands:

   ```
   CP TERM CONMODE 3270                    (There is no prompt returned.)
   CP IPL A80 LOADPARM 08A2CS              (There might be a substantial pause.)
   ```

   If CMS is running when the TERM command is issued, it will crash. This does no harm.

3. After a few seconds, z/VM might display a HOLDING indicator. Clear the screen, and you should see the first z/OS message. Your IPL address and LOADPARM might differ from that in the example, of course, depending on the particular needs of your z/OS operation. An example of an MVS operator display is shown in Figure 5-4 on page 54.
4. The first time you respond to a z/OS message (or try to enter a z/OS command), the terminal might revert to CP mode with the message:

   ```
   HCPGSP2627I The virtual machine is placed in CP mode due to a SIGP initial CPU reset
   from CPU00.
   ```

   Clear the screen to continue z/OS operation.[28]

[28] This happened with our z/OS 4.3 system, but has not happened with our z/VM 4.4 system.

5. From another z/VM terminal (on the Red Hat desktop or through a LAN attachment), obtain a z/VM logon screen and enter **`dial mvsad`** in the command line at the bottom of the screen. This should give you the z/OS VTAM logo screen, and you can log on to TSO from this.

   The **`dial`** command connects the terminal to the first unused SPECIAL device for the virtual machine named in the **`dial`** command. In our example, the first SPECIAL device is connected to address 701 for z/OS. z/OS VTAM owns this address and presents the VTAM logo screen when it detects a new terminal connection.

6. After logging off from TSO, the z/OS VTAM logo screen is presented. To return to the z/VM logo, it is easiest to disconnect the 3270 session and then reconnect. When doing this, you see the FLEX-*ES* Terminal Solicitor screen; select any one of the named terminals to connect (through the FLEX-*ES* Terminal Solicitor) to z/VM. (The names on the terminal solicitor screen correspond to the 3270 terminal names in the FLEX-*ES* definitions.)

   When using the x3270 emulator, the disconnect function is a selection under the File item on the toolbar. When the disconnect is complete, a Connect item appears on the toolbar. Other 3270 emulators have various methods for disconnecting and reconnecting the sessions.

If z/OS TCP/IP is configured properly, all the normal TCP/IP functions should work. In our example, the z/OS TCP/IP address is 192.168.0.112. This IP address is in the z/OS TCP/IP PROFILE data set. It is also in the FLEX-*ES* definitions, where it simply acts as a filter to prevent unwanted packets from being sent to z/OS.

```
          NONE
    - 17.01.32           d m=cpu
      17.01.33           IEE174I 17.01.32 DISPLAY M 646                        C
      PROCESSOR STATUS
      ID  CPU                  SERIAL
      0    +                   00F02B1245

      CPC SI = 1245.L01.FSI.FREM.000000000000F02B

      + ONLINE    - OFFLINE    . DOES NOT EXIST    W WLM-MANAGED
      N NOT AVAILABLE

      CPC SI  SYSTEM INFORMATION FROM STSI INSTRUCTION
    - 17.13.52 STC00029  IEF404I BPXAS - ENDED - TIME=17.13.52
    - 17.13.52 STC00031  IEF404I BPXAS - ENDED - TIME=17.13.52
    - 17.13.52 STC00025  IEF404I BPXAS - ENDED - TIME=17.13.52
    - 17.13.53 STC00030  IEF404I BPXAS - ENDED - TIME=17.13.53
    - 15.11.57           d a,l
   00  15.11.58           IEE114I 15.11.58 2003.339 ACTIVITY 671                C
       JOBS     M/S    TS USERS    SYSAS    INITS   ACTIVE/MAX VTAM     OAS
      00002    00012    00000      00029    00012    00000/00040       00008
       LLA      LLA      LLA      NSW  S  JES2     JES2     IEFPROC  NSW  S
       VLF      VLF      VLF      NSW  S  VTAM     VTAM     VTAM     NSW  S
       DLF      DLF      DLF      NSW  S  RACF     RACF     RACF     NSW  S
       TSO      TSO      STEP1    OWT  S  SDSF     SDSF     SDSF     NSW  S
       TCPIP    TCPIP    TCPIP    NSW  SO HTTPD1   HTTPD1   WEBSRV1  IN   SO
       NFSS     NFSS     GFSAMAIN NSW  SO PORTMAP  PORTMAP  PMAP     OWT  SO
       FTPD1    STEP1    FTPD     OWT  AO INETD4   STEP1    OMVSKERN OWT  AO
    IEE612I CN=01       DEVNUM=0700 SYS=P390


    IEE163I MODE= RD
   ================================================================================
```

Figure 5-4 z/OS operator console

5.1 Single-volume z/OS

Using our setup, we can log on to MVSSA, the stand-alone z/OS system, and IPL it. We can do this while MVSAD is running if we want. Two z/OS systems running on a ThinkPad do not provide much performance, but can work well on a larger xServer. The procedure for using MVSSA is exactly the same as for MVSAD except that the z/VM user ID is MVSSA. The IPL address for this system is A8C, and the LOADPARM is 0A8CSA. (Again, your setup might be different.)

5.2 Remote users

A remote user (connected through TCP/IP to the FLEX-*ES* machine) could use the following steps to connect to z/VM and a virtual machine:

1. Select a TN3270 emulator. This could be x3270, or PCOM, or some other emulator.

2. Using the TN3270 emulator, connect to the IP address of Red Hat Linux (on the machine running FLEX-*ES*), port 24. For example, using X3270, the command might be the following, assuming the address of Red Hat on the FLEX-*ES* machine is 192.168.0.110:

```
$ x3270 -model 3 -keymap pc -port tn3270 192.168.0.110 &
```

3. This should connect to the FLEX-*ES* Terminal Solicitor that should list all the currently unused 3270 sessions. These are the sessions defined with the 3174 control unit in the FLEX-*ES* definitions.
4. Select any session (because z/VM owns all these terminals). This should provide the z/VM logo/logon screen.
5. If you want to connect to a running virtual machine, issue a **`dial xxxx`** command on the z/VM logo screen, where xxxx is the name of the running virtual machine. This only works if the operating system running in the virtual machine is expecting dynamic 3270 connections.

6

Running Linux

We had a SUSE LINUX for S/390 system installed on a single 3390-3. This system was installed earlier and used natively (that is, without z/VM). The characteristics and installation of this SUSE system are briefly described in Chapter 9 in the redbook *EFS Systems on a Linux Base: Additional Topics*, SG24-7008. When run natively, this SUSE system does not use 3270 terminals. The IPL process uses the *hardware console* of a S/390; in a FLEX-*ES* system, this means that it uses the window with the `flexes>` prompt. Commands to the SUSE LINUX system must follow an **`hwc`** command on the FLEX-*ES* console. Terminal usage through this interface is limited and vi, for example, cannot be used.

After the SUSE LINUX is up, users normally log in to the system through telnet sessions, where full terminal functions are available. A user can export a DISPLAY definition and use X window graphics with the system.

When used under z/VM, SUSE LINUX for S/390 can use the z/VM 3270 terminal in a restricted 3215 mode. This is more convenient than the hardware console interface through the flexescli program, but still has restricted terminal functions and cannot run vi. Normal usage, under z/VM, still involves telnet sessions for full terminal functions.

6.1 Virtual machine definitions

The z/VM directory definitions we used for our SUSE LINUX system are as follows:

```
*
USER SUSE1 SUSE1 128M 256M G
mach esa
cpu 00
SPOOL 000C 2540 READER *
SPOOL 000D 2540 PUNCH A
SPOOL 000E 1403 A
CONSOLE 270 3215 T
LINK MAINT 0190 0190 RR
LINK MAINT 019D 019D RR
LINK MAINT 019E 019E RR
LINK TCPMAINT 592 592 RR
DEDICATE 2E0 2E0
DEDICATE 2E1 2E1
MDISK 191 3390 005 002 VMPAG1 MR
MDISK 200 3390 DEVNO 200 MWV
MDISK 201 3390 DEVNO 201 MWV
MDISK 202 3390 DEVNO 202 MWV
MDISK 203 3390 DEVNO 203 MWV
*
```

By default, the S/390 virtual machine size will be 128 MB. The user can expand it to a maximum of 256 MB. A single CPU is defined. Minimal CMS definitions are included, with a two-cylinder 191 minidisk. CMS is not needed to run the SUSE system, but it is included on general principles. A single LCS 3172 device is defined at real address 2E0/2E1 (as established by the FLEX-*ES* definitions) and at virtual address 2E0/2E1 for the SUSE system. This provides a TCP/IP interface for the SUSE system.

Four whole-volume disks are defined with the same real and virtual addresses (200-203). Our SUSE system was resident on the first disk and the others were unused. (The other three volumes are OFFLINE in our FLEX-*ES* definitions.) We include them in the virtual machine definition so that the SUSE system could be expanded or cloned onto the three extra disks.

If the SUSE system is cloned onto one or more of the extra disks, we would create a new virtual machine (probably named SUSE2) and divide the four 3390 volumes between the two virtual machines.

Note that the FLEX-*ES* Multi definitions have a second LCS device defined for Linux, at addresses 2E2/2E3. This device is not used in our basic system setup.

6.2 Running the system

To IPL and use the SUSE system, we first did the following:

```
flexes> iodelay 200 20
```

This is needed because our SUSE system did not have the fix installed for a problem with the dasd driver. This is described in the *Additional Topics* book. We logged on to the SUSE1 z/VM user ID and entered the following command when z/VM waited for a CP command:

```
ipl 200
```

After the Linux IPL had recognized the disks, we removed the iodelay:

```
flexes> iodelay 200 0
```

These **iodelay** commands are not needed if fixes for the dasd module (available in early 2003) have been applied to the Linux system. The z/VM screen needs to be cleared a number of times as the Linux IPL progresses. It eventually issued a login prompt:

```
t23 login:
```

The “t23” is the name we gave the SUSE system when we installed it. Typical screen content is shown in Figure 6-1. Again, note that the 3270 screen is not a full-function terminal for Linux and the common Linux editors cannot be used with it.

```
t23:/ # ifconfig
ifconfig
eth0      Link encap:Ethernet  HWaddr 00:D0:59:CD:58:B6
          inet addr:192.168.0.112  Bcast:192.168.0.255  Mask:255.255.255.0
          inet6 addr: fe80::2d0:59ff:fecd:58b6/10 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1492  Metric:1
          RX packets:48 errors:0 dropped:0 overruns:0 frame:0
          TX packets:45 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:100
          RX bytes:3592 (3.5 Kb)  TX bytes:3541 (3.4 Kb)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:6 errors:0 dropped:0 overruns:0 frame:0
          TX packets:6 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:450 (450.0 b)  TX bytes:450 (450.0 b)

t23:/ # ls /
ls /
.   bin   dev  home  lost+found  opt   root  srv  usr
..  boot  etc  lib   mnt         proc  sbin  tmp  var
t23:/ #




                                                          RUNNING   FLEXVM44
================================================================================
```

Figure 6-1 z/VM 3270 terminal connected to SUSE LINUX for S/390

At this point, we normally opened a new terminal window on the Red Hat desktop and issued the command:

```
$ telnet 192.168.0.112                          (The IP address for SUSE)
Trying 192.168.0.112...
Connected to 192.168.0.112.
Escape character is '^]'.
Welcome to SuSE SLES 8 (s390) - Kernel 2.4.19-3suse-SMP(0)

t23 login: ogden
Password: xxxxxx
```

If we want a graphics interface, we could do the following:

```
(On the Red Hat desktop) # xhost +              (Allow X window connections.)
(On the SUSE terminal)
$ export DISPLAY=192.168.0.110:0                (Export display to Red Hat.)
$ su                                            (Change to root to use yast2.)
# yast2
```

After a few seconds, a new window should open on the Red Hat desktop, and you should see the YaST Control Center initial screen.

The system is configured to reject a **telnet** logon for *root*. When installing our initial SUSE system, we added user ID *ogden*. Once logged in to the system, we can **su** to *root*. Similar connections can be made from other systems working on the same LAN as the FLEX-*ES* system.

6.2.1 Shutting down SUSE

The SUSE system can be shut down in the normal way. Typically, this is with a **shutdown now** command issued from a `root` prompt. This causes the usual Linux shutdown messages to be displayed on the 3270 terminal. The last message is usually `Give root password to login:`, and this indicates that the system is in single-user mode. At this point, a PA1 interrupt (which is left Alt + 1 on many x3270 terminal keyboards) can be used to obtain a CP prompt. This can be followed with the **logoff** command to end the z/VM session.

6.3 LAN interface change

The SUSE system ran without alterations under z/VM, as described in this chapter. However, we needed to make a minor change to run it natively (without z/VM) using the Multi definitions for FLEX-*ES*.

When we initially installed SUSE LINUX for S/390, we selected *LCS* and *auto* for LAN interface detection during the initial installation. This causes the installation process to automatically detect all LCS[29] interfaces on the installing system. When first installing the SUSE release, we had a simple set of FLEX-*ES* definitions with a single LCS interface at addresses 2E0/2E1.

The Multi FLEX-*ES* definitions have several LCS interfaces defined. When running under z/VM, the virtual machine used for SUSE dedicates the appropriate LCS interface to the SUSE LINUX system, and SUSE LINUX detects only a single LCS interface at the device addresses we want to use for this LINUX. However, when not running under z/VM, the SUSE LINUX system detects all the LCS devices defined in the Multi definitions and uses the first LCS device for its primary interface. This is not what we wanted and conflicts with the IP address filters contained in the 3172 device FLEX-*ES* definitions.

We made a single alteration to the SUSE LINUX for S/390 system so that it looks for its LCS device at a specific address (actually an address pair), instead of automatically detecting all existing LCS devices. With this change, our SUSE system works natively with the Multi definitions, as well as under z/VM with the Multi definitions. We added the following lines at the end of the /etc/chandev.conf file in SUSE LINUX:

```
noauto
lcs0,0x2E0,0x2E1,0,0
```

[29] LCS means LAN Channel Station and is a generic type of LAN interface. It requires two addresses (device numbers) in an even/odd pair. Our FLEX-*ES* definitions use 3172 devices as LCS devices.

This causes Linux to use only the LCS device defined at device numbers 2E0/E21. The Multi FLEX-*ES* definitions have the proper IP address filter here for the IP address we use for the SUSE system (which is IP address 192.168.0.112).

7

z/VM installation

We ordered z/VM 4.4.0 on CD OMA media and used this to build the z/VM system described in this redbook. This book is intended to discuss basic use of the installed z/VM system under FLEX-*ES*, and this chapter provides only a brief overview of the z/VM installation process we used.

z/VM 4.4.0 was delivered with six CDs. Four contained the distributed system, one contained service, and one contained optional features.

7.1 FLEX-ES preparations

The z/VM documentation indicated that three 3390-3 volumes were needed for installation. We planned to add a fourth volume for local minidisks and additional paging space. The standard volume names are 440RES, 440W01, and 440W02. We arbitrarily named our additional volume VMPAG1.

Our first step was to create four empty 3390-3 volumes for use with FLEX-*ES*. The commands (issued from a Red Hat terminal window) were:

```
$ ckdfmt -n -r /s390/440RES 3390-3
$ ckdfmt -n -r /s390/440W01 3390-3
$ ckdfmt -n -r /s390/440W02 3390-3
$ ckdfmt -n -r /s390/VMPAG1 3390-3
```

Each of these commands took about 10 minutes and resulted in an empty 3390 volume.

We next integrated these volumes into the FLEX-*ES* definitions described in 2.3, “FLEX-ES definitions” on page 7. These definitions were in the /usr/flexes/rundir/Multi file, and this is referred to as the Multi file or Multi definitions. Our system already had several z/OS volumes and Linux for S/390 volumes installed, and these are reflected in the Multi definitions.

Our goal was to use a single set of FLEX-*ES* definitions (in the Multi file) to run z/OS or Linux in stand-alone mode or under z/VM. We selected device addresses for z/VM (and for Linux for S/390) that did not conflict with z/OS IODF definitions.

7.2 Basic z/VM installation

Our z/VM installation followed the instructions in *z/VM: Guide for Automated Installation and Service, Version 4 Release 4.0*, GC24-6064. You should consult this document for background information.

7.2.1 Initializing the disks

We started FLEX-*ES*, using the Multi definitions, and mounted the first z/VM CD as an OMA volume and IPLed from it:

```
# resadm -s Multi.rescf                          (Start the FLEX-ES resource manager.)
$ . Multi.sh                                     (Shell script to start FLEX-ES instance.)
flexes> mount 460 /mnt/cdrom:/mnt/cdrom/tapes/uaa932.tdf
flexes> ipl 460
```

The background and use of the **resadm** command and shell scripts for starting FLEX-*ES* is described in the *Getting Started* redbook. We include them here (without additional explanation) for completeness. Note that the # prompt indicates a window running as *root* and the $ prompt indicates a window running as user ID *flexes*.

This IPL loaded the ICKDSF program that is used to initialize disk volumes for z/VM use. The format of the FLEX-*ES* OMA/2 **mount** command might appear odd. Use it exactly as shown here; there are no spaces before or after the colon in the command.

To use ICKDSF, we switched to a 3270 window and pressed Enter. This produced the message `Clear Screen When Ready`. With an x3270 emulator using an unmodified keymap, we cleared the screen with the Alt-c key combination, using the left-side Alt key. We proceeded with an ICKDSF dialog and cleared the screen whenever necessary:

```
ICKOO5E DEFINE INPUT DEVICE, REPLY'DDDD,CUU' or 'CONSOLE'
Enter                                   (Press the Enter key.)
ICKOO6E DEFINE OUTPUT DEVICE, REPLY 'DDDD,CUU' or 'CONSOLE'
Enter
-----DEVICE SUPPORT FACILITY 16.0X-----
ENTER INPUT/COMMAND:
cpvolume format unit(400) novfy volid(440RES) mode(ESA) nofiller
ICKOO3D REPLY U TO ALTER VOLUME 0400 CONTENTS, ELSE T
U
....takes about 10 minutes.  Clear screen as necessary for messages....
cpvolume format unit(401) novfy volid(440W01) mode(ESA) nofiller
U
...
cpvolume format unit(402) novfy volid(440W02) mode(ESA) nofiller
U
...
cpvolume format unit(403) novfy volid(VMPAG1) mode(ESA) nofiller
U
```

At this point, the four 3390 volumes were initialized in a z/VM format. We IPLed the OMA/2 “tape” again to load the DDR restore program:

```
flexes> ipl 460
```

We waited about 10 seconds and pressed Enter while on the 3270 screen. (If the 3270 keyboard is locked, the keyboard reset sequence for an unmodified x3270 is Alt-r.)

```
Enter                                   (Press the 3270 Enter key.)
z/VM DASD Dump/Restore Program
......
ENTER CARD READER ADDRESS OR CONTROL STATEMENTS
ENTER:
sysprint cons
ENTER:
input 500 tape (skip 1 rew
ENTER:
output 400 dasd 440RES
ENTER:
restore all
   (This takes about one minute and restores three extents on the disk.)
HOLDING
Enter                                   (Press the 3270 Enter key.)
ENTER:
Enter                                   (Press the 3270 Enter key.)
END OF JOB
```

7.2.2 z/VM installation

We next IPLed the z/VM volume. We included the address of the 3270 console as the IPL parameter:

```
flexes> ipl 400 700
```

This produced the Stand-Alone Loader screen for the Initial Installation System (IIS) for z/VM. We entered **cons=700**, as shown in Figure 7-1 on page 66, to indicate our console address

and pressed PF10 to load the system. This takes about a minute, during which disk activity can be seen. After it is loaded, the system produces the screen shown in Figure 7-2.

```
  STAND ALONE PROGRAM LOADER: z/VM VERSION 4 RELEASE 4.0

  DEVICE NUMBER:   0400       MINIDISK OFFSET:   00000000   EXTENT:  1

  MODULE NAME:     CPLOAD     LOAD ORIGIN:       1000

  --------------------------------IPL PARAMETERS--------------------------------
cons=700


  ----------------------------------COMMENTS----------------------------------




  --------------------------------------------------------------------------------



  9= FILELIST  10= LOAD  11= TOGGLE EXTENT/OFFSET
```

Figure 7-1 Stand Alone Program Loader

```
14:12:37 z/VM  V4 R4.0  SERVICE LEVEL 0000 (64-BIT)
14:12:40 SYSTEM NUCLEUS CREATED ON 2003-06-18 AT 08:29:04, LOADED FROM 440RES
14:12:40
14:12:41 ************************************************************
14:12:41 * LICENSED MATERIALS - PROPERTY OF IBM*                      *
14:12:41 *                                                           *
14:12:41 * 5739-A03 (C) COPYRIGHT IBM CORP. 1983, 2003. ALL RIGHTS    *
14:12:41 * RESERVED. US GOVERNMENT USERS RESTRICTED RIGHTS - USE,     *
14:12:42 * DUPLICATION OR DISCLOSURE RESTRICTED BY GSA ADP SCHEDULE   *
14:12:42 * CONTRACT WITH IBM CORP.                                    *
14:12:42 *                                                           *
14:12:42 * * TRADEMARK OF INTERNATIONAL BUSINESS MACHINES.           *
14:12:42 ************************************************************
14:12:42
14:12:42 HCPZCO6718I Using parm disk 1 on volume 440RES (device 0400).
14:12:42 HCPZCO6718I Parm disk resides on cylinders 391 through 435.
14:12:43
14:12:43 HCPISU951I CP VOLID 440W03 NOT MOUNTED
14:12:43 HCPISU951I CP VOLID 440W04 NOT MOUNTED
14:12:43 HCPISU951I CP VOLID 440W05 NOT MOUNTED
14:12:43 HCPISU951I CP VOLID 440W06 NOT MOUNTED
14:12:43 HCPISU951I CP VOLID 440W07 NOT MOUNTED
14:12:43 HCPISU951I CP VOLID 440W08 NOT MOUNTED
14:12:43 Start ((Warm|Force|COLD|CLEAN) (DRain) (DIsable)  (NODIRect)
14:12:43       (NOAUTOlog)) or (SHUTDOWN)


cold drain noautolog
                                                          CP READ   ZVMV4R40
```

Figure 7-2 Initial z/VM installation screen

We entered **cold drain noautolog** and later replied **no** to a question about the Time of Day (TOD) clock. At this point, we are logged on to the system as OPERATOR. The last message we received was about EREP records being accumulated. We then entered a **disc** command to disconnect from the OPERATOR user ID.

After disconnecting from OPERATOR (and pressing Enter), we received a standard z/VM logon screen. We logged on to the system as MAINT (with password MAINT) and pressed Enter to obtain a Ready prompt. We entered the command **instplan**. This produced the Installation Planning screen shown in Figure 7-3. We accepted all the default values and entered an X by the 3390 Triple option and an X by the language option; PF5 then was used to process these options.

```
                        *** z/VM INSTALLATION PLANNING ***

    Mark the product(s) selected to be installed into the VMSYS filepool with an
    "F" and those selected to be installed to minidisks with an "M"

    Install To    Product    Install To    Product    Install To    Product
    ----------   ---------   ----------   ---------   ----------   ---------
        M        VM              M        RSCS            M        TCPIP
        M        OSA             M        TSM             M        ICKDSF
        M        RTM             M        VMPRF           M        DIRM
        M        RACF            M        PERFTK          M        VMHCD


    Place a nonblank character in front of the System Default Language you would
    like for your system.

        X AMENG         _ UCENG         _ KANJI         _ GERMAN

    Place a nonblank character in front of the DASD model onto which your
    z/VM system will be loaded.  Only one model may be selected.

        _ 3390 Single        _ 3390 Double        x 3390 Triple



          PF1 = HELP   PF3/PF12 = QUIT   PF5 = Process   ENTER = Refresh
```

Figure 7-3 z/VM Installation Planning

At the next Ready prompt, we entered the command:

```
attach 460 * 181
```

At this point, we still had the first z/VM OMA CD mounted at FLEX-*ES* address 460, so no additional FLEX-*ES* commands were needed.[30] We then entered the **instvm cd** command. This caused a long pause, and we needed to clear the screen for more output. The screen shown in Figure 7-4 on page 68 was shown and we entered 181 as the address for all four VADDR fields. PF5 then started the loading process.

[30] If this were not the case, we would need to issue the flexescli **mount** command again.

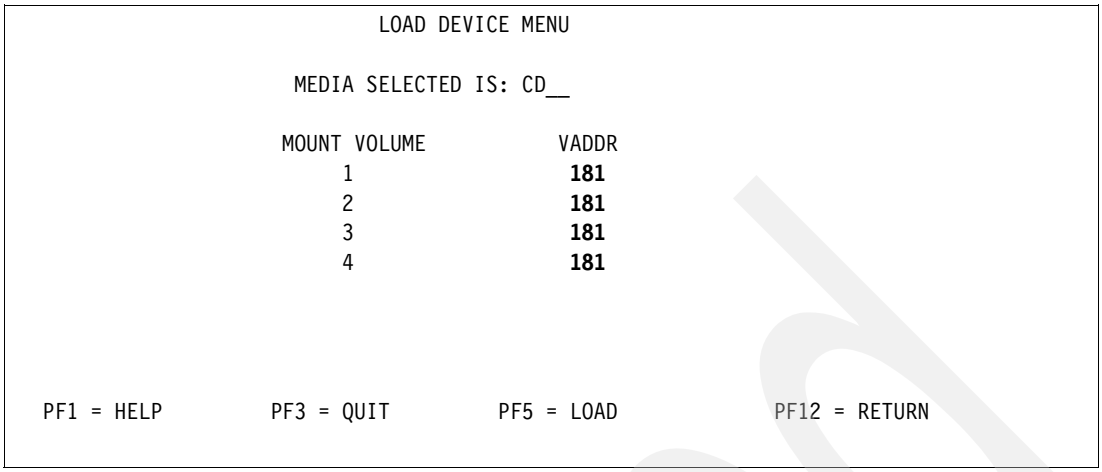

Figure 7-4 Load Device Menu

The loading process produces many screens of output that are automatically scrolled as needed. The installation process eventually issued the following messages:

```
INSTALL PROCESSING CONTINUES
(very long pause, several minutes)
PLEASE MOUNT VOLUME 2 ON TAPE DRIVE 181 THEN PRESS ENTER TO CONTINUE
```

At this point, we did the following:

```
from root          # umount /mnt/cdrom
   (Remove first CD, insert second CD.)
   (Wait for Linux automount or issue a mount /dev/cdrom /mnt/cdrom command.)
from flexescli     flexes> mount 460 /mnt/cdrom:/mnt/cdrom/tapes/uaa933.tdf
                   flexes>
on VM 3270         Enter
```

Processing continued with the second volume. Note that the CD file name was uaa933.tdf for the second volume.

This process was repeated for the remaining two z/VM CDs. That is, we umounted (using the **umount** command) the current CD on Red Hat, mounted the next CD, issued a flexescli **mount** command, and pressed Enter for z/VM. The relevant numbers for the last two CDs were uaa934.tdf and uaa935.tdf.

After the last CD was processed, a considerable number of messages were displayed mixed with pauses of various lengths. Eventually, the message `INSTVM EXEC ENDED SUCCESSFULLY` was displayed.

At this point, we were stilled logged on the system as MAINT and we still had 460 attached as 181. The next step was to install the Recommended Service Upgrade (RSU) CD. We did the following:

```
# umount /mnt/cdrom                                (Remove last z/VM CD.)
(Insert the RSU CD for Linux automount.)
flexes> mount 460 /mnt/cdrom:/mnt/cdrom/tapes/uaa940.tdf

z/VM:  ipl cms (and press Enter)                   (Enter this z/VM command.)
z/VM:  service                                     (Enter this z/VM command.)
```

The service function produced many screens of output (and cleared the screen automatically as needed). Eventually (after about 30 minutes), it produced the message `SERVICE processing completed successfully`.

The final step in the z/VM basic installation used the **put2prod** command. This required several minutes and produced multiple screens of output. It ended with the message `PUT2PROD processing completed successfully`.

This completed the basic z/VM installation. We entered a **shutdown reipl** command and verified that z/VM started correctly.

7.3 Customization

After completing the basic z/VM installation, we proceeded with the minor customization steps described in this section.

7.3.1 TCP/IP customization

We logged on to MAINT (password MAINT) and used a wizard to configure our TCP/IP:

```
access 193 e
ipwizard
   During the ipwizard dialog, we set:
      User ID of VM TCP/IP Stack Virtual machine: TCPIP
      Host Name: FLEXVM44
      Domain Name: itsoefs.ibm.com              (Not a real domain name!)
      DNS Addresses:  (None were set.)
      Gateway IP Address: 192.168.0.1           (Address of our small router.)
      Interface Name: ETH1
      Device Number: 4E0
      IP Address: 192.168.0.113
      Subnet Mask: 255.255.255.0
      Interface Type: LCS
         (PF8 to continue.)
      Network Type: Ethernet
      Port/Adapter number: 0
      Maximum Transmission Unit (MTU) Size: 1500
         (PF5 to process.)
```

Many of these selections are arbitrary. The device number (4E0) matches a 3172 device in the FLEX-*ES* definitions. The IP address (192.168.0.113) matches the FLEX-*ES* definitions. The gateway (192.168.0.1) is the router we use with FLEX-*ES*. This is described in the *Additional Topics* redbook. The domain name (itsoefs.ibm.com) is not a registered domain name and is not used for anything. The host name (FLEXVM44) matches the z/VM system ID we assigned.

7.3.2 Autolog1 profile

We logged on to AUTOLOG1 (password AUTOLOG1) and did the following:

```
ipl cms                                 (IPL CMS.)
filelist * * a                          (Look on the A disk.)
   Select file PROFILE EXEC for XEDIT.
   Update the profile to look like this:
      /*************************************************************/
      /*  Autolog Profile Exec                                     */
      /*************************************************************/
      ADDRESS COMMAND CP AUTOLOG VMSERVS
      ADDRESS COMMAND CP AUTOLOG VMSERVU
      ADDRESS COMMAND CP AUTOLOG VMSERVR
      ADDRESS COMMAND CP XAUTOLOG TCPIP
```

```
   'CP ENABLE ALL'
   'CP RECORDING ALL OFF'
   'EXEC TELL OP RECORDING ALL OFF was executed'
Save the changed file.
```

This change (at the next IPL) causes all 3270s to be enabled for logging on and disables the recording of various files to SPOOL. It also starts TCP/IP automatically.

7.3.3 Additional CP volume and CP configuration

We earlier created and formatted an extra CP disk at address 403 with volser VMPAG1. We decided to reserve the first 1000 cylinders for future minidisk space and use the remaining 2339 cylinders as additional z/VM paging space. To do this, we use the **cpfmtxa** command to allocate the appropriate cylinders for paging. (Cylinders on a CP disk that are not allocated for paging, spooling, temporary disks, or directory space are assumed to be PERM space that can be used for minidisks.)

```
Log on to maint (password maint).
attach 403 *                                    (Our extra volume is at address
403.)
cpfmtxa                                         (Start the cpfmtxa program.)
ENTER FORMAT, ALLOCATE, LABEL, or QUIT
allocate                                        (We want to allocate page space.)
ENTER THE VDEV TO BE PROCESSED OR QUIT
403                                             (We want to work on device 403.)
ENTER THE VOLUME LABEL FOR DISK 0403
vmpag1                                          (Verify the volser on the volume.)
..........
page 1000 end                                   (Page space: cylinder 1000 to end.)
end                                             (Must enter END to see results.)
..........
       TYPE    START   END   TOTAL
       ----    -----   ----  -----
       PERM        0    999   1000
       PAGE     1000   3338   2339
(End of cpfmtxa program.)
```

The next steps required access to the primary z/VM system configuration control minidisk. This minidisk is owned by MAINT and has virtual address CF1. The process for updating this is unique, and we combined several different tasks into one update of the configuration control minidisk:

```
cprel a                                         (Release from CP control.)
link * cf1 cf1 mr                               (Link so we can access it.)
acc cf1 e                                       (Access as file mode e.)
x system config e                               (Use XEDIT on this file.)
   (Use XEDIT to change various parts of this file:)
      (Change CP volume owned list, as shown in Figure 7-5 on page 71.)
      (Change console list, as shown in Figure 7-6 on page 71.)
      (Change system id, as shown in Figure 7-7 on page 72.)
      (Change disconnect timeout, as shown in Figure 7-8 on page 72.)
   file                                         (Save changes and exit XEDIT.)
acc 193 g                                       (Access disk containing cpsyntax.)
cpsyntax system config e                        (Check for errors in file.)
         NO ERRORS ENCOUNTERED
x local logo                                    (XEDIT another file on same disk)
   (Change the logo screen, as shown in Figure 7-9 on page 72.)
rel e                                           (Release the control minidisk.)
det cf1                                         (Detach it from our user ID.)
cpaccess maint cf1 a                            (Return it to CP control.)
```

```
shutdown                                          (Shut down z/VM.)

(ReIPL the system.)

flexes> ipl 400                                 (No need to specify console address
now.)
warm                                              (Warm start z/VM.)
(Log on to maint.)
q dasd all                                        (403 now CP owned.)
q alloc                                           (Paging space on 403 is present.)
```

The “artwork” we used for the system logo is rather basic. It is easy to change, and most z/VM owners devise their own logo.

We then added a minidisk definition in the z/VM directory corresponding to the paging space we just added. This will allow the **mapdisk** command to properly list the space on the CP disks. The line added in the directory is shown in Figure 7-10 on page 73.

```
 SYSTEM   CONFIG   E1  F 80  Trunc=80 Size=283 Line=56 Col=1 Alt=0

00070 /*********************************************************************/
00071 /*                    CP_Owned Volume Statements                     */
00072 /*********************************************************************/
00073
00074     CP_Owned   Slot   1  440RES
00075     CP_Owned   Slot   2  440WO1
00076     CP_Owned   Slot   3  440WO2
00077     CP_Owned   Slot   4  VMPAG1
00078     CP_Owned   Slot   5  RESERVED
```

Figure 7-5 CP-owned volume list

```
 SYSTEM   CONFIG   E1  F 80  Trunc=80 Size=283 Line=196 Col=1 Alt=1

00209 /*********************************************************************/
00210 /*                    Console Definitions                            */
00211 /*********************************************************************/
00212
00213  Operator_Consoles     0700,
00214                        System_Console System_3270
00215  Emergency_Message_Consoles   0700
00216
====>
================================================================================
```

Figure 7-6 Console definition

```
 SYSTEM   CONFIG   E1  F 80  Trunc=80 Size=283 Line=111 Col=1 Alt=3

00111 /*                  System_Identifier Information                    */
00112 /*********************************************************************/
00113
00114     System_Identifier_Default FLEXVM44
00115
====>
```

Figure 7-7 System identifier

```
 SYSTEM   CONFIG   E1  F 80  Trunc=80 Size=283 Line=167 Col=1 Alt=4

00167      Set_Privclass ,              /* Disallow SET PRIVCLASS command */
00168      LogMsg_From_File ,           /* No LOGMSG from SYSTEM LOGMSG   */
00169      Auto_Warm_IPL ,              /* Prompt at IPL always           */
00170      Clear_TDisk   ,              /* Don't clear TDisks at IPL time */
00171    Retrieve ,                     /* Retrieve options               */
00172      Default  7 ,                 /* Default.... default is 7       */
00173      Maximum  7 ,                 /* Maximum.... default is 7       */
00174    MaxUsers noLimit ,             /* No limit on number of users    */
00175    Passwords_on_Cmds ,            /* What commands allow passwords? */
00176      Autolog  yes ,               /* ... AUTOLOG does               */
00177      Link     yes ,               /* ... LINK does                  */
00178      Logon    yes,                /* ... and LOGON does, too        */
00179    DISCONNECT_TIMEout OFF
00180
====>
```

Figure 7-8 Disconnect timeout control

```
00000 * * * Top of File * * *
00001 ITSO/EFS PROJECT          ITSO/EFS/PROJECT          ITSO/EFS PROJECT
00002
00003
00004                      /  V       V  MM     MM      IBM Internal Use
00005            zzzzz     /    V     V   M M   M M
00006               z      /     V   V    M  M M  M          <-- + -->
00007              z       /      V V     M   M   M
00008             zzzzz    /       V      M       M      IBM Internal Use
00009
00010
00011   A basic z/VM 4.4.0 system for new VM owners running under FLEX-ES
00012
00013 ITSO/EFS PROJECT          ITSO/EFS/PROJECT          ITSO/EFS PROJECT
00014
00015 * * * End of File * * *
```

Figure 7-9 Customized logo file

```
 USER     DIRECT   C1  F 80  Trunc=72 Size=1885 Line=84 Col=1 Alt=1

00098 *
00099 USER $PAGE$   NOLOG
00100  MDISK A03 3390 257 134 440RES R
00101  MDISK A04 3390 1000 2338 VMPAG1 R
00102 *
====>
================================================================================
```

Figure 7-10 Minidisk definition for paging areas

7.3.4 User profiles

We added several virtual machine definitions (“users”) in our basic z/VM system. These include:

- A holding user ID to own our z/OS volumes. This user ID never logs on to the system. It is needed in order to allow the z/OS volumes to be used simultaneously by several virtual machines.
- A user ID for running the full z/OS system.
- A user ID for running a single-volume z/OS system.
- A user ID for running a simple, single-volume Linux for S/390 system.
- A basic CMS user ID.

The process used for adding these definitions is:

```
(Log on as MAINT.)
x user direct c                                  (XEDIT the directory source.)
   (Using XEDIT, add the definitions shown in “Directory additions” on page 90.)
file                                             (Save and exit from XEDIT.)
directxa user direct c                           (Activate the new directory.)
```

Notice that the new user IDs (except MVSDUMMY) each have a minidisk defined at address 191. This becomes the CMS A disk for the user ID. The minidisks are on the VMPAG1 volume and are allocated as follows:

```
USERID    STARTING CYLINDER     NUMBER OF CYLINDERS
mvsad           001                   2
mvssa           003                   2
suse1           005                   2
bill            007                   2
```

This leaves cylinders 9 through 999 (on VMPAG1) available for future minidisk allocations. (The remainder of the disk, starting at cylinder 1000, is allocated for z/VM paging.)

7.4 Creating the CD system

We used the method described in Chapter 5 of the *Additional Topics* book to create a CD containing our z/VM system. Briefly, this was done as follows:

```
$ mkdir /tmp/burn1                          (Create directory on Red Hat.)
```

(By earlier testing, we found that our compressed volumes would fit on a single CD.)

We created compressed backups of each of our four z/VM volumes using a **ckdbackup** command and a **gzip** command:

```
$ ckdbackup /s390/440RES | gzip -c9 > /tmp/burn1/440RES.gz
$ ckdbackup /s390/440W01 | gzip -c9 > /tmp/burn1/440W01.gz
$ ckdbackup /s390/440W02 | gzip -c9 > /tmp/burn1/440W02.gz
$ ckdbackup /s390/VMPAG1 | gzip -c9 > /tmp/burn1/VMPAG1.gz
```

We copied several PDF files (from various sources) into the burn1 directory, as well as the Multi file and shell script.

We then loaded a blank CD and burned the CD as follows, using the **mkisofs** and **cdrecord** commands:

```
$ su                                    (Need to be root.)
# cdrecord -scanbus                     (Can we see the CD writer?)
# mkisofs -R /tmp/burn1/* | cdrecord -v fs=6m dev=0,0 -
```

Our compressed volumes barely fit on a single CD. If future versions of our system do not fit on a single CD, we would create a /tmp/burn2 directory, place some of the compressed volumes there, and burn a second CD from this directory.

8

FLEX-ES Release 6.2.16

At the time of writing this book, the latest FLEX-*ES* Release is 6.2.16. This release contains several functional enhancements that might be of interest to many users. The major items are:

- Odd-sized CKD drives are supported. For example, you can define a 3390 drive with 1 to 32760 cylinders. This is fully supported by z/OS and z/VM.[31]
- A new network driver specification has been added. This allows LAN communication between S/390 systems running under FLEX-*ES* and the host Linux system.
- Additional FLEX-*ES* definition parameters have been added for LAN devices. These are especially useful for running guest machines under z/VM.
- The S90FLEXES startup script (automatically run during Red Hat Linux booting) is now automatically replaced by new FLEX-*ES* releases.
- Additional tape emulation types are provided.

The material in this chapter is provided for general information. The functions and the FLEX-*ES* release discussed here are not unique to z/VM usage and apply to all uses of FLEX-*ES* running under Linux.[32]

Remember that your FLEX-*ES* license code level must be later than or equal to the FLEX-*ES* release level you want to use. Also remember to recompile (with the **`cfcomp`** command) all your FLEX-*ES* definitions after you install a new FLEX-*ES* release.

[31] This is fully supported for 3390 drives; we are not certain about earlier disk types.
[32] The changes also apply to FLEX-*ES* under UnixWare, but minor details can differ.

8.1 Emulated CKD drive sizes

The “odd-sized” drive support applies to all CKD devices, although the following discussion and examples are only in terms of 3390 drives.

Earlier FLEX-*ES* releases supported only standard 3390 drives, corresponding to 3390-1, 3390-2, 3390-3, and 3390-9 drives. This release allows a 3390 drive to be any (integer) number of cylinders from 1 to 32760. z/OS and z/VM have supported this for several years.

This functional enhancement expands the syntax that can be used with several FLEX-*ES* commands and definitions. It is fully backward compatible; that is, no change is needed for existing FLEX-*ES* definitions or command scripts.

Examples of the expanded commands are:

```
$ ckdfmt -c 500 -n -r /s390/work01 3390-1

$ unzip -p /mnt/cdrom/x/work03.zip | ckdconvaws -c 500 -r - /s390/WORK03 3390-1

$ ckdfmt -c 500 -n /dev/raw/A90 3390-1
```

In these examples, a 3390 with 500 cylinders is created. The new command parameter is in the format **-c nnn**, where nnn is the number of cylinders wanted. The last example is in the typical format for a raw disk device. The second example assumes the zipped file was created from a 500 cylinder 3390.

In all three of these examples, we could use 3390-1, 3390-2, 3390-3, or 3390-9 as the device type. The maximum number of cylinders possible depends on the device type specified in the FLEX-*ES* definition or command. The maximums are:

Device type	Maximum number of cylinders
3390-1	1113
3390-2	2226
3390-3	3339
3390-9	32760

This means that a 3390-9 device must be specified in the FLEX-*ES* creation commands (and in the FLEX-*ES* definitions) for any 3390 larger than 3339 cylinders. Note that the largest size (32760 cylinders) is approximately 28 GB.

FLEX-*ES* definitions continue to use 3390-1, 3390-2, 3390-3, and 3390-9 specifications. The actual size of the volume is sensed when it is opened by the S/390 instance. For example:

```
CUA90: cu 3990
interface local(1)
device(00) 3390-1 /s390/WORK03
end CUA90
```

This is the proper definition for a standard 3390-1 volume or any odd-sized volume less than 1113 cylinders.[33]

An odd-sized 3390 is treated just like a standard 3390. For example, ICKDSF is used to create a label and VTOC. The only difference between odd-sized and standard-sized volumes (other than the number of cylinders) is that the “extra” cylinders (for alternate tracks and CE use) are not present on odd-sized volumes. This has no practical effect for z/OS, z/VM, or VSE.

[33] We could specify 3390-1, 3390-2, 3390-3, or 3390-9 in this example. The actual volume size (for an odd-sized volume) must be less than or equal to the maximum number of cylinders for the specified device type.

8.2 LAN support

Multiple “ipaddress=” operands can now be specified for control units used for TCP/IP. Up to 14 addresses can be specified. This is most useful when used with z/VM (with *proxyarp* support in the TCP/IP stack for z/VM) and Linux (and other) guests under z/VM.

The format “ipaddress=123.456.789.123/m” can also be used. In this format, the *m* operand specifies the number of bits in the net mask. The use of this format also specifies that network packets containing a destination IP address that matches any of the host addresses permitted by the net mask are to be sent to the emulated S/390 device. For example, ipaddress=10.20.30.0/28 specifies that the first 28 bits of the IP address are a network address and the last 4 bits are for host addresses. Network packets containing a destination IP address that matches any of the possible host addresses (16 addresses in this example[34]) are passed through the filter and will be seen by the emulated S/390 device.

The *tun* driver is new. This driver permits communication between a S/390 operating system (such as z/OS or z/VM) and Red Hat Linux without requiring an actual LAN connection.[35] (This is especially useful when giving demonstrations on ThinkPads.) It is important to note that the tun driver is usable only with Ethernet devices and definitions.[36]

An example using the new functions follows:

```
vm2E0: cu osa
interface local(1)
options 'ipaddress=192.168.20.1,ipaddress=192.168.21.0/24,tunipaddress=192.168.20.253'
device(00) osa tun
device(01) osa OFFLINE
end vm2E0
```

Using z/VM for the example, the following IP addresses are passed to z/VM on the emulated S/390 device:

- IP address 192.168.20.1 (probably for z/VM itself)
- IP addresses 192.168.21.0 through 192.168.21.255 (for a guest LAN under z/VM)
- IP address 192.168.20.253 (for the hosting Red Hat Linux system)

The “tun” parameter in the device definition and in the ipaddress parameter refers to the IP address used by the underlying Red Hat Linux system. The tun driver does not require that the physical Ethernet interface be connected to a router or hub. Prior to this FLEX-*ES* release, Red Hat 9.0 required that the physical Ethernet interface be connected to a hub or router before IP communication could be established between FLEX-*ES* emulation and Red Hat itself. This was especially awkward when using ThinkPads for demonstration systems.

Filter bit mask

As already mentioned, with FLEX-*ES* Release 6.2.16 we can specify up to 14 ipaddress= values. This can still be restrictive for a z/VM guest LAN.

Typically, a z/VM guest LAN is defined as a different subnet than what is used for the z/VM TCP/IP stack itself. For example, the IP address for z/VM might be 192.168.20.1 (on the

[34] Host addresses of all zeros or all ones are special cases, of course.

[35] This could be done without a special driver when using Red Hat 8.0. Red Hat 9.0 requires an actual Ethernet connection to a hub or router before TCP/IP communication between Red Hat and FLEX-*ES* emulated devices will work. This can be awkward when using a PC that is not connected to any external device. The tun driver bypasses this problem and permits internal TCP/IP connectivity without any external connection. (An alternative to the tun driver is the use of an Ethernet loop plug. This has not been formally tested by FSI or IBM, but a number of users have reported that a loop plug solves this problem.)

[36] The z/OS or z/VM definitions must specify Ethernet.

192.168.20 subnet) and the IP addresses for several guest machines might be 192.168.21.xxx (on the 192.168.21 subnet).[37]

The specification `ipaddress=192.168.21.0/24` does two things:

- It specifies subnet 192.168.21 (using the first 24 bits of the IP address).
- It accepts (passes through the filter) *all* host addresses on this subnet. That is, network packets containing a destination IP address that matches any of the IP addresses in the range 192.168.21.0 through 192.168.21.255 will be passed to the emulated S/390 machine.

We might assume that the emulated S/390 machine is running z/VM with a number of guest Linux virtual machines. We would assign IP addresses for the Linux virtual machines in the subnet 192.168.21 range. We would also specify *proxyarp* in the z/VM TCP/IP definitions. This causes z/VM to answer ARP requests on behalf of the guest virtual machines, and this improves general system performance.

When external IP traffic for one of the Linux guest machines is received, it passes through the ipaddress filter and goes to z/VM. z/VM then routes the traffic to the guest LAN[38] and the appropriate guest Linux will receive it.

8.3 Emulated tape drives

Several improvements are available for emulated tape drives.

Device type 3423

Device type 3423 is identical to device type 3422, but is used by z/OS to permit different esoteric names for selected devices. Type 3423 can be used in any FLEX-*ES* definition element where device type 3422 could be used. The 3423 device type should be used only with z/OS. (z/VM does not automatically recognize this device type.)

Device type 3590

The initial support for emulated 3590s is for *FakeTape* disk files only. SCSI-attached drives (including SCSI 3590 drives) are *not* supported as 3590 devices. ESCON®-attached 3590 drives will be supported when ESCON adapters become available.

Both the cu type and the device type are specified as 3590 in the FLEX-*ES* definitions. For example:

```
CU590: cu 3590
interface local(1)
device(00) 3590 OFFLINE
end CU590
```

SCSI tape names

The “normal” Linux name for a SCSI tape drive, such as /dev/st0, can now be used in FLEX-*ES* definitions or CLI **`mount`** commands. FLEX-*ES* will automatically convert these names into the appropriate SCSI generic names, such as /dev/sg1. (You can still use the /dev/sg1 type names.) Finding the correct device name for a SCSI tape can be slightly complicated; it is discussed in Chapter 10 of the *Additional Topics* redbook.

[37] Subnets need not be on “even” 8-bit boundaries, of course, but they usually are defined on 8-bit boundaries, and our examples use typical values.

[38] z/VM could also route it over a virtual CTC or IUCV connection, but a guest LAN is the most general technique.

8.4 S90FLEXES script

The first installation of FLEX-*ES* causes an *rc* file (containing a shell script) to be added to the base Linux system. This shell script is run when Linux is booted. Prior to this release, the FLEX-*ES* installation process checked for the existence of /etc/rc.d/init.d/FLEXES. If it existed, it was *not* replaced by the FLEX-*ES* installation process.

Starting with FLEX-*ES* Release 6.2.16, the FLEXES file (which is the real file linked to by S90FLEXES) is *always* replaced by an upgrade to a new FLEX-*ES* release. The first time this happens, the old file is saved in /var/adm/flexes/S90FLEXES.orig. We strongly recommend that you do not modify the FLEXES (S90FLEXES) file.

The EFS series of Redbooks has not discussed modifications to the S90FLEXES script. However, production users of FLEX-*ES* (and the FSI business partners who supply these systems) often customize the S90FLEXES shell script. This customization typically has two purposes: to automatically issue a **resadm** command while booting (to start the FLEX-*ES* resource manager) and to enable raw disk devices. (See Chapter 5 of the *Additional Topics* redbook for more information about the use of raw disk devices.)

The S90FLEXES shell script installed by Release 6.2.16 (and future releases) calls an *exit* from an appropriate place in the shell script. This exit, which is a shell script file in /var/adm/flexes/.flexes, can issue whatever customized commands are needed for the installation. (Note the period in the name of this shell script file.) Users (and business partners) can place their **resadm** command, raw device definitions, and whatever else is needed in this file.

9

FAQs

Question: Can I have multiple preferred guests with z/VM under FLEX-*ES*?
Answer: No. The V=F function required to do this is available only with PR/SM™ hardware. FLEX-*ES* does not provide PR/SM emulation. You can have one V=R guest when running z/VM under FLEX-*ES*.

Q: Does FLEX-*ES* emulate hipersockets?
A: The hardware hipersockets of zSeries processors are not emulated by FLEX-*ES* at this time. However, z/VM provides *guest LAN* support. From the viewpoint of a z/VM guest operating system (such as Linux or z/OS), this is very similar to hipersockets. (Users of z/VM 4.3 need to ensure that PTFs UM30439 and UM30592 are installed.)

Q: I tried to use the OMA @TDF file and there appears to be something wrong with the format of the file names. How does the FLEX-*ES* OMA reader handle this?
A: The TDF files contain lines such as:

```
\AUTONAME\FILE0000.  HEADERS
\AUTONAME\FILE0001.  HEADERS
```

The trailing period in the file name is not a standard convention. It worked on OS/2®, where OMA originated. FLEX-*ES* has included special code to handle these names. Other Linux utilities might choke on this format.

Q: Is a ThinkPad really a practical machine for running multiple guest machines (such as z/OS and Linux) under z/VM?
A: Barely, if done with care. The key problems are memory and paging. A “real” S/390 with 768 MB memory (the typical size of an emulated S/390 in a 1 GB ThinkPad) would probably do substantial paging when running multiple guests if their combined virtual machine sizes exceeded, say, 512 MB. The practical disk bandwidth on a ThinkPad can easily be exceeded by z/VM paging, plus normal I/O for z/OS, plus other operating systems. The problem can be minimized by defining small virtual machines for the guest systems and IPLing them one by one, waiting for I/O to minimize before IPLing another guest system.

Q: Is an xServer a practical machine for running multiple guest machines (such as z/OS and Linux) under z/VM?
A: Yes, definitely. An xServer can have more real memory; 2.5-3 GB is effective for a single FLEX-*ES* instance. It can have much more bandwidth to disks, typically through fast RAID adapters and raw disk interfaces. It can have multiple processors emulating a multiprocessor S/390.

Q: In the *Additional Topics* book you strongly recommend raw disk devices for performance. Why are they not used in this book?
A: We do strongly recommend raw disk devices for performance and for any production system. Nevertheless, these are somewhat more complex to set up and administer. We felt this would have complicated the presentation in this book. We wanted to concentrate on z/VM topics here and keep other complexity to a minimum. Note that the ITSO/EFS z/VM CD mentioned in the text can be restored to raw disk devices as well as to simple Linux files.

Q: In the process of installing my FLEX-*ES* system, I found that I needed to copy CDs several times. Is there an easy way to verify that a CD copy is correct?
A: Some CD writing software can perform a read-back verification of the newly written CD. If your CD writer software does not offer this option, you might try the following technique (described by Gary Eheman):

a. Perform the CD copy function, using your CD writer package.
b. Mount the *source* CD at /mnt/cdrom (in Red Hat Linux).
c. Change your command-line window to /tmp and run the following command from a Red Hat command line:
```
find /mnt/cdrom -name '*' | xargs md5sum -b $1 > md5sums.file
```
d. Unmount the source CD and mount the *copy* CD.
e. Run the following command:
```
md5sum --check md5sums.file
```
f. Observe the output. There should be an OK message for every file on the CD.

Q: I want to run a z/OS system and a SUSE LINUX for S/390 system. I believe I can do this with two FLEX-*ES* S/390 images, or with a single FLEX-*ES* image running z/VM. Which is better?
A: Both ways can work and there is no simple answer for this question. Considerations include the following:

– Running under z/VM requires basic z/VM skills. (This redbook is intended to help in this area.)
– z/VM can offer more balanced dispatching of virtual machines under it than might be found with multiple S/390 instances dispatched by the base Linux. We have no specific measurements to demonstrate this effect.
– z/VM can offer *guest LAN* support, allowing a sophisticated internal LAN environment. This is not used in the simple implementation described in this document.
– z/VM can allow a slight overcommitment of real S/390 storage (as specified in the FLEX-*ES* definitions). The overcommitment is handled by z/VM paging. However, this must be balanced against the S/390 storage needed by z/VM itself.

Q: I see many familiar names as user IDs in the z/VM directory. Can I use these? For example, can I activate RACF® by logging on to user ID RACFVM?
A: In general, the answer is no. Many of the user IDs in the directory are related to products that must be licensed and activated before the user ID can be used.

Q: How can I resize an x3270 window?
A: Select Options (in the toolbar), select font, and then select a different font size. This effectively resizes the window. An attempt to resize a window by “grabbing” a corner of the window with the pointer and attempting to move it usually results in the instant termination of the window. We do not know how to graphically resize an x3270 window with a mouse.

Q: The x3270 module supplied with FLEX-*ES* seems to be at a earlier level than other x3270 versions that I have. Can I use a later version?
A: It is better to use the FLEX-*ES* version of x3270 on the Linux system running FLEX-*ES*. It has been modified so that a FLEX-*ES* terminal name (such as *mstcon* or *L701*) can be supplied in the command line starting an x3270 window. An x3270 session connected through the Terminal Solicitor does not need this modification.

Q: When I start FLEX-*ES* I receive a strange error message that implies I have the wrong version of x3270. What should I do?
A: Try the Linux command **`which x3270`**. It should show that you are using /usr/flexes/bin/x3270. You might have another x3270 installed somewhere in the search PATH. The search PATH for user ID *flexes* should search /usr/flexes/bin first. If you cannot resolve the problem, change your **`x3270`** command to something like this:

```
/usr/flexes/bin/x3270 -model 3 -keymap pc -port tn3270 localhost:mstcon &
```

Q: I prefer to do all my operating as *root*. What is wrong with this?
A: It is important to follow the instructions in the *Getting Started* redbook. Except for using **`resadm`** to start or modify the FLEX-*ES* resource manager, all FLEX-*ES* operation should be under user ID *flexes*. This user ID is automatically installed when FLEX-*ES* is installed and has a unique PATH definition among other properties. (We have spent considerable time with several users trying to undo problems created by running FLEX-*ES* as root.)

Q: I need more user connections than you allowed. How do I increase the number of users that can connect at the same time?
A: The basic requirement is to increase the number of 3270 devices in the FLEX-*ES* definitions. Remember to change the cu definition so that it specifies the correct number of devices. If you want more than 32 devices, it would be better to create another 3174 definition using the existing 3174 definition as a model. Remember to recompile the FLEX-*ES* definitions with the **`cfcomp`** command after making any changes.

Also, if you are using an AD z/OS system, remember that only 10 TSO user connections are defined in the distributed systems. This can be increased to 40 by adding definitions in the A600 member of VTAMLST. For more than 40, the PARMLIB member for TSO must be modified.

Q: Why did you use 31-bit SUSE LINUX? Why not use the 64-bit version?
A: This question is discussed in the *Additional Topics* book. Briefly, we used the 31-bit version because it is what we had available. We made no attempt to obtain a 64-bit version because we expected that it would not be very practical on a ThinkPad, especially with z/OS running concurrently. Remember that FLEX-*ES* works with 32-bit (Pentium®) hardware. It will correctly emulate 64-bit zArchitecture, but this operation requires more Pentium instructions for emulation and is considerably slower.

Q: Will PCOM work with FLEX-*ES*? Do the **`send`** and **`receive`** commands work?
A: Yes and yes, assuming your question refers to PCOM connections to z/OS or z/VM running under FLEX-*ES*. The **`send`** and **`receive`** commands work with the TSO ready prompt; they do not work with ISPF option 6.

Q: I was told that I need to order the fastest possible ThinkPad for FLEX-*ES*. Is this correct?
A: Maybe. It depends on what you want to run under FLEX-*ES*. We have run OS/390®, with 10-15 student TSO users, on a 750 MHz ThinkPad with excellent results. We have run the z/OS WebSphere® Application Server on 3 GHz machines with barely acceptable performance. The z/VM system described in this document was run on a T23 ThinkPad. Basic z/VM performance was excellent. z/OS was usually good, depending on what other virtual machines were running at the same time. SUSE LINUX for S/390 was acceptable for basic operation, although x-Windows graphics and very large shell scripts were slow.

Q: Can I connect my z/VM system to the "real Internet?"
A: In principle, yes. However this is a complex question for reasons unrelated to FLEX-*ES*. Please see Chapter 3 of the *Additional Topics* book for a basic discussion of this question.

Q: Can I use small 3390 volumes (smaller than a 3390-1) under FLEX-*ES*? I could do this with my P/390 system. This could provide some of the advantages of minidisks with or without z/VM.
A: We sometimes refer to such volumes as *odd-sized* volumes. Support for these was added to FLEX-*ES* in Release 6.2.16. This was after the text of this redbook was written, and we did not use odd-sized volumes for this project.

Q: Could I use small emulated disks (available in the new FLEX-*ES* release) instead of z/VM minidisks when cloning Linux guests?
A: We assume so, but we have not tried this. (An obvious question is whether the dasd driver for Linux supports odd-size 3390 volumes.)

Q: Does my ThinkPad need to be at any particular BIOS level?
A: FSI strongly advises that you upgrade your BIOS (whether ThinkPad or xServer) to the latest levels. The only specific problem we know is with T40 ThinkPads. We have examples where BIOS level 1.05 (1RET34Ww) and BIOS level 2.11 (1RET84Ww, controller version 2.04) work correctly, while intermediate levels might have problems with the USB port, DMA support for the hard disk, or both.

Q: Can I run parallel Linux applications while using FLEX-*ES*?
A: This is not recommended. There are several concerns. The first is memory use. The additional applications might cause Linux to page; this has severe effects on S/390 emulation. (We are not aware of any simple, practical way to reserve a specified amount of real memory for FLEX-*ES* use and allow Linux to page in the remaining memory.) Another problem is that z/OS has many internal timers running to detect various types of I/O problems. If insufficient processor time is available to z/OS, it might begin I/O error recovery.

The problem is more severe when using simple Linux files for S/390 volumes. This usage involves the automatic I/O buffering for normal Linux file systems and the interactions of these I/O buffers and sudden virtual memory demands (when starting a new Linux graphics application, for example) can cause delays long enough to disturb normal z/OS operation.

If you must run parallel applications, use raw disk devices for S/390 volumes and buy more memory. (The use of raw disk devices for S/390 volumes is discussed in the *Additional Topics* redbook.)

Q: Are *all* the z990 instructions emulated?
A: No. When FLEX-*ES* is running in ALS3 mode (“64-bit”), all the basic zArchitecture instructions are emulated. zArchitecture (including the zSeries 990) allows various optional instruction sets; the presence of these can be checked with various instructions. For example, the presence of either of two sets of cryptographic instructions (the coprocessors on z900s and the four integrated instructions on z990s) can be determined this way. (Neither of these functions are emulated by FLEX-*ES* at this time.) Likewise, the *long displacement* instructions added with the z990 can be detected this way. FLEX-*ES* does not emulate the long displacement instructions at this time (Release 6.2.16).

Q: Is the SIE facility, as used by z/VM, emulated?
A: Yes.

Q: Can z/VM, under FLEX-*ES*, use more than one Intel® processor to emulated a multiprocessor S/390? Can guest machines use more than one S/390 processor?
A: Yes and yes. (But you must have the appropriate FLEX-*ES* license and you must generate a new Linux kernel.) The number of emulated S/390 processors cannot exceed the number of Intel processors. Multiple Intel processors cannot be used (in parallel) to emulate a single S/390 processor.

Q: How can I order the original z/VM OMA CDs? Can I order z/OS in this format?
A: z/VM can be ordered through the same process used for ordering any S/390 software from IBM. The proper feature code is used to specify OMA CD deliverables. As far as we know, z/OS is not available in this format.

Q: You did not discuss VSE. Can I run VSE under the z/VM you describe? Where can I obtain VSE in the proper format?
A: Yes, you can run VSE. We did not mention it because (1) we did not have a copy of VSE available, and (2) we did not have any VSE skills. At the time of writing, we did not have a VSE implementation on CDs suitable for use with FLEX-*ES*, but this might change in the future.

Q: It appears that I could specify 3390-9 in all my FLEX-*ES* resource definitions, because a 3390-9 specification is now valid for any size 3390 drive. The actual drive size would be sensed when the S/390 instance is started. Is this a good idea?
A: Yes, you could do this. However, we prefer to specify the closest appropriate size (3390-1, 3390-2, 3390-3, 3390-9) in the resource definitions as a matter of documentation. Moreover, you must specify the same model number in the FLEX-*ES* resource definitions as was used when you created the volume.

Q: Does FLEX-*ES* emulate the hardware compression function of S/390s?
A: Yes.

Q: Can I change the time-of-day clock in the underlying Linux system while FLEX-*ES* is running?
A: No, this will probably cause immediate problems in FLEX-*ES* operation.

A

Quick reference

Common FLEX-*ES* commands:

```
$ ckdfmt -n /dev/raw/A90 3390-1          (Raw device)
$ ckdfmt -n -r /s390/WORK01              (Linux file)
$ unzip -p /mnt/cdrom/zos14/w4res1.zip | ckdconvaws -r - /s390/W4RES1 3390-3
$ unzip -p /mnt/cdrom/zos14/w4res1.zip | ckdconvaws - /dev/raw/A80 3390-3
flexes> d mount A80
# resadm   [-s -T -k -x] xxxx.rescf
flexes> mount 460 /mnt/cdrom:/mnt/cdrom/tapes/990.tdf
$ ckddump /dev/raw/A80 0 0 | grep VOL1
```

x3270 control:

```
# cd /usr/lib/X11/app-defaults              (Edit x3270 keymap.)
$ x3270 -model 3 -keymap pc -port tn3270 localhost &
```

LVM and raw device commands:

```
# /sbin/pvcreate  /dev/hda6
# /sbin/vgcreate vg390 /dev/hda6
# /sbin/lvcreate -L 2792M -n S4RES1 vg390     (1862M, 933M for -2 and -1 volumes))
# /sbin/lvrename /dev/vg390/S4RES1 /dev/vg390/W4RES1
# /sbin/vgdisplay vg390
# /sbin/lvscan
# /sbin/lvremove /dev/vg390/WORK01
# chown flexes:flexes /dev/vg390/W4RES1
# mknod /dev/raw/A80 c 162 nnn
# ls -l /dev/raw                            (Lists minor numbers that have been used.)
# chown flexes:flexes /dev/raw/A80
# raw /dev/raw/A80 /dev/vg390/W4RES1
# rm /dev/raw/A94
```

Common z/OS operator commands:

```
k a,none                          (Remove display area.)
k e,d                             (Erase display area.)
d m=cpu
d m=stor
d a,l                             (Display job and TSO user names.)
c jobname
c u=tsouser
```

Useful z/VM commands (CMS and CP):

```
id                                  (Who is logged onto this terminal?)
shutdown                            (End z/VM.)
q n                                 (Who is logged onto z/VM?)
cp term conmode 3270                (Ready console for MVS use.)
ipl a80 loadparm 0a82cs             (Remember to use "loadparm" before parameter.)
XEDIT filename filetype filemode
   quit    file    +nn     -nn    top    bottom
link joe 123                        (Link to another user's minidisk.)
access 123 e                        (Allow CMS to access a minidisk.)
release e                           (Remove CMS access. Undo the access command.)
detach 123                          (Remove user access. Undo the link command.)
q disk                              (List my CMS accessed minidisks.)
q dasd all                          (List real disks, as defined by FLEX-ES.)
filelist * * a                      (List files on CMS disk a.)
diskmap user direct c               (Create a disk map.)
browse user diskmap a               (View the disk map.)
force tcpip                         (Force user ID TCPIP to terminate.)
xautolog tcpip                      (Log on another user ID.)
ind                                 (How busy is the system?)
q stor                              (How large is my virtual machine? For authorized
                                    users (MAINT) it shows size of real memory.)
q system                            (Lists all the minidisks accessed.)
q accessed                          (Minidisks currently accessed by this user.)
q all                               (For MAINT, lists all real devices used by z/VM.
                                    For normal users, lists virtual machines for user.)
q links 123                         (Lists all user IDs with links to this minidisk.)
q pf                                (PF key definitions.)
set pf12 retrieve
dial mvssa                          (Connect to SPECIAL 3270 definition.)
x user direct c                     (As MAINT, edit z/VM directory.)
directxa user direct c              (Update active directory.)
vmlink tcpmaint 198                 (Combine link and access commands.)
vary online 705                     (Vary real device online to z/VM.)
enable 705, disable 705             (Enable or disable read devices.)
```

B

Listings

This section contains listings.

Directory additions

The following listingcontains the additions we made to the z/VM directory. These consist of a dummy user to own z/OS volumes, user MVSAD (for running the full AD system), user MVSSA (for running the single-volume system distributed with the AD system), SUSE1 (for running a single-volume SUSE LINUX for S/390 system), and BILL (a simple CMS user).

```
00108 *
00109 ************************************
00110 *
00111 USER MVSDUMMY NOLOG 256M 256M G
00112 MDISK A80 3390 DEVNO A80 MWV
00113 MDISK A81 3390 DEVNO A81 MWV
00114 MDISK A82 3390 DEVNO A82 MWV
00115 MDISK A83 3390 DEVNO A83 MWV
00116 MDISK A84 3390 DEVNO A84 MWV
00117 MDISK A85 3390 DEVNO A85 MWV
00118 MDISK A86 3390 DEVNO A86 MWV
00119 MDISK A87 3390 DEVNO A87 MWV
00120 MDISK A88 3390 DEVNO A88 MWV
00121 MDISK A89 3390 DEVNO A89 MWV
00122 MDISK A8A 3390 DEVNO A8A MWV
00123 MDISK A8B 3390 DEVNO A8B MWV
00124 MDISK A8C 3390 DEVNO A8C MWV
00125 MDISK A8D 3390 DEVNO A8D MWV
00126 MDISK A8E 3390 DEVNO A8E MWV
00127 MDISK A8F 3390 DEVNO A8F MWV
00128 *
00129 USER MVSAD MVSAD 256M 512M G
00130 MACH ESA
00131 CPU 00
00132 SPOOL 00C 2540 READER *
00133 SPOOL 00D 2540 PUNCH A
00134 SPOOL 00E 1403 A
00135 CONSOLE 700 3215 T
00136 LINK MAINT 0190 0190 RR
00137 LINK MAINT 019D 019D RR
00138 LINK MAINT 019E 019E RR
00139 LINK TCPMAINT 592 592 RR
00140 DEDICATE E20 E20
00141 DEDICATE E21 E21
00142 SPECIAL 701 3270
00143 SPECIAL 702 3270
00144 SPECIAL 703 3270
00145 SPECIAL 704 3270
00146 SPECIAL 705 3270
00147 MDISK 191 3390 001 002 VMPAG1 MR
00148 LINK MVSDUMMY A80 A80 MW
00149 LINK MVSDUMMY A81 A81 MW
00150 LINK MVSDUMMY A82 A82 MW
00151 LINK MVSDUMMY A83 A83 MW
00152 LINK MVSDUMMY A84 A84 MW
00153 LINK MVSDUMMY A85 A85 MW
00154 LINK MVSDUMMY A86 A86 MW
00155 LINK MVSDUMMY A87 A87 MW
00156 LINK MVSDUMMY A88 A88 MW
00157 LINK MVSDUMMY A89 A89 MW
00158 LINK MVSDUMMY A8A A8A MW
00159 LINK MVSDUMMY A8B A8B MW
00160 LINK MVSDUMMY A8C A8C MW
```

```
00161 LINK MVSDUMMY A8D A8D MW
00162 LINK MVSDUMMY A8E A8E MW
00163 LINK MVSDUMMY A8F A8F MW
00164 *
00165 USER MVSSA MVSSA 256M 512M G
00166 MACH ESA
00167 CPU 00
00168 SPOOL 00C 2540 READER *
00169 SPOOL 00D 2540 PUNCH A
00170 SPOOL 00E 1403 A
00171 CONSOLE 700 3215 T
00172 LINK MAINT 0190 0190 RR
00173 LINK MAINT 019D 019D RR
00174 LINK MAINT 019E 019E RR
00175 LINK TCPMAINT 592 592 RR
00176 SPECIAL 701 3270
00177 SPECIAL 702 3270
00178 MDISK 191 3390 003 002 VMPAG1 MR
00179 LINK MVSDUMMY A80 A80 MW
00180 LINK MVSDUMMY A81 A81 MW
00181 LINK MVSDUMMY A82 A82 MW
00182 LINK MVSDUMMY A83 A83 MW
00183 LINK MVSDUMMY A84 A84 MW
00184 LINK MVSDUMMY A85 A85 MW
00185 LINK MVSDUMMY A86 A86 MW
00186 LINK MVSDUMMY A87 A87 MW
00187 LINK MVSDUMMY A88 A88 MW
00188 LINK MVSDUMMY A89 A89 MW
00189 LINK MVSDUMMY A8A A8A MW
00190 LINK MVSDUMMY A8B A8B MW
00191 LINK MVSDUMMY A8C A8C MW
00192 LINK MVSDUMMY A8D A8D MW
00193 LINK MVSDUMMY A8E A8E MW
00194 LINK MVSDUMMY A8F A8F MW
00195 *
00196 USER SUSE1 SUSE1 128M 256M G
00197 MACH ESA
00198 CPU 00
00199 SPOOL 00C 2540 READER *
00200 SPOOL 00D 2540 PUNCH A
00201 SPOOL 00E 1403 A
00202 CONSOLE 700 3215 T
00203 LINK MAINT 0190 0190 RR
00204 LINK MAINT 019D 019D RR
00205 LINK MAINT 019E 019E RR
00206 LINK TCPMAINT 592 592 RR
00207 DEDICATE 2E0 2E0
00208 DEDICATE 2E1 2E1
00209 MDISK 191 3390 005 002 VMPAG1 MR
00210 MDISK 200 3390 DEVNO 200 MWV
00211 MDISK 201 3390 DEVNO 201 MWV
00212 MDISK 202 3390 DEVNO 202 MWV
00213 MDISK 203 3390 DEVNO 203 MWV
00214 *
00215 USER BILL BILL 64M 128M G
00216 MACH ESA
00217 CPU 00
00218 SPOOL 00C 2540 READER *
00219 SPOOL 00D 2540 PUNCH A
00220 SPOOL 00E 1403 A
```

```
00221 CONSOLE 009 3215 T
00222 LINK MAINT 0190 0190 RR
00223 LINK MAINT 019D 019D RR
00224 LINK MAINT 019E 019E RR
00225 LINK TCPMAINT 592 592 RR
00226 MDISK 191 3390 007 002 VMPAG1 MR
00227 *
00228 ************************************************************
```

FLEX-ES definitions (Multi file)

The followng lists FLEX-EX definitions (Multi file).

```
system Multi:                                   # System file = Multi.syscf.
memsize(786432)                                 # S/390 size.
cachesize(4096)
instset(z)                                      # z machine.
cpu(0)
channel(0) local
channel(1) local
channel(2) local
# 3270s used by all
cu devad(0x700,7) path(0) resource(CU3174)      # 3270s at addresses 700, 701, ...
# MVS control units
cu devad(0xA80,16) path(2) resource(MVS3990)    # 16 x 3390s at A80, A81, ...
cu devad(0xE20,2) path(1) resource(MVS3172)     # TCP/IP 3172 at E20/E21.
# Linux control units
cu devad(0x200,4) path(2) resource(LN3990)      # 4 x 3390 at 200, 201, 202, ...
cu devad(0x2E0,2) path(1) resource(LN3172A)     # 3172 for Linux A.
cu devad(0x2E2,2) path(1) resource(LN3172B)     # 3172 for Linux B.
# VM control units
cu devad(0x400,4) path(2) resource(VM3990)      # 4 x 3390 at 400, 401, 402, ...
cu devad(0x4E0,2) path(1) resource(VM3172)      # 3172 at 4E0/4E1 for VM.
cu devad(0x460,1) path(2) resource(VM3480)      # Tape drive.
end Multi

resources Multi:                            # Resource name = Multi.rescf.
CU3174: cu 3174
interface local(1)
device(00) 3278 mstcon                      # The number of 3270s defined here will
device(01) 3278 L701                        # be the total number that can be logged
device(02) 3278 L702                        # onto VM, and dialed to MVS, and
device(03) 3278 L703                        # connected to SUSE (as a 3215).
device(04) 3278 L704                        #
device(05) 3278 L705                        # Add more devices if you need them, and
device(06) 3278 L706                        # update the count in the cu statement.
end CU3174

# MVS resources
MVS3990: cu 3990
interface local(1)
device(00) 3390-3 /s390/S4RES1              # address A80
device(01) 3390-3 /s390/S4RES2              # address A81
device(02) 3390-3 /s390/OS39M1              # address A82
device(03) 3390-3 OFFLINE                      # Use OFFLINE devices to make
device(04) 3390-2 OFFLINE                      # the volume addresses match the
device(05) 3390-3 OFFLINE                      # commonly documented addresses for
device(06) 3390-3 OFFLINE                      # the AD volumes.
device(07) 3390-3 /s390/S4USS1              # address A87
device(08) 3390-3 OFFLINE
```

```
device(09) 3390-2 OFFLINE
device(10) 3390-3 OFFLINE
device(11) 3390-3 OFFLINE
device(12) 3390-3 /s390/SARES1              # address A8C
device(13) 3390-3 OFFLINE
device(14) 3390-3 OFFLINE
device(15) 3390-1 /s390/WORK02              # address A8F
end MVS3990

MVS3172: cu 3172
interface local(1)
options 'ipaddress=192.168.0.121,adapternumber=0'  #
device(00) 3172 eth0                        # address E20
device(01) 3172 OFFLINE                     # address E21
end MVS3172

# Linux resources
LN3990: cu 3990
interface local(1)
device(00) 3390-3 /s390/LNXA01              # address 200
device(01) 3390-3 OFFLINE
device(02) 3390-3 OFFLINE
device(03) 3390-3 OFFLINE
end LN3990

LN3172A: cu 3172
interface local(1)
options 'ipaddress=192.168.0.122,adapternumber=0'
device(00) 3172 eth0                        # address 2E0
device(01) 3172 OFFLINE                     # address 2E1
end LN3172A

LN3172B: cu 3172
interface local(1)
options 'ipaddress=192.168.0.124,adapternumber=0'
device(00) 3172 eth0                        # address 2E2
device(01) 3172 OFFLINE                     # address 2E3
end LN3172B

# VM resources
VM3990: cu 3990
interface local(1)'
device(00) 3390-3 /s390/440RES              # address 400
device(01) 3390-3 /s390/440W01              # address 401
device(02) 3390-3 /s390/440W02              # address 402
device(03) 3390-3 /s390/VMPAG1              # address 403
end VM3990

VM3172: cu 3172
interface local(1)
options 'ipaddress=192.168.0.113,adapternumber=0'
device(00) 3172 eth0                        # address 4E0
device(01) 3172 OFFLINE                     # address 4E1
end VM3172

VM3480: cu 3422                             # Stanza name implies a 3480 drive.
interface local(1)                          # This misnaming does not hurt anything.
device(00) 3422 OFFLINE                     # address 460
end VM3480

end Multi
```

Shell script

This shell script (Multi.sh) is used to start our FLEX-*ES* S/390 instance:

```
flexes Multi.syscf
x3270 -model 3 -keymap pc -port tn3270 localhost:mstcon &
x3270 -model 3 -keymap pc -port tn3270 localhost:L701 &
flexescli localhost Multi
```

Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this redbook.

IBM Redbooks

For information about ordering these publications, see “How to get IBM Redbooks” on page 95. Note that some of the documents referenced here may be available in softcopy only.

- *EFS Systems on a Linux Base: Getting Started*, SG24-7007
- *EFS Systems on a Linux Base: Additional Topics*, SG24-7008

Other publications

These publications are also relevant as further information sources:

- *z/VM: CMS Planning and Administration*, SC24-6042
- *z/VM: CMS Command and Utility Reference*, SC24-6010
- *z/VM: System Messages and Codes—CMS*, GC24-6031
- *z/VM: CMS User’s Guide*, SC24-6009
- *z/VM: CP Planning and Administration*, SC24-6043
- *z/VM: CP Command and Utility Reference*, SC24-6008
- *z/VM: System Messages and Codes—CP*, GC24-6030
- *z/VM: Guide for Automated Installation and Service*, GC24-6064
- *z/VM: System Operation*, SC24-6000
- *z/VM: XEDIT Command and Macro Reference*, SC24-5973
- *z/VM: XEDIT User’s Guide*, SC24-5972

Online resources

This Web site is also relevant as a further information source:

- Fundamental Software, Inc., of Fremont, California

  http://www.funsoft.com

How to get IBM Redbooks

You can search for, view, or download Redbooks, Redpapers, Technotes, draft publications and Additional materials, as well as order hardcopy Redbooks or CD-ROMs, at this Web site:

ibm.com/redbooks

Help from IBM

IBM Support and downloads

ibm.com/support

IBM Global Services

ibm.com/services

Index

I

L

M

N

O

P

Q

R

S

T

U

V

W

X

Y

Z

(0.2”spine)
0.17”<->0.473”
90<->249 pages

z/VM on an EFS Base: Getting Started

ITSO/EFS Project

Practical introduction to simple z/VM usage

Specific examples using Linux and z/OS virtual machines

Assumes use of a specific z/VM implementation

This Redbook provides a basic introduction to z/VM usage in a specific environment and assumes the use of a particular z/VM implementation (which is described in detail). With this starting point, we discuss the initial structure of z/VM and describe a number of common tasks (step by step) along with the rationale for these tasks. The user is assumed to have a common z/OS implementation or a common Linux for S/390 system, or both, already installed. We assume that the total operation is on an IBM ThinkPad with the FLEX-ES product installed.

INTERNATIONAL TECHNICAL SUPPORT ORGANIZATION

BUILDING TECHNICAL INFORMATION BASED ON PRACTICAL EXPERIENCE

IBM Redbooks are developed by the IBM International Technical Support Organization. Experts from IBM, Customers and Partners from around the world create timely technical information based on realistic scenarios. Specific recommendations are provided to help you implement IT solutions more effectively in your environment.

For more information:
ibm.com/redbooks

SG24-7009-00 ISBN 0738498076